Uvod u teorijsko računarstvo

Zoran Ognjanović
Nenad Krdžavac

Beograd – Kragujevac, 2004.

Sadržaj

1

Uvod

Tekst ove knjige nastao je pre svega na osnovu sadržaja kursa "Algebra i logika u računarstvu" koji prvi autor drži u petom i šestom semestru grupe za matematiku Prirodno-matematičkog fakulteta u Kragujevcu i u kome se proučava teorijsko računarstvo[1], matematička disciplina koja se bavi modelima izračunavanja, veštačkom inteligencijom[2], formalnim jezicima koji se koriste u programiranju itd. Pored toga, knjiga sadrži i prikaz nekoliko originalnih naučnih rezultata koje je prvi autor (kao samostalni autor ili koautor) publikovao u vodećim svetskim časopisima iz oblasti teorijskog računarstva, dok je poglavlje 6 izvod iz magistarske teze drugog autora koji je zajedničkim radom prilagođen za potrebe ove knjige. Materija izložena u knjizi omogućava razumevanje specifičnih tema u računarstvu i uspostavljanje osnovnih matematičkih paradigmi. Oblasti koje se obrađuju su:

- teorija algoritama,
- matematička logika,
- teorija formalnih jezika i automata i
- složenost izračunavanja[3].

Teorijsko računarstvo svoje korene ima u teoriji algoritama (tj. izračunljivih funkcija), oblasti koja je nastala između 1930. i 1940. godine, dakle pre razvoja digitalnih računara, kao rezultat pretresanja osnova matematike zbog paradoksa koji su se pojavili krajem XIX i početkom XX veka. U razmatranju strogog zasnivanja matematike, postavljalo se pitanje[4] da li postoji opšti postupak utvrđivanja istinitosti matematičkih iskaza. Ovo pitanje vodi poreklo još od Gottfried-a Leibnitz-a (Lajbnić, 1646 – 1716) koji je u XVII veku, nakon uspešne konstrukcije mehaničke

[1] Theoretical computer science.

[2] Artificial intelligence.

[3] Computational complexity.

[4] Pitanje je poznato pod nemačkim nazivom *Entscheidungsproblem*, tj. problem odlučivanja. Nezavisno jedan od drugog, Čerč i Tjuring su negativno odgovorili na ovo pitanje, svodeći ga na probleme jednakosti λ-izraza, odnosno utvrđivanja da li će se proizvoljna Tjuringova mašina zaustaviti (halting problem).

računske mašine, razmišljao o konstrukciji mašine koja bi mogla manipulisati simbolima i na taj način odrediti istinitost iskaza. Problem je aktuelizovao David Hilbert (1862 – 1943), najpre na Kongresu matematičara održanom 1900. godine u poznatom desetom problemu, a zatim zajedno sa Wilhelm-om Ackermann-om (Akerman, 1896 – 1962) 1928. godine. Da bi se na ovo pitanje moglo odgovoriti bilo je neophodno precizirati šta se podrazumeva pod postupkom mehaničkog izvođenja. U tom istraživanju je matematička logika imala prevashodnu ulogu, tako da se teorija algoritama, a značajnim delom i teorijsko računarstvo, smatraju njenim disciplinama. Formalnu teoriju izračunljivosti su zasnovali Alonzo Church (Čerč, 1903 – 1995), Kurt Gödel (Gedel, 1906 – 1978), Jacques Herbrand (Erbran, 1908 – 1931), Stephen Kleene (Klini, 1909 – 1994), Emil Post (1897 – 1954), Alan Turing (Tjuring, 1912 – 1954), Андрей Марков (1903 – 1979) itd. čiji su rezultati imali značajan uticaj kako na teorijske, tako i na praktične aspekte razvoja računarstva. To se odnosi na principe programibilnog digitalnog računara opšte namene, koncept pisanja programa kao liste naredbi u formalnom jeziku, interpretiranje i prevođenje programa, razvoj programskih jezika uopšte itd.

Matematička logika pruža formalni jezik za opisivanje i oruđa za analizu problema koji se istražuju u računarstvu, a i sama je poligon za primenu rezultata iz te oblasti, recimo za automatske dokazivače teorema. Razne klasične i neklasične logike i drugi formalni sistemi se koriste u sistemima zaključivanja, poput ekspertnih sistema i verifikaciji programa i elektronskih sklopova računara. Jedan od ciljeva razvoja matematičke logike je konstrukcija dovoljno jakih sistema koji će biti u stanju da formalizuju sve šire oblasti ljudskog mišljenja, tako da njihova složenost bude u granicama tehnološke ostvarljivosti. Rezonovanja o znanju se pokazalo korisnim u veštačkoj inteligenciji, ali i u teoriji igara i analizi paralelnih računarskih sistema. I druge oblasti matematičke logike nalaze primene u računarstvu. Recimo, jedan efikasno implementiran segment klasične predikatske logike prvog reda se nalazi u osnovi sistema baza podataka i jezika SQL[5]. Teorija tipova čini okvir za razvoj i analizu programskih jezika u kome se pogodno prikazuju napredni koncepti savremenih jezika, poput nasleđivanja i polimorfizma, omogućava rezonovanje o programima i predlažu nove tehnike za kreiranje efikasnijih prevodilaca. Ideja o programskom jeziku Prolog i logičkom programiranju proizašla je iz istraživanja na polju automatskog dokazivanja teorema. Uopšte, postojeća sredstva matematičke logike u toj meri zauzimaju centralno mesto u računarstvu da se logika često smatra računom računarstva, čak i u većoj meri nego što je to matematika za prirodne nauke.

Osnivač teorije formalnih jezika i automata je Noam Chomsky (Čomski, 1928). U njoj se raspravljaju pitanja opisivanja formalnih jezika i strukture određenih klasa formalnih jezika. Formalni jezici i njima odgovarajući automati predstavljaju, između ostalog, osnov za kreiranje i implementaciju programskih jezika, ali i za razne druge algoritme prepoznavanja oblika, verifikacije itd.

Teorija složenosti izračunavanja je jedna od mlađih grana teorijskog računarstva proizašla iz analize funkcija u teoriji izračunljivih funkcija u kojoj se ispituje koliko su i zašto neki zadaci teški, tj. koliko je vremena i prostora potrebno da bi bili rešeni. Na ovo pitanje nije u potpunosti odgovoreno, ali značajan rezultat u formi elegentne klasifikacije problema u odnosu na složenost nam pruža argumente da sa

[5]Structured Query Language.

velikom pravom verujemo da su neki problemi jako teški za izračunavanje, mada to, možda nismo u stanju da precizno dokažemo. Potom se pruža mogućnost analize problema kako bi se pronašao uzrok njegove težine i da li je neka alternativna formulacija pogodnija za rešavanje ili postoji prihvatljiv postupak približnog rešavanja. Teorija složenosti ima suštinske primene u kriptologiji gde je njen zadatak donekle obrnut, naime traga se za problemima koji su teški kako bi bili korišteni u zaštiti podataka i računarskih resursa.

Danas se, iz raznih razloga, u prvi plan ističe razvoj tehnologija na kojima se baziraju računari. Imajući u vidu do sada spomenuto treba reći da je to jedna vrsta pojednostavljivanja, pa i zamagljivanja, stvari kojoj je sklona propaganda industrije računara i računarskih programa potpomognuta popularnim medijima. Pre svega teorijski rezultati su ono oruđe koje inspiriše i usmerava rad u računarstvu dajući sasvim nove ili, pak, elegantnije postupke. Poseban značaj teorijsko računarstvo ima u verifikaciji praktičnih rešenja. Dovoljno je samo setiti se čuvenih grešaka u Intel-ovim procesorima do kojih verovatno ne bi došlo da su primenjeni postupci zasnovani na teorijskim dostignićima o kojima će biti reči u ovoj knjizi. Takvih i sličnih primera ima na pretek, tako da je sasvim izvesno da će u budućnosti, kako je to i do sada bio slučaj, oblasti kojima će u nastavku teksta biti posvećena pažnja predstavljati glavne oslonce računarskih nauka i sa praktične i sa teorijske strane.

Na osnovu prethodnih napomena može se steći tek delimična predstava o dostignućima disciplina teorijskog računarstva, njihovoj raznolikosti i isprepletenosti. I pored svega postignutog, ova oblast se nalazi daleko od toga da bi bila okarakterisana kao da je poslednja reč u njoj data. Veliki broj problema, od kojih neke možemo nazvati i osnovnim, tek treba rešiti. Dalji razvoj teorijskog računarstva leži u takvim istraživanjima, kao i u proučavanju pitanja koja stalno izviru u svakodnevnom radu ogromnog broja projektanata, programera i korisnika računara.

Tekst knjige je podeljen na sledeća poglavlja: Izračunljivost. Odlučivost, Klasična iskazna logika, Bulove algebre, Predikatska logika prvog reda, Opisne logike, Neklasične logike, Verovatnosne logike, Teorija formalnih jezika i Teorija složenosti izračunavanja, nakon kojih slede (delimično rešeni) zadaci iz prethodno obrađenih oblasti i spisak korištene literature i indeks pojmova.

Poglavlje Izračunljivost. Odlučivost sadrži prikaz osnovnih rezultata teorije algoritama. Najpre su ukratko opisani intuitivni pojam algoritma i formalni modeli izračunavanja, a zatim detaljno model Tjuringove mašine i klasa parcijalno rekurzivnih funkcija, dokaz ekvivalentnost ovih modela izračunljivosti i Čerčova teza. Poslednji deo poglavlja posvećen je odlučivosti i aritmetičkoj hijerarhiji složenosti. Dokazano je više osnovnih teorema u vezi klasifikacije odlučivih i neodlučivih predikata. U ovom, ali isto tako i u sledećim poglavljima, kao veoma važan pojavljuje se koncept nedeterminističkog izračunavanja. Utisak mi je da koncept nedeterminizma nije u dovoljnoj meri blizak ni iskusnijim istraživačima sa naših prostora zbog čega će na više mesta biti posebno istaknuta njegova uloga.

U poglavljima Klasična iskazna logika, Bulove algebre, Predikatska logika prvog reda i Opisne logike izlaže se materija u vezi iskazne i predikatske klasične logike. Najpre se klasična iskazna logika analizira sa stanovišta modela, definišu se pojmovi interpretacije, zadovoljivosti, tautologija i semantičkih posledica. Razmatraju se različite normalne forme iskaznih formula, a posebno je opisana definiciona forma koju (u najgorem slučaju) karakteriše polinomijalno veća dužina u

odnosu na dužinu polazne formule. Zatim su dati osnovni pojmovi u vezi formalnih sistema i aksiomatizacije iskaznog računa i dokazana teorema proširene potpunosti pristupkom Henkina u kome se konstruiše maksimalno konzistentno proširenje konzistentnog skupa formula i uz pomoć teoreme dedukcije pokazuje da se takav skup koristi u definisanju modela. Opisane su dve procedure pogodne za automatsko dokazivanje teorema (rezolucija i metoda tabloa) i dokazana njihova potpunost. Ispitivanje zadovoljivosti iskaznih formula predstavlja jedno od glavnih pitanja primena matematičke logike. Jednom specifičnom pristupu tom problemu, preko genetskih algoritama, je posvećen deo poglavlja u kome se izlažu neki originalni rezultati prvog autora. Deo teksta o iskaznoj logici završava se izlaganjem o Bulovim algebrama sa osvrtom na metodu BDD koja se koristi u logičkoj sintezi i verifikaciji elektronskih kola. Pojmovi uvedeni u iskaznom slučaju se uopštavaju u delu posvećenom predikatskoj logici prvog reda, nakon čega se dokazuje teorema neodlučivosti za ovu logike. Dokaz Erbanove teoreme predstavlja uvod za prikaz metode rezolucije u predikatskoj logici, a poglavlje se završava uopštenjem iskazne metode tabloa. Konačno, u poglavlju Opisne logike uvodi se jedna klasa logika novijeg datuma čije proučavanja je inspirisano primenama u semantičkom web-u, bazama podataka itd.

Na početku poglavlja Neklasične logike se pomoću Kripkeovih modela opisuje semantika modalnih operatora i klasifikacija modalnih logika u zavisnosti od tipa relacije dostižnosti. Potom se razmatraju i neke posebne vrste modalnih logika, poput temporalne logike, logike znanja. U nekoliko primera se prikazuje primena ovih logika u automatskom zaključivanju i proveri ispravnosti programa. Opisana je i generalizacija metode tabloa za modalne logike, takozvana metoda prefiksiranih tabloa, a takođe i originalna metoda dualnih tabloa koju je prvi autor predložio u magistarskoj tezi i kasnije publikovao. U drugom delu poglavlja se prikazuju logike alternativne klasičnoj logici: nemonotone logike, intuicionistička i viševrednosna logika.

Poglavlje Verovatnosne logike sadrži prikaz originalnih rezultata iz doktorske disertacije prvog autora koji su kasnije razrađeni i publikovani u relevantnim međunarodnim časopisima. Verovatnosne logike su posebna vrsta neklasičnih logika koje imaju niz sličnosti sa modalnim logikama, a omogućavaju strogo, formalno, zaključivanje o verovatnoći.

U poglavlju Teorija formalnih jezika analiziraju se klase jezika definisane hijerarhijom Čomskog. Definisani su pojmovi formalnih gramatika i apstraktnih automata i pokazana ekvivalencije klasa jezika definisanih gramatikama i klasa koje prepoznaju odgovarajući tipovi automata. Takođe, razmatra se odlučivost različitih problema iz ove oblasti, poput problema pripadanja reči jeziku neke od klasa u hijerarhiji. Primene teorije formalnih jezika ilustrovane su opisima programskih sistema Lex i Yacc za automatsko generisanje leksičkih, odnosno sintaksnih, analizatora. Poglavlje se završava opisom konačnih automata nad beskonačnim rečima i njihovom primenom u automatskoj verifikaciji.

Poglavlje Teorija složenosti izračunavanja, počinje razmatranjem apstraktne teorije složenosti koja ne zavisi od vrste resursa koji se procenjuje. Zatim se analizira teorija složenosti bazirana na kategorizaciji u odnosu na vreme i upotrebljeni prostor prilikom izračunavanja. Opisane su osnovne klase složenosti: L, NL, P, NP, PSPACE i EXP i njihovi karakteristični problemi. Posebno je analiziran prob-

lem odnosa klasa P i NP što se smatra jednim od osnovnih pitanja savremenog računarstva. Dokazano je da problem SAT, utvrđivanje zadovoljivosti iskaznih formula, predstavlja NP-kompletan problem. Opisane su i manje poznate, ali veoma značajne, verovatnosne klase složenosti RP i BPP, sistemi interaktivnih dokaza i protokoli bez prenosa znanja, a zatim i njihova primena u kriptologiji. Poglavlje se završava kratkim prikazom drugog pristupa problemu složenosti zasnovanom na radovima Kolmogorova.

Konačno, poslednje poglavlje Zadaci sadrži delimično rešene zadatke iz prethodno prikazanih oblasti teorijskog računarstva u kojima se primenjuju opisane tehnike i razrađuju dokazi koji su u tekstu samo skicirani.

Iako po prirodi pisanja dužih tekstova razdvojene u posebna poglavlja, oblasti teorijskog računarstva su međusobno tesno povezane tako da se često rezultati iz jedne koriste u drugim oblastima. Na pojedine takve primere ukazuje se u samom tekstu. Usled velikog obima materijala u narednim poglavljima ponegde će biti dati samo iskazi tvrđenja, a povremeno će dokazi biti prepričani bez detaljisanja kako bi čitalac mogao steći predstavu o korištenoj metodologiji. Jedan od suštinskih delova teksta, na koji čitaocima skrećemo posebnu pažnju, su primeri koji ilustruju stvarne primene pojedinih oblasti teorijskog računarstva. Na kraju svakog poglavlja nalazi se odeljak Za dalje proučavanje u kome se ukazuje na literaturu posvećenu upravo izloženoj materiji. Ti tekstovi pružavaju daleko detaljnije informacije o nastanku, trenutnom stanju i otvorenim problemima odgovarajućih oblasti nego što je to bilo moguće izložiti u ovoj knjizi. Zbog toga iskreno preporučujemo zainteresovanim čitaocima da obrate pažnju na te izvore tim pre što im je često moguće pristupiti preko Interneta.

Značajan deo teksta u poglavljima Izračunljivost. Odlučivost i Teorija formalnih jezika napisan je na osnovu beleški sa predavanja dr Aleksandra Jovanovića, dr Žarka Mijajlovića i dr Zorana Markovića koje je prvi autor pratio u okviru redovnih studija na Matematičkom fakultetu u Beogradu tokom 1986. i 1987. godine. Zahvaljujemo se recenzentima, dr Žarku Mijajloviću, profesoru Matematičkog fakuleteta u Beogradu, dr Zoranu Markoviću, višem naučnom saradniku Matematičkog instituta SANU, dr Dragiću Banković, profesoru Prirodno-matematičkog fakulteta u Kragujevcu, i dr Miodragu Raškoviću, profesoru Učiteljskog fakulteta u Beogradu koji su su svojim komentarima doprineli značajnim poboljšanjima teksta. Zahvaljujemo se i dr Đorđu Kadijeviću, docentu Univerziteta Megatrend, čiji su metodološki saveti unapredili formu i jezik izlaganja. Sa druge strane, autori snose punu odgovornost za sve greške u tekstu.

1.1 Za dalje proučavanje

U [15, 47] se na pregledan način prikazuju teme spomenute u ovoj glavi, kao i veliki broj drugih interesantnih referenci. Tekstovi se mogu pronaći na Internet-adresi http://www.math.ucla.edu/˜ asl/bsl/0702-toc.htm.

2

Izračunljivost. Odlučivost

Rešavanje problema razvojem algoritama[1] i pisanjem programa je jedan od osnovnih zadataka u računarstvu. Recimo, potrebno je izračunati sumu ulaznih veličina ili urediti članove niza u rastućemo poretku. Međutim, iako probleme treba rešavati, nisu samo oni mogući predmet razmatranja. Matematičkim sredstvima proučavaju se i sami postupci rešavanja, algoritmi.

Formalni modeli izračunavanja koje ćemo razmatrati biće Tjuringove mašine i parcijalno rekurzivne funkcije. Iako naizgled različiti, ovi modeli određuju jednu te istu klasu algoritama što navodi na tezu da ti modeli izračunavanja upravo određuju granice mogućnosti mehaničkog izračunavanja. Te granice razdvajaju klase problema na one na za koje, u principu, postoji mogućnost programskog rešavanja i one za koje to nije slučaj, tesno povezujući pojmove izračunljivosti i odlučivosti.

2.1 Intuitivni pojam algoritma

Pojam algoritama spada, poput geometrijskih pojmova kao što su tačka ili prava, u osnovne matematičke koncepte i kao takav se ne definiše. Međutim, sledeći opšti uslovi se, prihvataju kao kriterijumi za nazivanje nekog postupka algoritmom (efektivnom procedurom):

- postupak se opisuje konačnim nizom jednostavnih naredbi[2],
- postoji idealna mašina (računar) koja izvršava te naredbe prema unapred utvrđenim pravilima,
- ta mašina započinje izračunavanje u nekom inicijalnom stanju; primenjena na ulazne podatke mašina izvršava naredbe u diskretnim koracima u kojima menja svoja stanja,
- izvršavanje svake naredbe se izvodi u konačnom vremenu pri čemu se koristi konačan memorijski prostor,

[1] Reč algoritam je nastala od imena persijskog matematičara Abu Džafar Mohamed ibn Musa al-Hovarizmij-a (780 – 850).

[2] Primetimo da algoritmi mogu biti izraženi više ili manje detaljno.

- izvršavanje naredbe je determinističko: iz jednog stanja izvršavanjem iste naredbe mašina uvek prelazi u isto stanje i
- prelaskom u završno stanje mašina prestaje sa izračunavanjem.

Osobina determinisanosti izvršavanja naredbi se drugačije može formulisati kao mogućnost ponavljanja izvršavanja algoritama. Ako ga prihvatimo, postupci koji uključuju slučajnost[3] ne spadaju u algoritme. U pojedinim slučajevima mi ćemo odbaciti ovaj uslov i razmatrati i nedeterminističke algoritme.

Primetimo da se među navedenim uslovima ne nalazi zahtev da se algoritam uvek završi, tj. da se rezultat uvek dobije u konačnom vremenu[4], odnosno da se ne zahteva da se dobije odgovor za sve moguće ulazne podatke, dok se taj zahtev postavlja za svaki pojedinačni korak izvršavanja. Slično je i sa zahtevom za ukupno memorijsko zauzeće. Kao što ćemo u nastavku teksta videti ovakav pristup u teorijskim razmatranjima pruža pogodnosti za elegantno opisivanje formalnih metoda.

Algoritam predstavlja opis funkcije koja ulazne podatke preslikava u odgovor. Funkcije za koje postoje algoritmi zato nazivamo algoritamskim funkcijama (efektivnim funkcijama, izračunljivim funkcijama).

2.2 Formalni modeli izračunavanja

Poznat je veliki broj algoritama. Na primer, to su postupak za množenje celih brojeva, tablični metod ispitivanja da li je neka iskazna formula tautologija, Euklidov algoritam nalaženja najvećeg zajedničkog delioca dva broja itd. Za probleme za koje poznajemo postupke rešavanja lako utvrđujemo da jesu algoritamski rešivi. Međutim, kako se napreduje u razvoju matematike, nailazi se na probleme za koje nismo u stanju da damo rešenje. Postavlja se pitanje da li je to samo posledica naše nesposobnosti ili je reč o principijelnoj nemogućnosti. Da bi se na to pitanje odgovorilo potrebno je formalno precizirati pojmove algoritma i izračunljivih funkcija, čime bi se jasno odredila za sada dosta nejasna ideja o granicama efektivnosti, tj. dosega algoritama.

Problem postojanja efektivnog postupka za utvrđivanje da li proizvoljna diofantovska jednačina $p(x_1, \ldots, x_m) = 0$ ima nenegativna celobrojna rešenja je primer za ovakvu situaciju. U prethodnoj jednačini $p(x_1, \ldots, x_m)$ je polinom sa celobrojnim koeficijentima i promenljivim x_1, $\ldots$, x_m, recimo $x_1^4 x_3 - 3x_2^5 + 6$. Sa jedne strane, nabrajanjem svih m-torki prirodnih brojeva i proverom da li predstavljaju nule polinoma bi se, pre ili posle, stiglo do rešenja jednačine, ako ono postoji. Međutim, kako neke jednačine ovog tipa, recimo $x^2 - 2 = 0$, nemaju rešenja, prethodno opisani postupak se u takvim slučajevima ne bi nikada završio, zbog čega i ne predstavlja rešenje problema. Provera postojanja rešenja diofantovskih jednačina je zapravo ekvivalentna formulacija desetog problema koji je postavio David Hilbert u svom istorijskom predavanju na Kongresu matematičara održanom 1900. godine u Parizu. Primetimo da svaki eventualni odgovor na ovo pitanje mora na neki način ponuditi

[3] Recimo, postupci u kojima prelazak sa jednog na drugi korak zavisi od dogadjaja kao što su dobijena strana prilikom bacanja novčića. Postupci takvog tipa su nedeterministički.

[4] Ovakav zahtev bi se mogao nazvati konačnost, finitnost, algoritma. Videti s tim u vezi odeljak 2.10.8.

i formalnu definiciju onoga što se podrazumeva pod efektivnim postupkom, bilo u smislu da ponuđeno rešenje potpada pod tu definiciju, bilo da ne postoji rešenje sa zahtevanim svojstvima. Formalna definicija efektivnog postupka pojavila se razvojem teorije algoritama, dok je Jurij Matijaševič 1970. godine sredstvima razvijenim u okviru te teorije negativno rešio sam problem, o čemu će biti više reči u primeru 2.8.15.

U razvoju teorije algoritama ponuđeno je više pristupa formalizaciji ovih granica:

- Sistem izračunljivosti predstavljen u formalnom sistemu aritmetike je predložio Gedel između 1931. i 1934. godine, pri čemu se funkcija f smatra izračunljivom ako za svako m i n za koje je $f(m) = n$, u formalnom sistemu važi $\vdash f(m) = n$.

- Prikazivanje izračunljivih funkcija kao jedinstvenih rešenja sistema funkcionalnih jednačina je u istom periodu opisao takođe Gedel, a prema ideji Erbrana.

- λ-račun koji je razvio Čerč do 1936. godine je jednostavan formalni jezik za koji se definiše pojam redukcije koji predstavlja izračunavanje, a funkcija je izračunljiva ako se može opisati u jeziku.

- Aritmetički opis zasnovao je Klini takođe do 1936. godine, a baziran je na generalisanom pojmu definisanja indukcijom.

- Sistemi zasnovani na automatima, među kojima su:

  – Tjuringove mašine iz 1936. godine,
  – Postova mašine predstavljena takođe 1936. godine,
  – Neograničena registarska mašina[5] koju su Shepherdson (Šeferdson) i Sturgis (Stargis) opisali 1963. godine,

  formalizuju pojam algoritma opisujući idealne modele računara[6]. Zanimljivo je da su neki sistemi dati pre nastanka digitalnih računara.

- Sistemi produkcija (nekad se nazivaju i sistemi sa prezapisivanjem[7]), među kojima su:

  – Postovi sistemi iz 1943. godine,
  – Markovljevi algoritmi uvedeni 1954. godine i
  – Gramatike Čomskog predložene 1956. godine,

  su jedna vrsta formalnih sistema u kojima se opisuju moguće transformacije (pravila izvođenja) jednih u druge reči na unapred fiskiranom alfabetu. Funkcije se opisuju kao skupovi parova reči (u, v) za koje postoji niz reči koje se dobijaju počev od u primenama pravila izvođenja i koji završava rečju v.

[5] Unbounded Register Machine, URM.

[6] Bez obzira na upotrebu reči *mašina* koja se javlja u nazivima ovih formalizama, uvek treba imati u vidu da se ovde radi o apstraktnim matematičkim konceptima, a ne realnim fizičkim objektima.

[7] Rewriting systems.

- *while*-programi su vrsta notacije proizašle iz ideja Goldstine-a (Goldštajna) i János-a (John) von Neumann-a (Fon Nojman, 1903 – 1957) o algoritamskim šemama[8] kao formalizmu za prikazivanje izračunljivih funkcija. *while*-programi se sastoje samo od naredbi dodeljivanja, nizanja naredbi i *while*-naredbi.

Njihovi izvori inspiracije se međusobno značajno razlikuju, ali se pokazuje da su sistemi međusobno ekvivalentni. Ovo, kao i neuspeh pokušaja konstrukcije zadatka i postupka njegovog rešavanja koji ne potpadaju pod ove klasifikacije daje za pravo verovanju da je postignut nekakav apsolutni koncept i da se svi algoritmi mogu izraziti u svakom od ovih sistema, što je formulisano Čerčovom tezom koja se razmatra u odeljku 2.6.

2.3 Tjuringova mašina

2.3.1 Model mašine za izračunavanje

Digitalni računar se na apstraktnom nivou obično prikazuje kao celina sastavljena od procesora, memorije i ulazno-izlaznih uređaja. Procesor iz memorije pribavlja naredbe i podatke nad kojima se vrši obrada u skladu sa značenjem naredbi, a dobijeni rezultati se vraćaju u memoriju. Preko ulazno-izlaznih uređaja podaci koji će biti obrađeni se unose u memoriju, odnosno iz memorije se preuzimaju rezultati obrade i prikazuju na odgovarajući način. Komunikacija delova računara se obavlja preko magistrala.

Tjuringova mašina je preteča ovakvog modela računara, pri čemu su neka svojstva idealizovana. To se pre svega odnosi na memoriju za koju se pretpostavlja da je potencijalno beskonačna. Preciznije, na početku izvršavanja Tjuringove mašine zauzet je samo konačan broj memorijskih registara, a isto važi i u svakom koraku izračunavanja. Ne postoji ograničenje koliki je taj konačan broj registara. U svakom koraku izračunavanja moguće je i zahtevati novi, do tada neiskorišteni memorijski registar i svaki takav zahtev se ispunjava. Sa druge strane, Tjuringova mašina je restrikcija koncepta savremenog računara u smislu operacija koje je u stanju da izvršava, a koje su elementarne. Kako ćemo videti, zanimljivo je da su te operacije ipak dovoljne za opisivanje proizvoljnih algoritama. Njihova prednost u odnosu na bogatije programske jezike je upravo u jednostavnosti koja olakšava analizu.

U teorijskom računarstvu se, inače, razmatraju i druge, slabije, klase mašina koje su restrikcije druge vrste u odnosu na aktuelne računare: neki modeli nemaju memoriju (konačni automati[9]) ili je memorija organizovana na poseban način (stek kod potisnih automata[10]), ulazno-izlazni podaci su ograničeni na reči[11], neki čak nemaju izlaz (konačni automati). O ovim modelima biće reči u poglavlju 9.

[8] Flowchart.
[9] Finite automata.
[10] Push-down automata.
[11] String.

2.3.2 Alfabet

Pre opisa Tjuringove mašine ukratko ćemo izložiti nekoliko pojmova koji će biti detaljnije opisani u poglavlju 9.

Svaki problem se izražava u nekom jeziku. Alfabet je skup znaka koji su nedeljive celine. Reč na nekom alfabetu je bilo koja konačna sekvenca znaka tog alfabeta. Sekvenca od nula znaka se naziva prazna reč. Reči se razdvajaju znakom blanko koji se ne smatra delom alfabeta već pomoćnim simbolom. Jezik je neki podskup skupa svih reči odgovarajućeg alfabeta. Reč t je podreč reči q ako postoje, možda i prazne, reči u i v tako da je $q = utv$.

Obično je alfabet konačan skup znaka, jer sve što se može iskazati beskonačnim prebrojivim alfabetom $\{a_1, a_2, \ldots\}$ može se iskazati i najjednostavnijim, unarnim, alfabetom $A = \{1\}$. Reči ovog alfabeta: 1, 11, 111, ...se mogu identifikovati sa znacima proizvoljnog beskonačnog alfabeta. U nastavku ovog poglavlja, sem ako se posebno ne naglasi, koristićemo unarni alfabet $A = \{1\}$. Pored simbola 1 koristićemo i blanko-znak za čije označavanje ćemo zbog preglednosti upotrebljavati znak 0.

2.3.3 Neformalni opis Tjuringove mašine

Tjuringova mašina se sastoji od:

- *trake* podeljene u ćelije, memorijske registre, koja se neograničeno pruža na levo i desno; broj ćelija (tj. dužina trake) je neograničen; sadržaj svake ćelije je ili znak 1 ili blanko znak (znak 0),
- *glave* koja se uvek nalazi nad tačno jednom ćelijom trake i može:
  - pročitati sadržaj ćelije nad kojom se nalazi i
  - upisati u ćeliju nad kojom se nalazi znak 1 ili 0 (blanko znak, tj. obrisati ćeliju) ili pomeriti se za jedan korak u levo ili u desno u odnosu na trenutnu poziciju,
- *indikatora stanja mašine.*

Tjuringova mašina se u svakom trenutku nalazi u tačno jednom od konačno mnogo stanja koje se eventualno menja nakon svakog koraka izračunavanja. Skup svih stanja mašine označićemo sa $S = \{q_0, q_1, \ldots\}$. Izvršavanje mašine se izvodi pod dejstvom programa koji čini neki konačan niz naredbi. Svaka naredba je četvorka oblika:

$$q_i \ s \ o \ q_j$$

gde su q_i i q_j neka stanja iz skupa S, s je znak nad kojim se nalazi glava mašine, a $o \in \{1, 0, L, R\}$ je oznaka operacije. U svakom koraku rada mašina analizira stanje u kojem se nalazi i sadržaj ćelije nad kojom je glava, a zatim izvršava naredbu koja ima odgovarajuće vrednosti parametara q_i i s. Efekat izvršenja naredbe je dvojak. Najpre se, u zavisnosti od vrednosti parametra o obavi:

- ako je $o = 1$, u ćeliju nad kojom se nalazi glava upisuje se znak 1,

- ako je $o = 0$, u ćeliju nad kojom se nalazi glava upisuje se 0, tj. blanko znak,
- ako je $o = L$, glava se pomera ulevo za jednu ćeliju i
- ako je $o = R$, glava se pomera udesno za jednu ćeliju.

Potom mašina menja stanje i prelazi u stanje q_j.

Primeri naredbi su:
q_5 0 1 q_{17},
q_1 0 0 q_2 i
q_0 1 L q_0.
U prvoj naredbi, ako se mašina nalazi u stanju q_5, a glava nad znakom blanko, u ćeliju se upisuje znak 1 i prelazi u stanje q_{17}. U drugoj naredbi, ako se mašina nalazi u stanju q_1, a glava nad znakom blanko, u ćeliju se upisuje blanko znak i prelazi u stanje q_2. Ovakva naredba služi samo za promenu stanja mašine. U trećoj naredbi, ako se mašina nalazi u stanju q_0, a glava nad znakom 1, glava se pomera ulevo, a mašina ostaje u istom stanju.

Primetimo da, ako se želi da mašina radi deterministički, program sme sadržati samo jednu naredbu za svaku kombinaciju stanja q_i i sadržaja s ćelije nad kojom je glava. Na primer, u jednom programu se ne smeju pojaviti sledeće naredbe:
q_4 1 1 q_5 i
q_4 1 L q_2
jer im se vrednosti parametara q_i i s poklapaju, a vrednosti parametara o i q_j razlikuju. U slučaju nedeterminističkih mašina ovaj zahtev ne postoji.

U vezi sa Tjuringovom mašinom prihvatićemo sledeće konvencije. Stanje $q_0 \in S$ nazvaćemo *početnim stanjem*. Inicijalno, mašina se uvek nalazi u početnom stanju. Pri tome traka sadrži samo konačno mnogo ćelija u koje je upisan znak 1, dok sve ostale ćelije sadrže znak 0. Reč se na traci prikazuje kao neprekidan niz ćelija koje sadrže znak 1, a sa leve i desne strane tog niza se nalazi najmanje po jedan blanko znak, tj. znak 0. Po pravilu, na početku i na kraju izvršavanja glava mašine se nalazi iznad najlevlje ćelije koja sadrži znak 1. Skup stanja S proširićemo jednim novi stanjem q_z koje ne pripada do sada razmatranom skupu stanja. Stanje q_z nazvaćemo *završnim stanjem*. Kada se mašina nađe u stanju q_z ona prekida sa izvršavanjem.

O Tjuringovoj mašini možemo razmišljati na dva načina:

- kao o jedinstvenoj mašini koja izvršava sve programe (u smislu savremenog računara) i
- kao o posebnoj mašini za svaki program u kom slučaju se pojam mašine poistovećuje sa pojmom programa, tj. svaki program predstavlja posebnu mašinu

koja se u suštini međusobno ne razlikuju.

Definicija 2.3.1 Pod *konfiguracijom* Tjuringove mašine podrazumevamo opis koji sadrži: opis sadržaja trake, položaj glave i stanje mašine. *Standardna konfiguracija* je konfiguracija u kojoj je:

- ili traka prazna (tj. sve ćelije sadrže blanko znak) ili sadrži najviše konačno mnogo nepraznih reči razdvojenih po jednim blanko znakom,

$\ldots 01q_0 1000\ldots$ $\ldots 0111q_3 10\ldots$
$\ldots 011q_0 000\ldots$ $\ldots 011q_3 110\ldots$
$\ldots 0110q_0 00\ldots$ $\ldots 01q_3 1110\ldots$
$\ldots 0111q_1 00\ldots$ $\ldots 0q_3 11110\ldots$
$\ldots 01110q_2 0\ldots$ $\ldots 01q_z 1110\ldots$
$\ldots 01111q_3 0\ldots$

Slika 2.1. Izvršavanje Tjuringove mašine iz primera 2.3.2.

- glava mašine je iznad prve (gledano sa leva) ćelije trake koja sadrži znak 1 i
- ako počinje sa izvršavanjem, mašina se nalazi u početnom stanju q_0, a ako završava sa radom u završnom stanju q_z.

Sada se programi mogu shvatiti kao funkcije koje preslikavaju skup konfiguracija mašine u samog sebe.

Primer 2.3.2 Neka je na traci data samo jedna reč sastavljena od jedinica (a sve ostale ćelije sadrže znak 0) nad čijim krajnjim levim znakom se nalazi glava. Sledeći program dopisuje dva znaka 1 sa desne strane reči, a zatim se glava vraća na levo, na početak reči, nakon čega mašina staje:

$q_0\ 1\ R\ q_0$	glava se pomera udesno, na kraj reči
$q_0\ 0\ 1\ q_1$	na mestu prve 0 upisuje se 1
$q_1\ 1\ R\ q_2$	glava se pomera udesno
$q_2\ 0\ 1\ q_3$	na mestu druge 0 upisuje se 1
$q_3\ 1\ L\ q_3$	glava se pomera ulevo
$q_3\ 0\ R\ q_z$	do prve 0, ide udesno i zaustavlja se

Izvršavanje ove Tjuringove mašine, pod pretpostavkom da je traka na početku sadržala binarnu reč 11, prikazano je na slici 2.1. Pozicija oznake stanja (q_i) na slici predstavlja položaj glave trake, tj. glava je iznad ćelije u kojoj se nalazi cifra levo od oznake stanja. ■

Primetimo da se u opisu Tjuringove mašine ne kaže šta se događa ako za sadržaj ćelije nad kojim se nalazi glava i tekuće stanje mašine u programu ne postoji odgovarajuća naredba. Ova situacija bi odgovarala 'zaglavljivanju' programa pisanih na standardnim programskim jezicima i može se formalizovati kompletiranjem programa naredbama koje u takvim situacijama ne menjaju ni stanje, ni poziciju glave, ni sadržaj ćelije nad kojom se glava nalazi. Recimo, ako u programu ne postoji naredba koja odgovara situaciji kada je mašina u stanju q_0, a sadržaj ćelije nad kojom se nalazi glava 0, možemo program proširiti naredbom:

$q_0\ 0\ 0\ q_0$

koja predstavlja jednu beskonačnu petlju. S obzirom na ovakvu mogućnost, na dalje nećemo voditi računa da program bude u opisanom smislu kompletan.

2.3.4 Tjuringove mašine i funkcije

U ovom odeljku ćemo opisati kako se Tjuringove mašine mogu iskoristiti kao algoritmi, tj. za izračunavanje funkcija koje preslikavaju prirodne brojeve u prirodne

brojeve.

Definicija 2.3.3 *Aritmetička funkcija* je preslikavanje f za koje važi:

- domen preslikavanja, $Dom(f)$, je podskup skupa $\mathbb{N}^k$ $(k > 0)$ i
- kodomen preslikavanja, $Im(f)$, je podskup skupa $\mathbb{N}$.

Ako je za neki $k > 0$, $Dom(f) = \mathbb{N}^k$, f je *totalna funkcija*. Ako je $Dom(f) \subset \mathbb{N}^k$, za neki $k > 0$ i $Dom(f) \neq \mathbb{N}^k$, f je *parcijalna funkcija*.

Definicija 2.3.4 *Unarna reprezentacija prirodnog broja n* u unarnom alfabetu $A = \{1\}$ je reč koja sadrži $n + 1$ znak 1.

Definicija 2.3.5 Neka je f aritmetička funkcija oblika $f : X \to \mathbb{N}$, gde je $X \subset \mathbb{N}$. Funkcija f je *Tjuring-izračunljiva* ako postoji program P za Tjuringovu mašinu tako da je za svaki $m \in X$:

- pre početka izvršavanja programa P Tjuringova mašina u standardnoj konfiguraciji, pri čemu je jedina reč zapisana na traci unarna reprezentacija broja m i
- po završetku rada programa P Tjuringova mašina u standardnoj konfiguraciji, pri čemu je jedina reč zapisana na traci unarna reprezentacija broja $f(m)$.

Program P tada *izračunava* funkciju f.

Primetimo da su prema definiciji 2.3.5 Tjuring-izračunljive funkcije parcijalne, odnosno ako se neki m ne nalazi u domenu Tjuring-izračunljive funkcije f, odgovarajući program P ne staje.

Primer 2.3.6 Sledeći program:
q_0 0 0 q_0
q_0 1 1 q_0
nikada ne staje, pa izračunava jedino funkciju čiji je domen prazan skup. ■

Analogno definiciji 2.3.5 moguće je definisati k-arne aritmetičke Tjuring-izračunljive funkcije. Jedina razlika je u tome što početna standardna konfiguracija mašine odgovara traci na kojoj je prikazano k argumenata funkcije.

Sa $P(x_1, x_2, \ldots, x_k) \downarrow y$ označavamo da program P polazeći od standardne konfiguracije u kojoj traka sadrži unarne reprezentacije prirodnih brojeva x_1, x_2, $\ldots$, x_k završava rad pri čemu se mašina nalazi u standardnoj konfiguraciji u kojoj traka sadrži unarnu reprezentaciju prirodnog broja y. Oznaka $P(x_1, x_2, \ldots, x_k) \downarrow$ znači da je za neko y ispunjeno $P(x_1, x_2, \ldots, x_k) \downarrow y$. Oznaka $P(x_1, x_2, \ldots, x_k) \uparrow$ znači da nije $P(x_1, x_2, \ldots, x_k) \downarrow$.

Definicija 2.3.7 Program P *konvergira* za ulaz x_1, x_2, $\ldots$, x_k ako je ispunjeno $P(x_1, x_2, \ldots, x_k) \downarrow$. Program P *divergira* za ulaz x_1, x_2, $\ldots$, x_k ako je ispunjeno $P(x_1, x_2, \ldots, x_k) \uparrow$.

Sledi nekoliko primera programa i Tjuring-izračunljivih funkcija o kojima će biti reči u kasnijim odeljcima.

Primer 2.3.8 Sledeći program izračunava funkciju $f(x) = 0$.

$q_0\ 1\ 0\ q_1$
$q_0\ 0\ 1\ q_z$
$q_1\ 0\ R\ q_0$

Sadržaj trake se briše, pri čemu se glava pomera na desno. Kada se naiđe na prvi znak 0, upisuje se znak 1 i završava rad. Dakle, $P(x) \downarrow 0$. ■

Primer 2.3.9 Sledeći program izračunava funkciju naslednika prirodnog broja u nizu prirodnih brojeva, $f(x) = x'$.

$q_0\ 1\ L\ q_0$
$q_0\ 0\ 1\ q_z$

U programu se glava najpre pomera na levo, nakon čega se nalazi iznad ćelije koja sadrži znak 0. U tu ćeliju se upisuje znak 1 i prelazi u završno stanje. Dakle, $P(x) \downarrow x'$. ■

Primer 2.3.10 Sledeći program za fiksirane k i i ($k \geq i \geq 1$) izračunava funkciju koja se naziva i-ta projekcija, $f(x_1, \ldots, x_k) = x_i$.

$q_0\ 1\ 0\ q_1$	briše zapisa broja x_1
$q_0\ 0\ R\ q_2$	
$q_1\ 0\ R\ q_0$	
$\ldots$	
$q_j\ 1\ 0\ q_{j+1}$	briše zapisa broja x_{i-1}
$q_j\ 0\ R\ q_{j+2}$	
$q_{j+1}\ 0\ R\ q_j$	
$q_{j+2}\ 1\ R\ q_{j+2}$	prelazi zapis broja x_i
$q_{j+2}\ 0\ R\ q_{j+3}$	
$q_{j+3}\ 1\ 0\ q_{j+4}$	briše zapisa broja x_{i+1}
$q_{j+3}\ 0\ R\ q_{j+5}$	
$q_{j+4}\ 0\ R\ q_{j+3}$	
$\ldots$	
$q_l\ 1\ 0\ q_{l+1}$	briše zapisa broja x_k
$q_l\ 0\ L\ q_s$	
$q_{l+1}\ 0\ R\ q_l$	
$q_s\ 0\ L\ q_s$	vraća se na početak zapisa broja x_i
$q_s\ 1\ L\ q_{s+1}$	
$q_{s+1}\ 1\ L\ q_{s+1}$	
$q_{s+1}\ 0\ R\ q_z$	

Na početku izvršavanja unarne reprezentacije brojeva x_1, ..., x_k su na traci razdvojene jednim blanko znakom. Glava se najpre pomera do kraja zapisa broja x_{i-1} i pri tom briše sve jedinice, zatim prelazi preko zapisa broja x_i i ponovo briše zapise brojeva x_{i+1}, ..., x_k. Konačno, mašina se vraća na početak zapisa broja x_i i staje. U programu je dato rešenje u kojem se podrazumeva da je $k > i > 1$, ali se on jednostavno prerađuje za preostale slučajeve. ■

2.3.5 Tjuring-neizračunljive funkcije

Definicijom 2.3.5 povezana je jedna klasa funkcija nazvanih Tjuring-izračunljivim sa programima za Tjuringovu mašinu. Kako je svaki program konačan niz naredbi, a svaka naredba konačan niz simbola iz nekog prebrojivog skupa, to postoji samo prebrojivo mnogo programa. Kako svih aritmetičkih funkcija ima neprebrojivo mnogo, to znači da postoje funkcije koje nisu Tjuring-izračunljive. Prethodno obrazloženje je u stvari dokaz sledećeg tvrđenja:

Teorema 2.3.11 Tjuring-izračunljivih funkcija ima prebrojivo mnogo. Postoje funkcije koje nisu Tjuring-izračunljive.

2.3.6 Formalni opis Tjuringove mašine

U ovom odeljku ćemo strožije definisati već uvedene pojmove.

Definicija 2.3.12 *Tjuringova mašina* je uređena petorka $\langle S, q_0, q_z, A, \delta\rangle$ gde su:

- S konačan skup stanja,
- q_0 je početno stanje, $q_0 \in S$,
- q_z je završno stanje, $q_z \in S$,
- $A = \{1, 0\}$ alfabet,
- $\delta : (S \setminus \{q_z\}) \times A \to (A \cup \{L, R\}) \times S$.

Na dalje ćemo pretpostaviti da su ćelije trake numerisane celim brojevima, tako da je ćelija nad kojom je glava pre početka izvršavanja programa označena brojem 0, ćelije levo od nje redom sa -1, -2, -3, ..., a desno od nje sa 1, 2, 3, ... *Opis trake* shvatamo kao preslikavanje $F : \mathbb{Z} \to A$, sa idejom da je $F(z) = 1$, ako je u ćeliji označenoj brojem z upisan znak 1, inače ako ćelija z sadrži blanko znak, $F(z) = 0$.

Definicija 2.3.13 *Konfiguracija* Tjuringove mašine $M = \langle S, q_0, q_z, A, \delta\rangle$ je svaka trojka $\langle F, q, e\rangle$ gde su:

- F opis trake,
- $q \in S$ stanje i
- $e \in \mathbb{Z}$.

Ideja u definiciji konfiguracije je da je q tekuće stanje, a e broj ćelije nad kojom se nalazi glava. Izvršavanje programa se odvija u koracima u kojima se menjaju konfiguracije.

Definicija 2.3.14 *Naredba* Tjuringove mašine $M = \langle S, q_0, q_z, A, \delta\rangle$ je svaka četvorka $\langle q_i, s, o, q_j\rangle$ gde je $\delta(q_i, s) = (o, q_j)$. Preslikavanje δ se naziv *program.*

Primetimo da iz uslova da je δ preslikavanje sledi da ne postoje dve naredbe koje se poklapaju po vrednostima parametara q_i i s, a koje se razlikuju bar po jednoj koordinati para (o, q_j).

Definicija 2.3.15 *Računski korak* Tjuringove mašine $M = \langle S, q_0, q_z, A, \delta \rangle$ za naredbu $I = \langle q_i, s, o, q_j \rangle$ je svaki par konfiguracija $((F', q', e'), (F'', q'', e''))$ za koje je ispunjeno:

- $q_i = q'$,
- $F'(e') = s$,
- $q_j = q''$,
- ako je operacija $o = 1$, onda važi:
  - $e' = e''$ i
  - F' i F'' se poklapaju, sem što je $F''(e'') = 1$,
- ako je operacija $o = 0$, onda važi:
  - $e' = e''$ i
  - F' i F'' se poklapaju, sem što je $F''(e'') = 0$,
- ako je operacija $o = L$, onda važi:
  - $e' - 1 = e''$ i
  - F' i F'' se poklapaju,
- ako je operacija $o = R$, onda važi:
  - $e' + 1 = e''$ i
  - F' i F'' se poklapaju,

Navedeni uslovi označavaju da je stanje u prvoj konfiguraciji q_i, da se glava u prvoj konfiguraciji nalazi iznad ćelije koja sadrži znak s, zatim da je stanje u drugoj konfiguraciji q_j i opisuju kako se menja opis trake i pozicija glave nakon upisivanja znaka 1, odnosno 0, tj. nakon pomeranja glave ulevo, odnosno udesno. Sa $(F', q', e') \vdash_I (F'', q'', e'')$ ćemo označavati da je $((F', q', e'), (F'', q'', e''))$ računski korak naredbe I.

Definicija 2.3.16 *Izračunavanje* za Tjuringovu mašinu $M = \langle S, q_0, q_z, A, \delta \rangle$ i program P opisan funkcijom δ je niz konfiguracija (F_0, q^0, e_0), (F_1, q^1, e_1), $\ldots$, (F_m, q^m, e_m) sa osobinom da važi:

- za $(F_0, q^0, e_0) \vdash_{I_1} (F_1, q^1, e_1)$, za neku naredbu I_1 programa P i stanje $q^0 = q_0$,
- za svaki prirodan broj k, $0 < k < m - 1$ je $(F_{k-1}, q^{k-1}, e_{k-1}) \vdash_{I_k} (F_k, q^k, e_k)$, neku naredbu I_k programa P,

- $(F_{m-1}, q^{m-1}, e_{m-1}) \vdash_{I_m} (F_m, q^m, e_m)$, za neku naredbu I_m programa P i stanje $q^m = q_z$,
- za svaki $k \in \{0, m-1\}$, stanje $q^k \neq q_z$,
- $e_0 = 0$ i za svaki $i < 0$, $F_0(i) = 0$,
- postoje $n > 0$ i $k_1, \ldots, k_n$ takvi da je $e_0 \leq k_1 < k_2 < \ldots < k_n$ i za svaki i za koji je $e_0 \leq i \leq k_1$, $F_0(i) = 1$, $F_0(k_1 + 1) = 0$, za svaki i za koji je $k_1 + 1 < i \leq k_2$, $F_0(i) = 1$, $F_0(k_2 + 1) = 0$, $\ldots$, za svaki i za koji je $k_{n-1} + 1 < i \leq k_n$, $F_0(i) = 1$ i za svaki i za koji je $k_n < i$, $F_0(i) = 0$,
- $F_m(e_m) = 1$ i za svaki $i < e_m$, $F_m(i) = 0$ i
- postoji $k \geq e_m$ tako da je za svaki i za koji je $e_m \leq i \leq k$, $F_m(i) = 1$ i za svaki $j > k$, $F_m(j) = 0$.

2.3.7 Kombinovanje Tjuringovih mašina

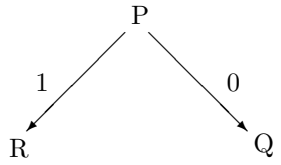
Dosadašnji primeri programa za Tjuringovu mašinu su bili uglavnom jednostavni. U ovom odeljku ćemo opisati kako se kombinovanjem jednostavnih programa mogu graditi složeniji programi, što podseća na ideju potprograma kod standardnih programskih jezika.

Program iz primera 2.3.9 je izračunavao funkciju naslednika $f(x) = x'$, što je zapravo $f(x) = x + 1$. Pretpostavimo da želimo napisati program koji izračunava funkciju $g(x) = x + 2$. Tada možemo postupiti na jedan od dva načina: napisati novi program, sličan programu iz primera 2.3.2, ili uočiti da je $g(x) = f(f(x))$, tj. da se funkcija g dobija kompozicijom funkcije f sa samom sobom, te iskoristiti program iz primera 2.3.9 za kreiranje novog programa.

Neka je P neki program za Tjuringovu mašinu. Sa $S(P)$ označimo skup stanja programa P. Dva programa P_1 i P_2 imaju disjunktne skupove stanja ako je $S(P_1) \cap S(P_2) = \{q_0, q_z\}$.

Posmatrajmo sada dve kopije programa iz primera 2.3.9 sa tako imenovanim stanjima da imaju disjunktne skupove stanja. Neka je q^1 takvo stanje da se ne nalazi ni u $S(P_1)$ ni u $S(P_2)$. Zatim zamenimo sve pojave završnog stanja u prvoj kopiji programa i sve pojave početnog stanja u drugoj kopiji programa stanjem q^1 i nadovežimo drugu kopiju programa na prvu. Ovako dobijeni program je *kompozicija* dva razmatrana programa.

Transformacija slična kompoziciji se može napraviti i u nešto složenijim okolnostima. Najpre ćemo oslabiti zahtev iz definicija 2.3.14 i 2.3.16 programa za Tjuringovu mašinu: u standardnoj konfiguraciji glava mašine ne mora da se pozicionira nad kranjom levom ćelijom trake koja sadrži znak 1 ni na početku, ni na kraju izvršavanja mašine. Neka su P, Q i R takvi programi. Uz to pretpostavimo i da su im skupovi stanja medjusobno disjunktni i označimo završno stanje programa P sa q_z^P, a početna stanja programa Q i R sa q_0^Q i q_0^R. Posmatrajmo sada sledeći program:



Slika 2.2. Grananje pri komponovanju Tjuringovih mašina.

P	naredbe programa P
q_z^P 0 0 q_0^Q	prelazi se na program Q
q_z^P 1 1 q_0^R	prelazi se na program R
Q	naredbe programa Q
R	naredbe programa R

šematski prikazan na slici 2.2. Nakon izvršavanja programa P, ako se glava nalazi nad ćelijom koja sadrži znak 0, nastavlja se sa izvršavanjem programa Q, a ako je sadržaj ćelije znak 1, nastavlja se sa izvršavanjem programa R. Očigledno je da ova konstrukcija podseća na grananje u programima pisanim na nekom standardnom programskom jeziku.

Sledeći jednostavni programi za Tjuringovu mašinu mogu poslužiti kao osnovne gradivne jedinice složenijih programa koji se dobijaju kompozicijom i grananjem:

- program koji u tekuću ćeliju upisuje znak 1 i potom staje,
- program koji u tekuću ćeliju upisuje znak 0 i potom staje,
- program koji pomera glavu za jedno mesto u levo i potom staje i
- program koji pomera glavu za jedno mesto u desno i potom staje.

2.3.8 Varijante Tjuringove mašine

Ovde izabran pristup u definisanju Tjuringove mašine je samo jedan od mogućih. Recimo, posmatraju se Tjuringove mašine u kojima:

- alfabet kojim se zapisuje sadržaj ćelija trake ne mora biti unarni,
- pored završnog stanja q_z uvode se i neka specijalna završna stanja, recimo q_{da} i q_{ne} koja, intuitivno, znače pozitivan, odnosno, negativan odgovor na postavljeni problem,
- dozvoljena je traka koja je beskonačna samo na jednu stranu, tj. postoji krajnja leva ćelija, dok se na desno traka pruža neograničeno,
- umesto samo jedne postoji više traka, a za svaku traku postoji posebna glava,
- nad jednom trakom postoji više glava umesto samo jedne,
- traka je dvodimenzionalna, a ne jednodimenzionalna, tj. traka podseća na beskonačnu šahovsku ploču,

- u jednoj naredbi mašine moguće je i upisati znak u ćeliju i pomerati glavu,
- ne važi zahtev za determinisanošću, tj. dozvoljeno je da postoje naredbe koje odgovaraju istom stanju i znaku u ćeliji nad kojom se nalazi glava, a koje se razlikuju po dejstvu (operaciji koja se izvršava i/ili stanju u koje se prelazi) itd.

Zanimljivo je da u smislu izračunljivosti gotovo sve od ovih varijanti Tjuringove mašine odgovaraju istoj klasi funkcija, tj. klasi Tjuring-izračunljivih funkcija, kao i osnovna verzija mašine. Izuzetak predstavljaju neki slabiji, restriktivni slučajevi: recimo, mašina čija traka je ograničena sa jedne strane i koristi unarni alfabet ili mašina koja ima samo dva stanja i koristi alfabet od dva znaka. Ekvivalencija varijanti Tjuringove mašine se dokazuje tako što se pokaže da za svaki program P za neku od varijanti Tjuringove mašine postoje programi za preostale varijante koji simuliraju izvršenje programa P i izračunavaju istu funkciju. Skiciraćemo neke od ovih postupaka.

Izbor varijante Tjuringove mašine zavisi od primene kojom se bavimo. Recimo, u analizi složenosti algoritama u glavi 10 se koristi više varijanti mašina zavisno od klase složenosti koja se proučava.

Tjuringova mašina sa bogatijim alfabetom

Kao što je napomenuto u odeljku 2.3.2 reči bilo kog prebrojivog alfabeta se mogu prikazati pomoću unarnog alfabeta, tako da se u slučaju Tjuringove mašine sa trakom koja nije ograničena ni sa jedne strane i u većini drugih slučajeva alfabet može, zavisno od potrebe, slobodno određivati. Na primer, u glavi 10, kada bude analizirana složenost algoritama, prirodni brojevi će biti dati u binarnoj, a ne kao do sada u unarnoj, reprezentaciji. Ostvarena ušteda će biti značajna pošto je unarna reprezentacija prirodnih brojeva eksponencijalno duža od binarne.

Tjuringova mašina sa trakom koja ima početak sa leve strane

Alfabet kod mašina ove vrste sadrži još jedan specijalni znak, recimo $\triangleright$, koji služi da se prepozna početna ćelija sa leve strane. Preko tog simbola se nikada ne prepisuje ni jedan drugi simbol, niti se glava sme pomeriti levo, tako da je jedina moguća naredba kada je znak $\triangleright$ u ćeliji ispod glave oblika:

$q_i \triangleright R\ q_j$

za neka stanja q_i i q_j.

Očigledno je da se sve funkcije koje se izračunavaju pomoću Tjuringove mašine čija traka ima najlevlju ćeliju mogu izračunati i pomoću standardne varijante mašine: jednostavno se neće koristiti deo trake koji se nalazi sa leve strane ćelije iznad koje je pozicionirana glava pre početka izvršavanja.

Posmatrajmo Tjuringovu mašinu M_1 koja na traci neograničenoj u oba smera izračunava neku funkciju f. Konstruišimo mašinu M_2 čija traka je ograničena sa leve strane. Najpre, označimo ćelije trake mašine M_1 celim brojevima kao u definiciji opisa trake. Tada je sa 0 označena ćelija nad kojom je glava, ćelije na levo od nje sa -1, -2, ..., a desno od nje sa 1, 2, ... Zamislićemo da je traka presavijena tako da formira dva reda, kao na slici 2.3, pri čemu su u gornjem redu ćelije numerisane sa 0, 1, 2, ..., a u donjem sa -1, -2, ... i da je dodata još jedna

$$\triangleright \left\{ \begin{array}{ccc} 0 & 1 & 2 \\ -1 & -2 & -3 \end{array} \right.$$

Slika 2.3. Jednostrana traka.

posebna ćelija koja predstavlja levi kraj tako dobijene trake. Ćelije trake mašine M_2 sadržaće simbole koji bi se nalazili u odgovarajućim ćelijama trake mašine M_1.

Alfabet koji će koristiti mašina M_2 će biti $A_2 = \{(0,0),(1,0),(0,1),(1,1),\triangleright\}$, dakle sadržaće parove znakova 1 i 0 i simbol $\triangleright$. Znak $\triangleright$ će se nalaziti samo u najlevljoj ćeliji. Parovi znakova će opisivati sadržaj odgovarajućih parova ćelija trake mašine M_1. Stanja ove mašine će biti sledećeg oblika: za svako stanje q mašine M_1 postojaće stanja $\langle q,1\rangle$ i $\langle q,2\rangle$ koja će redom značiti 'mašina M_2 u stanju q radi nad ćelijama iz gornjeg reda' i 'mašina M_2 u stanju q radi nad ćelijama iz donjeg reda'. Stanja $\langle q_z,1\rangle$ i $\langle q_z,2\rangle$ će oba biti završna stanja.

Naredbe programa za mašinu M_1 se transformišu na sledeći način:

- Ako je naredba oblika $q_i\ s\ o\ q_j$ posmatraju se dve naredbe koje simuliraju izvršavanje na levoj, odnosno desnoj strani trake mašine M_1:
  - $\langle q_i,1\rangle\ (s,x)\ \overline{o}\ \langle q_j,1\rangle$, gde je

$$\overline{o} = \left\{ \begin{array}{ll} L & \text{za } o = L \\ R & \text{za } o = R \\ (1,x) & \text{za } o = 1 \\ (0,x) & \text{za } o = 0 \end{array} \right.$$

    i $x \in \{0,1\}$. Ovom naredbom se menjaju položaj glave, odnosno sadržaj ćelija u gornjem redu koji odgovara desnoj strani trake mašine M_1,
  - $\langle q_i,2\rangle\ (x,s)\ \overline{o}\ \langle q_j,2\rangle$, gde je

$$\overline{o} = \left\{ \begin{array}{ll} L & \text{za } o = R \\ R & \text{za } o = L \\ (x,1) & \text{za } o = 1 \\ (x,0) & \text{za } o = 0 \end{array} \right.$$

    i $x \in \{0,1\}$. Ovom naredbom se menjaju položaj glave, odnosno sadržaj ćelija u donjem redu koji odgovara levoj strani trake mašine M_1. Zbog numeracije ćelija u traci sa dva reda, kretanje glave mašine M_1 u levo nad ćelijama sa negativnim brojevima, ovde se prikazuje kao kretanje glave u desno.

- Za svako stanje q mašine M_1 dodaju se naredbe:
  - $\langle q,1\rangle \triangleright R\ \langle q,2\rangle$ i
  - $\langle q,2\rangle \triangleright R\ \langle q,1\rangle$

  koje se izvršavaju kada se glava mašine M_2 nađe nad krajnjom levom ćelijom u koju je upisan posebni simbol $\triangleright$. Na ovu ćeliju se može doći pri pomeranju

ulevo glave koja obrađuje gornji red trake mašine M_2, nakon čega treba nastaviti kretanje obrađujući donji red trake (tada se kretanje u levo kod mašine M_1 prikazuje kao kretanje u desno mašine M_2). Prelazak sa obrađivanja gornjeg na obrađivanje donjeg reda je opisan promenom druge kordinate u stanju mašine. Slično se dešava i kada se obrađujući donji red trake glava pomera ulevo, što bi odgovaralo pomeranju udesno glave mašine M_1.

Pretpostavimo da mašina M_1 za neki ulaz x izračunava rezultat y. Tada, opis mašine M_2 garantuje analogno ponašanje, pri čemu će sadržaj ćelija 0, 1, 2, ... mašine M_1 biti predstavljen prvim koordinatama znaka u ćelijama trake mašine M_2 koje se nalaze neposredno do ćelije trake koja sadrži znak $\triangleright$.

Tjuringova mašina sa više traka

Pretpostavićemo da je ova varijanta mašina organizovana na sledeći način. Za neki $k > 0$ Tjuringova mašina sa k traka sastoji se od traka označenih sa 1, 2, 3, ..., k. Sve trake imaju kraj sa leve strane, što nije ograničenje, kao što je pokazano u odeljku 2.3.8. Nad svakom trakom nalazi se posebna glava. U svakom koraku čita se tekuća ćelija na svakoj od traka i preduzima odgovarajuća akcija, tj. neke glave upisuju znak, neke se pomeraju levo, a neke desno. Na početku izvršavanja ulazni podaci se smeštaju na prvu traku, dok su sve ostale trake prazne. Ako izvršavanje mašine shvatimo kao izračunavanje neke funkcije, rezultat se smešta u poslednju, k-tu traku.

U nekim situacijama mašina sa više glava olakšava programiranje. Takav slučaj je, recimo, sa ispitivanjem da li je neka reč palindrom[12]. Najpre se reč iz prve trake iskopira na drugu, glava prve trake se vrati na levo, dok glava druge ostaje u krajnjoj desnoj poziciji. Konačno, dve glave se kreću u suprotnim smerovima i ispituju da li se nalaze nad jednakim znacima.

Pošto Tjuringova mašina sa jednom trakom trivijalno spada u klasu mašina sa više traka, svaka funkcija izračunljiva mašinom prve vrste je automatski izračunljiva i u drugom slučaju. Obrnuto, pretpostavimo da je M_1 Tjuringova mašina sa k traka koja izračunava neku funkciju f. Konstruisaćemo mašinu M_2 sa jednom trakom koja će simulirati izvršavanje mašine M_1. Sadržaji k traka mašine M_1 biće redom smešteni u jedinoj traci mašine M_2. Uobičajeni unarni alfabet će biti proširen na sledeći način:

- novi znak $\triangleright$ će označavati početak trake sa leve strane, tako da se glava trake ne sme kretati levo od njega,
- novi znak $\triangleright'$ će označavati početak sadržaja svake od k traka mašine M_1 i preko njega će glava mašine M_2 smeti da prelazi na levo,
- novi znak $\triangleleft$ će služiti kao oznaka desnog kraja trenutnog zauzeća svake od predstavljenih traka, a dva uzastopna znaka $\triangleleft$ će označavati kraj sadržaja svih k traka mašine M_1 zapisanih na jedinoj traci mašine M_2,
- novi znak $\triangleleft'$ će služiti kao privremena varijanta znaka $\triangleleft$ prilikom simuliranja nekih koraka izvršavanja mašine M_1 i

[12]Reč je palindrom ako se čitanjem sa leva na desno i sa desna na levo dobija isti niz znaka.

- novi znaci $\underline{1}$, $\underline{0}$ će označavati da se glava neke od traka nalazi upravo iznad znaka 1, odnosno 0 u toj ćeliji

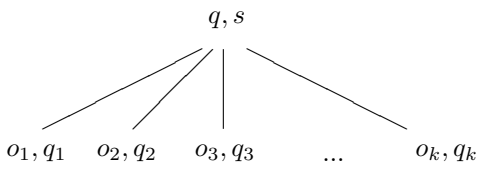
Recimo, polazni sadržaj trake mašine M_2 bi izgledao kao:

$$\triangleright \; \triangleright' \, w \; \triangleleft \underbrace{\triangleright' \triangleleft \ldots \triangleright' \triangleleft}_{k-1} \triangleleft$$

gde w predstavlja ulazne podatke zapisane na prvoj traci mašine M_1 i prvi znak iz w je podvučena verzija znaka jer se nad njim nalazi glava.

Mašina M_2 simulira jedan korak izvršavanja mašine M_1 tako što dva puta prelazi sadržaj svoje trake sa leva na desno i nazad. U prvom prolazu prikuplja informacije o k znaka koji se trenutno nalaze ispod glava mašine M_1 - to su podvučeni znaci na traci mašine M_2. Da bi se omogućilo pamćenje potrebno je uvesti nova stanja koja odgovaraju kombinacijama stanja mašine M_1 i k-torki simbola. Na primer, ako se mašina M_1 nalazi u stanju q, a simboli u ćelijama iznad kojih su glave traka su redom $o_1, \ldots, o_k$, odgovarajuće stanje mašine M_2 možemo označiti sa $q(o_1, \ldots, o_k)$. Na osnovu svog stanja nakon prvog prelaska, mašina M_2 u drugom prelasku izvršava akciju koja simulira razmatrani korak rada mašine M_1. To se izvodi tako što se prelazi preko ćelija trake mašine M_2 i eventualno menja sadržaj ćelija u koje su upisani podvučeni znaci. Promena se ogleda ili u prepisivanju tekućeg podvučenog znaka novim podvučenim znakom (ako je operacija pisanje) ili zamenom povučene verzije znaka nepodvučenom i promenom jednog od susednih znaka u podvučenu verziju (ako je akcija pomeranje glave). Jedini problem nastaje kada je trenutna pozicija upravo desni kraj zapisa neke od traka mašine M_1, a glava treba da se pomakne na desno. To znači da se u ćeliji iznad koje je glava nalazi znak $\triangleleft$. Kod mašine M_1 to ne bi predstavljalo problem, ali ovde desno od znaka $\triangleleft$ nema slobodnog prostora jer se tu nalaze zapisi sadržaja ostalih traka mašine M_1. Zato se najpre znak $\triangleleft$ prepisuje znakom $\triangleleft'$, pa se pomere svi znaci počev od njega za po jedno mesto u desno, zatim se glava vraća na levo do prvobitne pozicije znaka $\triangleleft'$ i prepisuje ga blanko znakom. Konačno, zbog kopiranja sadržaja u prvoj desnoj ćeliji je još jedan znak $\triangleleft'$ koji se prepisuje znakom $\triangleleft$. Simulacija se nastavlja dok se mašina M_2 ne zaustavi, nakon čega briše sve delove trake koji prikazuju sadržaje traka mašine M_1, sem za k-tu traku koja predstavlja rezultat.

Pretpostavimo da $|x|$ označava dužinu ulaznih podataka mašine M_1, a f funkciju za koju važi da je broj koraka izračunavanja mašine M_1 za ulaz x najviše $f(|x|)$ i primetimo da ni na jednoj od k traka mašine M_1 ne može biti zauzeto više od $f(|x|)$ ćelija. Prema tome, svakom koraku izvršavanja mašine M_1 odgovara najviše $c_1 \cdot kf(|x|)$ koraka rada mašine M_2 potrebnih za dvostruki prelazak trake i $c_2 \cdot kf(|x|)$ koraka rada mašine M_2 potrebnih za eventualno pomeranje sadržaja u desno, za neke konstante c_1 i c_2. Mašina M_2 treba da na početku izvršavanja postavi polazni sadržaj trake pri čemu pomera ulazne podatke i dopisuje znake $\triangleright'$ i $\triangleleft$, kao i da na kraju obriše deo trake zauzet prikazom prve $k-1$ trake mašine M_1, za šta joj je potrebno $c_3 \cdot f(|x|)$ koraka, za neku konstantu c_3. Na osnovu ove analize je očigledno da je ukupna dužina izvršavanja mašine M_2 ograničena odozgo sa $c_4 \cdot (f(|x|))^2$ za neku konstantu c_4, tako da simulacija radi samo polinomijalno duže od mašine M_1.



Slika 2.4. Nedeterministički korak u izvršavanju Tjuringove mašine.

Nedeterministička Tjuringova mašina

U komentaru u odeljku 2.3.3, ponašanje do sada korištenih verzija Tjuringove mašine je okarakterisano kao determinističko, tj. za svaku kombinaciju tekućeg stanja i znaka bila je predviđena samo jedna akcija. Kod *nedeterminističke Tjuringove mašine* ovaj zahtev ne postoji. U odnosu na definiciju 2.3.12 Tjuringove mašine, jedina razlika je u sledećem:

- $\delta : (S \setminus \{q_z\}) \times A \to \mathbb{P}((A \cup \{L, R\}) \times S)$,

gde je $\mathbb{P}()$ oznaka partitivnog skupa, odnosno za stanje $q \in S \setminus \{q_z\}$ i znak $s \in A$, $\delta(q, s)$ je proizvoljan konačan podskup skupa $((A \cup \{L, R\}) \times S)$. Drugim rečima, za tekuću konfiguraciju mašine može postojati više različitih konfiguracija do kojih se dolazi izvršavanjem neke od naredbi programa.

U izvršavanju nedeterminističke Tjuringove mašine postoji svojevrsna mogućnost izbora: u slučaju da za neko stanje q neki znak s postoji više mogućih naredbi treba izabrati neku od njih i nastaviti izvršavanje, što je šematski prikazano kao deo jednog drveta na slici 2.4. Grananje u drvetu je konačno, što znači da u svakom koraku izvršavanja postoji samo konačno mnogo opcija za izbor, dok grane predstavljaju moguće redoslede izvršavanja programa.

Deterministička Tjuringovim mašinama definisana je jedna klasa izračunljivih funkcija. Kod nedeterminističkih mašina situacija je donekle izmenjena. One su pre svega pogodne za davanje odgovora 'da' ili 'ne' na pitanja oblika 'da li za ulazne podatke važi ...?' Imajući u vidu ideju o uvođenju novih stanja q_{da} i q_{ne} zaustavljanje u nekom od ovih stanja ima značenje pozitivnog, odnosno negativnog, odgovora. Snaga, odnosno na jeziku savremenih računara - brzina, nedeterminističkih Tjuringovih mašina je posledica sledeće asimetrične konvencije: mašina potvrdno odgovara na pitanje ako se bar jedno od mogućih izračunavanja završava u stanju q_{da}, dok jedino okončanje svih mogućih izračunavanja u stanju q_{ne} znači da je odgovor 'ne'. Na osnovu ovog dogovora, nedeterministička mašina se može zamisliti kao višeprocesorski sistem koji se ponaša na sledeći način. U svakom koraku svaki od procesora kreira onoliko novih procesora koliko ima različitih konfiguracija u koje taj procesor može preći izvršavanjem tekuće naredbe. Ako mu u nastavku izvršavanja bilo koji od njegovih potomaka vrati informaciju o potvrdnom odgovoru, procesor tu informaciju prosleđuje svom neposrednom pretku. Negativan odgovor se prosleđuje samo ako je dobijen od svih neposrednih potomaka. Zapravo, svaki procesor izračunava disjunkciju odgovora svojih potomaka. Primetimo da, ako ni jedno izračunavanje ne dovodi do stanja q_{da} i bar jedno izračunavanje ne dovodi ni do kog završnog stanja, nedeterministička mašina divergira.

Ovakav model mašine je pogodan za rešavanje nekih složenih problema, o čemu će više reči biti u glavi 10. Na primer, pretpostavimo da želimo ispitati da li je neki prirodan broj n složen ili prost. Običnom Tjuringovom mašinom problem bi se mogao rešiti na sledeći način: delili bismo broj svim prirodnim brojevima između 2 i $\frac{n}{2}$ i na osnovu toga dali odgovor. U slučaju nedeterminističke Tjuringove mašine na jednom mestu bismo imali mogućnost izbora broja kojim delimo broj n, pa ako je n složen, a izabrani broj delilac, mogli bismo dati odgovor u jednom koraku, što bi bio značajan dobitak u odnosu na deterministički postupak. Lako je uočiti da ovaj postupak nije realan, u smislu da izbor delioca podrazumeva da mi već znamo da je n složen, tj. da nam je poznat bar jedan njegov činilac. Međutim, i pored toga, nedeterministička Tjuringova mašina se može simulirati determinističkom mašinom, tako da se izražajnost u smislu onoga šta mašina može odgovoriti ne menja.

Pretpostavimo da je M_1 nedeterministička Tjuringova mašina. Odgovarajuća deterministička Tjuringova mašina M_2 će sistematski prelaziti sve moguće redoslede izvršavanja mašine M_1, najpre dužine 1, pa dužine 2 itd[13]. Ovo obezbeđuje da ni jedno moguće konačno izvršavanje neće biti preskočeno. Zato, ako bi se mašina M_1 u nekom trenutku izvršavanja našla u stanju q_{da}, to isto će pre ili posle biti slučaj i sa mašinom M_2. Ako svi mogući redosledi izvršavanja mašine M_1 dovode do stanja q_{ne}, i mašina M_2 će se naći u tom stanju kada iscrpi sve mogućnosti. Konačno ako mašina M_1 divergira za date ulazne podatke x i mašina M_2 se neće zaustaviti.

Deterministička mašina M_2 će imati tri trake: prva traka uvek sadrži ulazni podatak i nikada se ne menja, na drugoj traci će se simulirati izvršavanje mašine M_1, a na trećoj će se, kao na nekom steku, pamtiti niz brojeva koji predstavljaju prikaz izabranih pravaca u svim mogućim trenucima izbora naredbe koja se izvršava. Na početku izvršavanja mašine M_2 ulazni podatak se nalazi na prvoj traci, dok su preostale dve trake prazne. U osnovnom ciklusu rada mašina kopira sadržaj prve na drugu traku i koristeći sadržaj treće trake, kao uputstvo za redosled koraka, simulira jedan deterministički redosled izvršavanja mašine M_1.

Očigledno je da deterministička mašina M_2 u najgorem slučaju bar jednom posećuje svaki čvor drveta koje prikazuje izvršavanje nedeterminističke mašine M_1. Ovih čvorova može biti eksponencijalno više nego što je dužina najkraćeg mogućeg izračunavanja mašine M_1 koje dovodi do stanja q_{da}, ako takvo uopšte postoji. Zato je izvršavanje determinističke mašine M_2 u najgorem slučaju eksponencijalno duže nego izvršavanje nedeterminističke mašine M_1. Za sada nije poznato da li je simulaciju moguće izvesti uz samo polinomijalni gubitak vremena. U vezi sa tim je čuveni problem da li je $P = NP$ o čemu će biti reči u glavi 10.

U istoj glavi biće korištena i jedna specijalna vrsta nedeterminističkih Tjuringovih mašina, koje se zovu *precizne*, a u kojima u svakom koraku izvršavanja koji nije poslednji korak postoje tačno dva moguća nastavka rada. Ovakvoj mašini odgovara drvo stepena granjanja dva. Svaka nedeterministička mašina se može prevesti u mašinu ovog tipa tako što se svaka situacija sa višestrukom mogućnošću grananje prevodi u niz naredbi u kojima postoje samo po dve mogućnosti, dok se u situacijama u kojima nema izbora veštački doda naredba koja ne menja ni poziciju glave, ni sadržaj trake.

[13]Ovo podseća na mehanizam pretrage po širini.

2.3.9 Univerzalna Tjuringova mašina

Univerzalna Tjuringova mašina, u oznaci UTM, je svojevrstan primer programibilnog digitalnog računara opšte namene sa programom i podacima smeštenim u memoriju koji simulira izvršavanje ostalih Tjuringovih mašina. Ulazni podaci koji se smeštaju na traku univerzalne Tjuringove mašinu su opis neke posebne mašine, tj. njen program, i ulazni podaci te mašine, a rezultat izvršavanja je rezultat rada simulirane posebne mašine.

Postojanje univerzalne Tjuringove mašine se može dokazati direktno, davanjem njenog opisa, što ovde nećemo raditi. Umesto toga, pozvaćemo se na teoreme 2.5.1 i 2.5.2 iz odeljka 2.5 kojima se uspostavlja jednakost između klasa Tjuring-izračunljivih funkcija i parcijalno rekurzivnih funkcija. Postojanje univerzalne funkcije opisane u odeljku 2.4.12 kao posledicu ima postojanje univerzalne Tjuringove mašine.

2.4 Rekurzivne funkcije

U ovom odeljku ćemo analizirati drugi pristup definiciji izračunljivosti koji je naizgled potpuno različit od Tjuringovih mašina. Međutim, kao što ćemo videti, oba pristupa određuju jednu te istu klasu funkcija. Proučavanje ćemo započeti takozvanim primitivno rekurzivnim funkcijama. Iako sadrži sve uobičajene funkcije sa kojima se redovno srećemo, u ovoj klasi ipak nisu sve funkcije za koje intuitivno smatramo da su izračunljive. Zbog toga se klasa širi što dovodi do pune klase parcijalno rekurzivnih funkcija.

2.4.1 Primitivno rekurzivne funkcije

Prilikom definisanja aritmetičkih funkcija često se javlja induktivni postupak kojim se uvodi, recimo, funkcija faktorijel:

$$0! =_{def} 1$$

$$(n+1)! =_{def} (n!) \cdot (n+1).$$

Ovde je funkcija faktorijel induktivno definisana pomoću funkcije množenja. U opštem slučaju induktivni postupak definisanja funkcije f na osnovu funkcije g je oblika $f(0) = m$, $f(n+1) = g(f(n), n)$. Pod pretpostavkom da je funkcija g definisana, lako se vidi da je i funkcija f dobro definisana: $f(0) = m$, $f(1) = g(f(0), 0) = g(m, 0)$, $f(2) = g(f(1), 1) = g(g(m, 0), 1)$, $\ldots$, $f(n+1) = g(f(n), n) = \ldots = g(g(\ldots(g(m, 0), 1), \ldots, n-1), n)$.

U definiciji klase primitivno rekurzivnih funkcija koristićemo upravo ovu ideju u definisanju funkcija: polazeći od jednog malog broja jednostavnih funkcija definisaćemo nove koristeći donekle uopšteni postupak induktivnog definisanja.

2.4.2 Definicija primitivno rekurzivnih funkcija

Definicija 2.4.1 Klasa *primitivno rekurzivnih funkcija* sadrži osnovne funkcije:

- nula funkciju $Z(n) = 0$, za svako $n \in \mathbb{N}$,

- funkciju naslednika $S(n)$ prirodnog broja n u nizu prirodnih brojeva i
- funkcije projekcije $P_k^i(x_1, \ldots, x_k) = x_i$ za sve i i k takve da je $1 \leq i \leq k$

i sve funkcije koje se od njih dobijaju konačnim brojem primena osnovnih operacija:

- *kompozicije*, tj. ako su već definisane funkcije $g : \mathbb{N}^m \to \mathbb{N}$ i $h_1, \ldots, h_m : \mathbb{N}^k \to \mathbb{N}$, definisana je i funkcija $f : \mathbb{N}^k \to \mathbb{N}$ tako da je za sve $(x_1, \ldots, x_k) \in \mathbb{N}^k$:

  $$f(x_1, \ldots, x_k) = g(h_1(x_1, \ldots, x_k), \ldots, h_m(x_1, \ldots, x_k))$$

  i

- *primitivne rekurzije*, tj. ako su već definisane funkcije $g : \mathbb{N}^m \to \mathbb{N}$ i $h : \mathbb{N}^{m+2} \to \mathbb{N}$, definisana je i funkcija $f : \mathbb{N}^{m+1} \to \mathbb{N}$ tako da je za sve $(x_1, \ldots, x_m) \in \mathbb{N}^m$:

  - $f(0, x_1, \ldots, x_m) = g(x_1, \ldots, x_m)$ i
  - $f(n+1, x_1, \ldots, x_m) = h(f(n, x_1, \ldots, x_m), n, x_1, \ldots, x_m)$ za $n \in \mathbb{N}$,.

  Pri tome je funkcija f definisana primitivnom rekurzijom nad h sa bazom g.

Iako je za funkciju naslednika ispunjeno $S(n) = n + 1$, da smo u definiciji 2.4.1 izabrali taj način uvođenja funkcije S, sabiranje bi moralo biti osnovna funkcija. Ovako, sabiranje će biti definisano. Primetimo i da se operacija primitivne rekurzije u slučaju kada je funkcija f unarna svodi na induktivnu definiciju: funkcija g je u tom slučaju konstanta, $f(0) = d$ i $f(n+1) = h(f(n), n)$.

Drugi način definisanja klase primitivno rekurzivnih funkcija je da se ona uvede kao najmanja klasa koja sadrži osnovne funkcije i zatvorena je za osnovne operacije. Ova definicija pretpostavlja postojanje beskonačne kolekcije klasa koje sadrže ovakve funkcije, pri čemu se najmanja klasa dobija kao presek svih takvih klasa. Sa druge strane definicija 2.4.1 je konstruktivni način za opisivanje klase i svojevrstan način da se izbegne beskonačnost u analizi izračunljivosti.

Primetimo da su sve osnovne funkcije totalne, a da osnovne operacije primenjene na totalne funkcije takođe daju totalne funkcije. Prema tome, primitivno rekurzivne funkcije su totalne. Iz teoreme 2.5.1 takođe sledi da su primitivno rekurzivne funkcije Tjuring-izračunvljive.

2.4.3 Primeri primitivno rekurzivnih funkcija

Da bi se za neku funkciju pokazalo da je primitivno rekurzivna potrebno je naći njen opis u smislu definicije 2.4.1, tj. pokazati da je ona inicijalna ili da se može iz inicijalnih funkcija dobiti konačnim nizom primena operacija kompozicije i/ili primitivne rekurzije.

Primer 2.4.2 Sve konstantne funkcije $K_k^j : \mathbb{N}^k \to \mathbb{N}$ za koje je $K_k^j(x_1, \ldots, x_k) = j$ za sve $(x_1, \ldots, x_k) \in \mathbb{N}^k$ su primitivno rekurzivne. Najpre, za $k = 1$ reč je o konstantnim funkcijama oblika $K_1^j : \mathbb{N} \to \{j\}$, gde je $j \in \mathbb{N}$ konstanta, koje su primitivno rekurzivne jer ih možemo definisati kao $K_1^j(x) = S^j(Z(x))$. Neka je $k \geq 1$. Tada možemo definisati funkciju K_{k+1}^j primitivnom rekurzijom nad projekcijom P_{k+2}^1 sa bazom K_k^j:

$$K^j_{k+1}(0, x_1, \ldots, x_k) = K^j_k(x_1, \ldots, x_k),$$

$$K^j_{k+1}(n+1, x_1, \ldots, x_k) = P^1_{k+2}(K^j_{k+1}(n, x_1, \ldots, x_k), n, x_1, \ldots, x_k).$$

Primetimo da je moguće definisati i $K^j_k(x_1, \ldots, x_k) = S^j(Z(P^1_k(x_1, \ldots, x_k)))$. ■

Primer 2.4.3 Funkcija sabiranja $f(n, x) = n+x$ je primitivno rekurzivna. Najpre uočimo da je funkcija $S(P^1_3(x, y, z))$ primitivno rekurzivna jer se dobija kompozicijom osnovnih funkcija naslednika $S()$ i projekcije $P^1_3()$. Funkciju sabiranja sada možemo definisati sa:

$$f(0, x) = P^1_1(x)$$

$$f(n+1, x) = S(P^1_3(f(n, x), n, x)).$$

U prvom redu definicije koristimo unarnu funkciju projekcije, a u drugom funkcije naslednika i projekcije. Prema tome, funkcija f je definisana primitivnom rekurzijom nad $S(P^1_3())$ sa bazom $P^1_1()$. ■

Primer 2.4.4 Funkcija prethodnik definisana sa

$$\mathrm{p}reth(x) = \begin{cases} 0 & \text{za } x = 0 \\ \text{prethodnik od } x \text{ u nizu prirodnih brojeva} & \text{za } x > 0 \end{cases}$$

je primitivno rekurzivna jer se dobija primenom operacije primitivne rekurzije:

$$\mathrm{p}reth(0) = 0$$

$$\mathrm{p}reth(n+1) = P^2_2(\mathrm{p}reth(n), n).$$

Primetimo da je funkcija p$reth$, kako je ovde definisana, totalna, za razliku od istoimene Tjuring-izračunljive funkcije iz zadatka 11.5 koja je parcijalana. ■

2.4.4 Primitivno rekurzivne operacije

Neku operaciju nazvaćemo primitivno rekurzivnom ako primenjena na primitivno rekurzivne funkcije daje primitivno rekurzivnu funkciju, tj. ako se može simulirati korištenjem kompozicije, primitivne rekurzije i osnovnih primitivno rekurzivnih funkcija. Primitivno rekurzivne operacije omogućiće nešto jednostavnije definisanje primitivno rekurzivnih funkcija.

Eksplicitna transformacija

Definicija 2.4.5 Neka je $g : \mathbb{N}^k \to \mathbb{N}$ k-arna funkcija i neka su $h_1, \ldots, h_k : \mathbb{N}^l \to \mathbb{N}$ l-arne funkcije od kojih je svaka ili projekcija ili konstantantna funkcija. Funkcija f definisana sa:

$$f(x_1, \ldots, x_l) = g(h_1(x_1, \ldots, x_l), \ldots, h_k(x_1, \ldots, x_l))$$

je dobijena iz funkcije g *eksplicitnom transformacijom*.

Očigledno je da je funkcija f dobijena iz funkcije g dupliranjem i/ili permutovanjem argumenata i zamenom argumenata konstantama. Eksplicitna transformacija predstavlja jednu vrstu kompozicije, pa važi:

Teorema 2.4.6 Ako je funkcija g primitivno rekurzivna, primitivno rekurzivne su i sve funkcije dobijene iz nje eksplicitnom transformacijom.

Primer 2.4.7 Neka je funkcija $g : \mathbb{N}^2 \to \mathbb{N}$ primitivno rekurzivna. Tada je i funkcija $f(x, y, z) = g(z, x)$ primitivno rekurzivna jer je dobijena eksplicitnom transformacijom iz primitivno rekurzivne funkcije g. ■

Primer 2.4.8 Funkcija *monus*[14] definisana sa

$$\dot{-}(n, x) = \begin{cases} 0 & \text{za } x \leq n \\ x - n, & \text{za } x > n \end{cases}$$

je primitivno rekurzivna jer se dobija primenom operacije primitivne rekurzije:

$$\dot{-}(0, x) = P_1^1(x)$$

$$\dot{-}(n + 1, x) = h(\dot{-}(n, x), n, x)$$

gde je funkcija h defnisana sa $h(x, y, z) = p\mathit{reth}(P_3^1(x, y, z))$, tj. kao prethodnik prve projekcije, pa je prema teoremi 2.4.6 primitivno rekurzivna jer je dobijena eksplicitnom transformacijom iz primitivno rekurzivne funkcije $p\mathit{reth}$. ■

Napomenimo da funkcija $x - y$ nije totalna, pa nije ni primitivno rekurzivna, zbog čega se i razmatra funkcija monus.

Ograničena suma i ograničeni proizvod

Definicija 2.4.9 *Ograničena suma* je funkcija oblika $f_0(x) + \ldots + f_n(x)$ za koju ćemo koristiti oznaku $\sum_{i=0}^{n} f_i(x)$.

Teorema 2.4.10 Neka su funkcije $f_0, \ldots, f_n$ primitivno rekurzivne. Tada je i ograničena suma $\sum_{i=0}^{n} f_i(x)$ primitivno rekurzivna funkcija.

Dokaz. Ako je $n = 0$, funkcija $\sum_{i=0}^{0} f_i(x) = f_0(x)$ je trivijalno primitivno rekurzivna funkcija. Pretpostavimo da je za sve $k < m$ tvrđenje dokazano. Tada je funkcija $\sum_{i=0}^{m} f_i(x) = +(\sum_{i=0}^{m-1} f_i(x), f_m(x))$ primitivno rekurzivna funkcija jer je dobijena kompozicijom primitivno rekurzivnih funkcija. ■

Definicija 2.4.11 *Ograničeni proizvod* je funkcija oblika $f_0(x) \cdot \ldots \cdot f_n(x)$ za koju ćemo koristiti oznaku $\Pi_{i=0}^{n} f_i(x)$.

Teorema 2.4.12 Neka su funkcije $f_0, \ldots, f_n$ primitivno rekurzivne. Tada je i ograničeni proizvod $\Pi_{i=0}^{n} f_i(x)$ primitivno rekurzivna funkcija.

Dokaz. Dokaz se sprovodi analogno dokazu teoreme 2.4.10. ■

Slično se definišu ograničene sume i proizvodi veće arnosti.

[14] Drugi naziv za ovu funkciju je ograničeno oduzimanje.

Primitivno rekurzivni predikati

Predikat je relacija, odnosno neki podskup skupa $\mathbb{N}^k$ za neki prirodan broj $k > 0$. Unarni predikat R je primitivno rekurzivan ako je primitivno rekurzivna njegova karakteristična funkcija

$$C_R(n) = \begin{cases} 1 & \text{ako važi } R(n) \\ 0 & \text{ako ne važi } R(n) \end{cases}$$

i slično za predikate proizvoljne arnosti.

Primer 2.4.13 Predikat $=$, tj. relacija jednakosti, je primitivno rekurzivan jer je primitivno rekurzivna njegova karakteristična funkcija $C_=(x, y) = \dot{-}(sgn(\dot{-}(x, y) + \dot{-}(y, x)), 1)$. Prema zadatku 11.15 funkcija sgn je primitivno rekurzivna. ■

Primer 2.4.14 Predikat $<$, tj. relacija biti manji, je primitivno rekurzivan jer je primitivno rekurzivna njegova karakteristična funkcija $C_<(x, y) = sgn(\dot{-}(x, y))$. ■

Definicija 2.4.15 Neka su P i Q dva unarna predikata.

Konjunkcija predikata P i Q, u oznaci $P \wedge Q$, je unarni predikat sa karakterističnom funkcijom

$$C_{P\wedge Q}(n) = \begin{cases} 1 & \text{ako važi } P(n) \text{ i } Q(n) \\ 0 & \text{ako ne važi } P(n) \text{ i } Q(n) \end{cases}$$

Disjunkcija predikata P i Q, u oznaci $P \vee Q$, je unarni predikat sa karakterističnom funkcijom

$$C_{P\vee Q}(n) = \begin{cases} 1 & \text{ako važi } P(n) \text{ ili } Q(n) \\ 0 & \text{ako ne važi } P(n) \text{ ili } Q(n) \end{cases}$$

Negacija predikata P, u oznaci $\neg P$, je unarni predikat sa karakterističnom funkcijom

$$C_{\neg P}(n) = \begin{cases} 1 & \text{ako ne važi } P(n) \\ 0 & \text{ako važi } P(n) \end{cases}$$

Analogno se definišu konjunkcija, disjunkcija i negacija za k-arne predikate.

Teorema 2.4.16 Ako su P i Q dva primitivno rekurzivna predikata, tada su primitivno rekurzivni i predikati $P \wedge Q$, $P \vee Q$, $\neg P$.

Dokaz. Neka su predikati P i Q unarni. Predikati $P \wedge Q$, $P \vee Q$, $\neg P$ su primitivno rekurzivni jer su im primitivno rekurzivne karakteristične funkcije $C_{P\wedge Q}(n) = C_P(n) \cdot C_Q(n)$, $C_{P\vee Q}(n) = sgn(C_P(n) + C_Q(n))$, $C_{\neg P}(n) = \dot{-}(C_P(n), 1)$. Tvrđenje se slično dokazuje i za k-arne predikate. ■

Primitivno rekurzivni predikati se koriste u daljem definisanju primitivno rekurzivnih funkcija. Slično binarnoj konjunkciji i disjunkciji za neko fiksirano $n \in \mathbb{N}$, definisale bi se i konjunkcija i disjunkcija n predikata i pokazalo da, ako su ti predikati primitivno rekurzivni, da su primitivno rekurzivni predikati n-arne konjunkcije i disjunkcije.

Definicija po slučaju

Neka su $A_0, \ldots, A_m$ unarni primitivno rekurzivni predikati takvi da je za svaki prirodan broj n ispunjen tačno jedan od njih i neka su funkcije $h_0, \ldots h_m$ primitivno rekurzivne. Kažemo da je funkcija f definisana po slučaju ako je oblika

$$f(n) = \begin{cases} h_0(n) & \text{ako važi } A_0(n) \\ \ldots & \\ h_m(n) & \text{ako važi } A_m(n) \end{cases}$$

Analogno se po slučaju definišu k-arne funkcije.

Teorema 2.4.17 Funkcija f definisana po slučaju u odnosu na predikate $A_1, \ldots, A_m$ i funkcije $h_1, \ldots h_m$ je primitivno rekurzivna.

Dokaz. Funkcija f se drugačije može definisati kao ograničena suma $f(x_1, \ldots, x_k) = \sum_{i=0}^{m} h_i(x_1, \ldots, x_k) \cdot C_{A_i}(x_1, \ldots, x_k)$, pa je primitivno rekurzivna. ■

Jednostavan primer primene definicije po slučaju se javlja kada se funkcija definiše eksplicitnim zadavanjem vrednosti za prvih l prirodnih brojeva, a u preostalim slučajevima koristeći neku drugu primitivno rekurzivnu funkciju.

Ograničena kvantifikacija

Definicija 2.4.18 Neka je k prirodan broj i R jedan $k+1$-arni predikat.

Ograničena egzistencijalna kvantifikacija predikata R je $k+1$-arni predikat Q za koji važi $Q(x_1, \ldots, x_k, m)$ ako i samo ako postoji prirodan broj i, takav da je $0 \leq i \leq m$ i važi $R(x_1, \ldots, x_k, i)$. Oznaka za $Q(x_1, \ldots, x_k, m)$ je $(\exists i \leq m)R(x_1, \ldots, x_k, i)$.

Ograničena univerzalna kvantifikacija predikata R je $k+1$-arni predikat Q za koji važi $Q(x_1, \ldots, x_k, m)$ ako i samo ako za svaki prirodan broj i, takav da je $0 \leq i \leq m$ i važi $R(x_1, \ldots, x_k, i)$. Oznaka za $Q(x_1, \ldots, x_k, m)$ je $(\forall i \leq m)R(x_1, \ldots, x_k, i)$.

Teorema 2.4.19 Ako je R primitivno rekurzivni $k+1$-arni predikat, primitivno rekurzivni su i predikati ograničene egzistencijalne i univerzalne kvantifikicije za R.

Dokaz. Primitivno rekurzivna je karakteristična funkcija predikata ograničene univerzalne kvantifikicije $C_{\forall i \leq}(x_1, \ldots, x_k, m) = \Pi_{i=0}^{m} C_R(x_1, \ldots, x_k, i)$. Kako $(\exists i \leq m)R(x_1, \ldots, x_k, i)$ važi ako i samo ako važi $\neg(\forall i \leq m)\neg R(x_1, \ldots, x_k, i)$, to se karakteristična funkcija predikata ograničene egzistencijalne kvantifikicije zapisuje kao $C_{\exists i \leq}(x_1, \ldots, x_k, m) = \dot{-}(\Pi_{i=0}^{m}(\dot{-}(C_R(x_1, \ldots, x_k, i), 1), 1)$, pa je i on primitivno rekurzivan. ■

Sledi nekoliko primera predikata definisanih pomoću ograničene kvantifikacije.

Primer 2.4.20 Neka je P predikat arnosti 4 za koji važi $P(x, y, z, t)$ ako i samo ako $x^t + y^t = z^t$. On je primitivno rekurzivan jer se karakteristična funkcija C_P dobija kompozicijom primitivno rekurzivnih funkcija stepenovanja i karakteristične funkcije $C_=$. Neka je $m \in \mathbb{N}$ fiksiran prirodni broj. Binarni predikat Q definisan tako da važi $Q(k, m)$ ako i samo ako $x^k + y^k = z^k$ za bar neke $x, y, z \leq m$ je primitivno rekurzivan jer se definiše sa $(\exists x \leq m)(\exists y \leq m)(\exists z \leq m)P(x, y, z, k)$. ■

Primer 2.4.21 Neka predikat n*eparno*(n) važi ako i samo ako je n neparan broj. On je primitivno rekurzivan jer važi ako i samo ako važi $(\exists m \leq n)(n = 2 \cdot m + 1)$. Predikat p*arno*$(n)$, tj. biti paran, je primitivno rekurzivan jer važi ako i samo ako važi $(\exists m \leq n)(n = 2 \cdot m)$. ■

Primer 2.4.22 Neka predikat $\div(m, n)$ važi ako i samo m deli n bez ostatka. On je primitivno rekurzivan jer važi ako i samo ako važi $(\exists k \leq n)(n = k \cdot m)$. Predikat p*rost*$(n)$, tj. biti prost broj, je primitivno rekurzivan jer važi ako i samo ako važi $(1 < n) \wedge (\forall m \leq n)((m = 1) \vee (m = n) \vee (\neg \div (m, n))$. ■

Ograničena minimizacija

Definicija 2.4.23 Neka je k prirodan broj i R jedan $k+1$-arni predikat. *Ograničena minimizacija* za predikat R je $k+1$-arna funkcija $f(x_1, \ldots, x_k, n)$ definisana sa

$$(\mu p \leq n)R(x_1, \ldots, x_k, p) = \begin{cases} \text{najmanji prirodni broj } p \leq n & \\ \text{za koji je } R(x_1, \ldots, x_k, p) & \text{ako takav postoji} \\ & \\ 0 & \text{inače} \end{cases}$$

Teorema 2.4.24 Neka je k prirodan broj i R jedan $k+1$-arni primitivno rekurzivni predikat. Ograničena minimizacija za predikat R je primitivno rekurzivna funkcija.

Dokaz. Posmatrajmo funkciju

$$h(p, n, x_1, \ldots, x_k) = \begin{cases} p & \text{ako } (\exists i \leq n)R(x_1, \ldots, x_k, i) \\ n+1 & \text{ako } R(x_1, \ldots, x_k, n+1) \wedge \neg(\exists i \leq n)R(x_1, \ldots, x_k, i) \\ 0 & \text{ako nije } (\exists i \leq n+1)R(x_1, \ldots, x_k, i) \end{cases}$$

Ona je primitivno rekurzivna jer je definisana po slučaju korištenjem primitivno rekurzivnih predikata i funkcija. Sada se funkcija $(\mu p \leq n)R(x_1, \ldots, x_k, p)$ može definisati primitivnom rekurzijom:

$$(\mu p \leq 0)R(x_1, \ldots, x_k, p) = 0$$

$$(\mu p \leq n+1)R(x_1, \ldots, x_k, p) = h((\mu p \leq n)R(x_1, \ldots, x_k, p), n, x_1, \ldots, x_k)$$

U drugom koraku ove definicije, vrednost funkcije je minimalno $p \leq n$ za koje važi $R(x_1, \ldots, x_k, p)$, ako takvo postoji ili $n+1$, ako takvo p ne postoji, a važi $R(x_1, \ldots, x_k, n+1)$ ili 0, ako ni za jedno $p \leq n+1$ nije $R(x_1, \ldots, x_k, p)$. Pošto je h primitivno rekurzivna funkcija, takva je i funkcija ograničene minimizacije $(\mu p \leq n)R(x_1, \ldots, x_k, p)$. ■

Primer 2.4.25 Neka je vrednost funkcije $p(n)$ jednaka n-tom prostom broju. U primeru 2.4.22, pokazano je da je predikat p*rost*(n), tj. biti prost broj, primitivno rekurzivan. Posmatrajmo sada funkciju $h(n, y) = (\mu x \leq (!(n) + 1))((n < x) \wedge p\mathit{rost}(x))$ za koju se kao i u teoremi 2.4.24 može pokazati da je primitivno rekurzivna[15]. Funkciju $p(n)$ definišemo primitivnom rekurzijom

[15] U definiciji funkcije se koristi poznato tvrđenje: ako je p prost broj, onda između p i $p! + 1$ postoji bar još jedan prost broj.

$p(0) = 2,$

$p(n+1) = h(p(n), n).$

Neka funkcija $[x]_n$ izračunava stepen n-tog prostog broja u dekompoziciji broja x. Ona je primitivno rekurzivna jer je $[x]_n = (\mu y \leq x)(\neg \div (p(n)^{y+1}, x))$, tj. $[x]_n$ je najmanje y manje od x sa osobinom da $y+1$-ti stepen n-tog prostog broja ne deli x.

Funkcija dužina dekompozicije prirodnog broja d$uzina(x)$ daje redni broj najmanjeg prostog broja čiji je stepen u dekompoziciji broja x jednak 0. I ova funkcija je primitivno rekurzivna jer se definiše sa d$uzina(x) = (\mu n \leq x)([x]_n = 0)$. Na primer, kako je u nizu prostih brojeva $p(0) = 2$, $p(1) = 3, \ldots$, za $x = 280 = 2^3 \cdot 5^1 \cdot 7^1$ je stepen uz 3, $[x]_1$, jednak nula, pa je d$uzina(x) = 1$. ■

2.4.5 Gedelizacija

U proučavanju modela izračunavanja postupak kodiranja igra značajnu ulogu.

Primer 2.4.26 Jedan postupak kodiranja se sprovodi u računaru. ASCII-znaci se kodiraju brojevima između 0 i 127 (255 kod takozvanog proširenog skupa ASCII-znaka), a zatim se reči zapisuju kao nizovi brojeva, tj. brojevi u nekoj pozicionoj notaciji. Na primer, znak 'a' se kodira sa 97, a znak '*' sa 76. Reč 'a*' se kodira sa $256^1 \cdot 97 + 76$. ■

Kodiranje omogućava da se objektima, a u našem slučaju to mogu biti funkcije ili programi, pridruže prirodni brojevi. Nakon toga se umesto polaznih objekata sredstvima aritmetike razmatraju njihove brojevne slike. Ovaj postupak je veoma moćan jer dozvoljava da se neka funkcija, tj. njena slika, odnosno kod, javlja kao sopstveni argument, što ćemo iskoristiti u dokazima više tvrđenja.

Kodiranje ispunjava da:

- ni koja dva objekta nemaju isti pridruženi broj,
- za svaki objekat možemo u konačnom broju koraka izračunati njemu pridruženi broj i
- za dati prirodan broj može se u konačnom broju koraka proveriti da li je to broj nekog objekta i naći taj objekat, ako postoji,

naziva se *gedelizacija*, a kod pridružen objektu *Gedelov broj*. Ako kodiranje zadovoljava prethodne uslove, onda sama njegova forma nije bitna, jer tvrđenja i dokazi ne zavise od nje.

U odeljku 2.4.11 ćemo detaljno opisati postupak kodiranja jedne klase funkcija koja sadrži sve primitivno rekurzivne funkcije. Prema tome, primitivno rekurzivnih funkcija, a time i unarnih primitivno rekurzivnih funkcija, ima prebrojivo mnogo, te ih je moguće poređati u niz. Pozicija u nizu je *indeks* funkcije.

U kodiranju koje ćemo koristiti značajni su prosti brojevi i jedinstvenost prikazivanja svakog broja u obliku proizvoda prostih brojeva dignutih na neki stepen. Recimo, broj 270 se dekomponuje kao $2^1 \cdot 3^3 \cdot 5^1$. Od pomoćnih funkcija upotrebićemo

funkcije: $p(n)$ (n-ti prost broj), $[x]_n$ (eksponent n-tog prostog broja u dekompoziciji broja x), d$uzina(x)$ (dužina dekompozicije broja x) za koje je u primeru 20 pokazano da su primitivno rekurzivne.

Neka je dat niz brojeva $(a_0, a_1, \ldots, a_n)$. Takav konačan niz brojeva ćemo kodirati na sledeći način:

$$gb(a_0, a_1, \ldots, a_n) = p(0)^{a_0+1} \cdot p(1)^{a_1+1} \cdot \ldots \cdot p(n)^{a_n+1}.$$

Za kodiranje broja a_i koji se nalazi na i-tom mestu u nizu brojeva iz tehničkih razloga koristićemo oblik $p(i)^{a_i+1}$, a za dekodiranje funkciju $(x)_i = \dot{-}(1, [x]_i)$ koja daje za 1 umanjen stepen n-tog prostog broja u dekompoziciji. Razlog za ovo je što množenje sa $p(n)^0 = 1$ ne menja vrednost proizvoda, a komplikuje dekompoziciju. Primetimo da je funkcija gb primitivno rekurzivna. Takođe ćemo usvojiti dogovor da za svaki broj x važi da kodira sekvencu dužine do d$uzina(x)$[16], a da broj x za koji je d$uzina(x) = 0$ kodira praznu reč. Na primer, za $x = 756 = 2^2 \cdot 3^3 \cdot 7 = p(0)^2 \cdot p(1)^3 \cdot p(3)$ je d$uzina(x) = 2$, pa x kodira $(1, 2)$. Kako je $gb(1, 2) = 108$, očigledno je da razni brojevi mogu predstavljati isti niz brojeva. Ali, različiti nizovi se uvek kodiraju raznim brojevima.

2.4.6 Izračunljive i primitivno rekurzivne funkcije

Jedna od posledica mogućnosti nabrajanja svih primitivno rekurzivnih funkcija je da možemo zamisliti intuitivno izračunljivu totalnu funkciju koja nije primitivno rekurzivna. Posmatrajmo samo unarne primitivno rekurzivne funkcije i bilo koje njihovo nabrajanje: $f_0, f_1, f_2, \ldots$ Definišimo unarnu funkciju $g(x) = f_x(x)+1$. Iako je ova funkcija totalna i intuitivno izračunljiva[17], ona nije primitivno rekurzivna. Ako pretpostavimo da funkcija jeste primitivno rekurzivna, ona ima neki indeks k, pa bi prema definiciji funkcije bilo $f_k(k) = g(k) = f_k(k) + 1$, što je kontradikcija.

Upravo primenjen postupak naziva se *dijagonalizacija*.

Sledeća funkcija, nazvana po Akermanu koji ju je definisao 1928. godine:

$$A(0, n, k) = n + k$$

$$A(1, n, 0) = 0$$

$$A(2, n, 0) = 1$$

$$A(m + 1, n, 0) = n, \text{ za } m > 1,$$

$$A(m + 1, n, k + 1) = A(m, n, A(m + 1, n, k)), \text{ za } k \geq 0$$

je intuitivno izračunljiva, recimo može se napisati program na programskom jeziku Pascal, ili program za Tjuringovu mašinu koji za ulazne podatke m, n i k u konačnom, iako možda velikom, broju koraka izračunava vrednost $A(m, n, k)$. Za funkciju A važi: $A(0, n, k) = n + k$, $A(1, n, k) = n \cdot k$, $A(2, n, k) = n^k$, ...i može se pokazati da dijagonala ove funkcije $f(n) = A(n, n, n)$ raste brže od svake unarne

[16] Odnosno, x kodira sekvencu $((x)_0, \ldots, (x)_{duzina(x)-1})$.

[17] Pod pretpostavkom da su primitivno rekurzivne funkcije izračunljive, zamislimo da za x konstruišemo f_x, izračunamo $f_x(x)$ i dodamo 1.

primitivno rekurzivne funkcije, pa da ni funkcija A nije primitivno rekurzivna[18]. Isto važi i za sličnu funkciju definisanu sa $A'(0,n) = n+1$, $A'(m+1,0) = A'(m,1)$, $A'(m+1,n+1) = A'(m,A'(m+1,n))$.

Prema tome, iako se u klasi primitivno rekurzivnih funkcija nalazi veliki broj interesantnih funkcija, postoje funkcije koje jesu intuitivno izračunljive a koje ne pripadaju klasi primitivno rekurzivnih funkcija. U odeljku 2.4.7 ćemo videti koje su još operacije nad osnovnim funkcijama potrebne za širenje klase funkcija.

2.4.7 Parcijalno rekurzivne funkcije

Jedno od ograničenja koje u sebi ima klasa primitivno rekurzivnih funkcija je da je zatvorena za operaciju minimizacije koja sme biti samo ograničena. Klasa parcijalno rekurzivnih funkcija se dobija kada se eliminiše to ograničenje. Pokazaćemo da je to dovoljno da se klasa funkcija proširi toliko da se poklopi sa Tjuring-izračunljivim funkcijama.

2.4.8 Definicija parcijalno rekurzivnih funkcija

Definicija 2.4.27 Klasa *parcijalno rekurzivnih funkcija* sadrži osnovne funkcije:

- nula funkciju $Z(n) = 0$, za svako $n \in \mathbb{N}$,
- funkciju naslednika $S(n)$ prirodnog broja n u nizu prirodnih brojeva,
- funkcije projekcije $P_k^i(x_1,\ldots,x_k) = x_i$ za sve i i k takve da je $1 \leq i \leq k$

i sve funkcije koje se od njih dobijaju konačnim brojem primena operacija:

- *kompozicije*, tj. ako su već definisane m-arna funkcija g i k-arne funkcije h_1, $\ldots$, h_m, definisana je i k-arna funkcija f tako da je za sve $(x_1,\ldots,x_k) \in \mathbb{N}^k$:

$$f(x_1,\ldots,x_k) = g(h_1(x_1,\ldots,x_k),\ldots,h_m(x_1,\ldots,x_k)),$$

- *primitivne rekurzije*, tj. ako su već definisane m-arna funkcija g i $m+2$-arna funkcija h, definisana je i $m+1$-arna funkcija f tako da je za sve $(x_1,\ldots,x_m) \in \mathbb{N}^m$.
  - $f(0,x_1,\ldots,x_m) = g(x_1,\ldots,x_m)$ i
  - $f(n+1,x_1,\ldots,x_m) = h(f(n,x_1,\ldots,x_m),n,x_1,\ldots,x_m)$ za sve $n \in \mathbb{N}$.

  Pri tome je funkcija f definisana primitivnom rekurzijom nad h sa bazom g i

[18] Primetimo da se rekurzija u Akermanovoj funkciji sprovodi istovremeno po dva argumenta - prvom i trećem. Pokazalo se da se istovremenom rekurzijom po tri, četiri itd. argumenta dobijaju redom sve šire i šire klase funkcija koje su intuitivno izračunljive i čine takozvanu Grzegorczyk-ovu hijerarhiju. Pokazuje se da je klasa funkcija koje se definišu rekurzijom po n argumenata pravi podskup klase funkcija definisanih rekurzijom nad $n+1$ argumentom, za svaki $n \geq 0$. Zanimljivo je da klasa intutivno izračunljivih funkcija nije podskup ni jedne klase u toj hijerarhiji.

- *neograničene minimizacije*, tj. ako je već definisana $k+1$-arna funkcija f, definisana je i funkcija

$$(\mu y)(f(x_1,\ldots,x_k,y)=0)=\begin{cases}\text{najmanje } z \text{ za koje je } f(x_1,\ldots,x_k,z)=0 \text{ i}\\ \text{za svako } y<z \text{ je } f(x_1,\ldots,x_k,y) \text{ definisano}\\ \\ \text{nedefinisano, inače.}\end{cases}$$

Primetimo da je $(\mu y)(f(x_1,\ldots,x_k,y)=0)$ u stvari funkcija arnosti k čiji argumenti su x_1, ..., x_k i da je intuitivno izračunljiva sledećim postupkom. Najpre se računa $f(x_1,\ldots,x_k,0)$. Ako je funkcija definisana za te argumente i ima vrednost 0, rezultat je 0. Ako je funkcija definisana za te argumente, a vrednost joj nije 0, prelazi se na izračunavanje $f(x_1,\ldots,x_k,1)$. Ako funkcija nije definisana za argumente $(x_1,\ldots,x_k,0)$ ni funkcija $(\mu y)(f(x_1,\ldots,x_k,y)=0)$ neće biti definisana. Operacijom neograničene minimizacije mogu se dobiti i parcijalne funkcije.

Primer 2.4.28 Funkcija $(\mu y)(\frac{x}{y}-1=0)$ nije definisana ni za jedno x pošto nije definisano ni deljenje nulom. ■

Očigledno je da prema definiciji 2.4.27 klasa parcijalno rekurzivnih funkcija sadrži sve primitivno rekurzivne funkcije, pošto se u njoj nalaze osnovne funkcije i zatvorena je za osnovne operacije. Sa druge strane, u klasi parcijalno rekurzivnih funkcija se nalaze i funkcije koje nisu totalne, pa samim tim nisu ni primitivno rekurzivne[19]

Definicija 2.4.29 *Rekurzivne funkcije* su totalne parcijalno rekurzivne funkcije.

Definicija 2.4.30 Predikat R je *rekurzivan* ako je njegova karakteristična funkcija C_R rekurzivna.

Rekurzivi predikati se često nazivaju i *odlučivim* predikatima.

Oznake $f(x_1,\ldots,x_k)\downarrow$, $f(x_1,\ldots,x_k)\downarrow y$ i $f(x_1,\ldots,x_k)\uparrow$, analogno oznakama za Tjuringove mašine, znače da je funkcija f arnosti k i da je definisana za argumente x_1, ..., x_k, da je njen rezultat y, odnosno da nije definisana za te argumente.

Definicija 2.4.31 Dve parcijalno rekurzivne funkcije f i g su *jednake*[20] ($f\simeq g$) ako su iste arnosti i za iste arumente ili obe nisu definisane ($f(\ldots)\uparrow$ i $g(\ldots)\uparrow$) ili su obe definisane ($f(\ldots)\downarrow$ i $g(\ldots)\downarrow$) i imaju istu vrednost ($f(\ldots)=g(\ldots)$).

2.4.9 Parcijalno rekurzivne operacije

Slično primitivno rekurzivnim operacijama i ovde postoje operacije čija korektnost se dokazuje u odnosu na definiciju 2.4.27, a koje primenjene na parcijalno rekurzivne funkcije daju nove parcijalno rekurzivne funkcije.

[19]Takođe, postoje i totalne funkcije koje nisu primitivno rekurzivne, a koje jesu parcijalno rekurzivne. Primer takve funkcije je Akermanova funkcija.

[20]Ekstenzionalno se poklapaju, tj. poklapaju se kao skupovi uredjenih $k+1$-torki.

Teorema 2.4.32 Ako je predikat $R \subset \mathbb{N}^{k+1}$ rekurzivan, a h $k+1$-arna parcijalno rekurzivna funkcija, parcijalno rekurzivne su i sledeće funkcije:

- $(\mu y)(h(x_1, \ldots x_k, y) = a)$, za $a \in \mathbb{N}$ i
- $(\mu y)(R(x_1, \ldots x_k, y))$.

Dokaz. U prvom slučaju umesto $h(x_1, \ldots x_k, y)$ dovoljno je posmatrati funkciju $sgn(\dot{-}(h(x_1, \ldots x_k, y), a) + \dot{-}(a, h(x_1, \ldots x_k, y))$ i tvrđenje sledi neposredno. U slučaju funkcije $(\mu y)(R(x_1, \ldots x_k, y))$, umesto predikata $R(x_1, \ldots x_k, y)$ posmatra se jednakost $C_R(x_1, \ldots x_k, y) = 1$. ■

2.4.10 Izračunljive i parcijalno rekurzivne funkcije

U odeljku 2.4.11 ćemo prikazati jedan postupak gedelizacije parcijalno rekurzivnih funkcija, odakle sledi da ovih funkcija ima prebrojivo mnogo. Posebno, unarnih parcijalno rekurzivnih funkcija ima prebrojivo mnogo. U analizi odnosa izračunljivih i primitivno rekurzivnih funkcija u odeljku 2.4.6 to je iskorišteno da bi se primenio postupak dijagonalizacije. Ovde, međutim tako nešto nije moguće, jer iako se može definisati $g(x) = f_x(x) + 1$, možda $f_x(x)$ nije definisano. Sledeća teorema pokazuje da se taj problem ne može zaobići, tj. da se problem da li je funkcija f_x definisana za x ne može rešiti u opštem slučaju.

Teorema 2.4.33 Ne postoji rekurzivna funkcija definisana sa

$$g(x) = \begin{cases} 1 & \text{ako je } f_x(x) \text{ definisano} \\ 0 & \text{ako } f_x(x) \text{ nije definisano.} \end{cases}$$

Dokaz. Pretpostavimo da je moguće definisati ovu funkciju. Tada bi funkcija

$$h(x) = \begin{cases} \text{nedefinisana} & \text{ako je } f_x(x) \text{ definisano} \\ 0 & \text{ako } f_x(x) \text{ nije definisano} \end{cases}$$

bila parcijalno rekurzivna funkcija oblika $(\mu y)(y + g(x) = 0)$, pa bi i ona bila jednaka funkciji f_l, za neki indeks l, iz niza svih unarnih parcijalno rekurzivnih funkcija. Ali, tada bi bilo

$$h(l) = \begin{cases} \text{nedefinisano} & \text{ako je } h(l) = f_l(l) \text{ definisano} \\ 0 & \text{ako } h(l) = f_l(l) \text{ nije definisano} \end{cases}$$

što je kontradikcija. ■

2.4.11 Nabrajanje parcijalno rekurzivnih funkcija

Postupak koji ćemo primeniti u kodiranju parcijalno rekurzivnih funkcija u osnovi ima sledeće korake koji koriste kodiranje nizova brojeva opisano u odeljku 2.4.5:

- osnovne funkcije se kodiraju sa $gb(Z) = 0$, $gb(S) = gb(1)$, $gb(P_k^i) = gb(1, i)$,

- ako su $gb(g) = a$, $gb(h_1) = b_1$, ..., $gb(h_m) = b_m$ već određeni, onda je broj funkcije koja se dobija kompozicijom ovih funkcija jednak $gb(g(h_1, \ldots, h_m)) = gb(a, gb(b_1, \ldots, b_m), 0)$,
- ako su $gb(g) = a$ i $gb(h) = b$ već određeni, onda je broj funkcije koja se dobija primitivnom rekurzijom nad h sa osnovom g jednak $gb(a, b, 0, 0)$ i
- ako je $gb(g) = a$ već određen, onda je broj funkcije koja se dobija neograničenom minimizacijom $(\mu y)(g(x_1, \ldots, x_k, y) = 0)$ jednak $gb(a, 0, 0, 0, 0)$.

Broj promenljivih ne figuriše u samom kodiranju, pa se on određuje prilikom dekodiranja, što na neki način odgovara izračuvanju funkcija sa promenljivim brojem argumenata koje je moguće definisati, recimo u programskom jeziku C. Primetimo da su u postupku kodiranja različitim funkcijama i operacijama pridruženi kodovi nizova različite dužine. Prilikom dekodiranja ćemo koristi dogovor da broj x kodira sekvencu dužine do $duzina(x)$, zbog čega će beskonačno mnogo različitih brojeva označavati istu funkciju. Na primer, funkciji koja se dobija primenom primitivne rekurzije nad funkcijama sa indeksima a i b biće pridružen svaki broj x sa osobinom da je $duzina(x) = 4$ i $(x)_0 = a$, $(x)_1 = b$. Takođe, da bismo obezbedili da je svaki prirodan broj kod neke funkcije, pretpostavićemo da, ako je $duzina(x) \geq 5$, onda x kodira funkciju dobijenu neograničenom minimizacijom funkcije $f_{(x)_0}$.

Prilikom raspakivanja broja x, zbog jedinstvenosti dekompozicije na proste faktore, redom će se dobijati konačni nizovi prirodnih brojeva sve dok se ne stigne do potpunog opisa odgovarajuće funkcije, odnosno opisa kako se ta funkcija definiše i izračunava. Recimo, neka je $x = gb(3276, 817, 0, 0)$. Tada je funkcija f_x dobijena primenom primitivne rekurzije na funkciju f_{817} sa bazom koju predstavlja funkcija f_{3276}. Dalje se analizom indeksa 817 i 3276 postupak ponavlja dok se ne stigne do osnovnih funkcija. Pošto se kodiranje sprovodi u skladu sa načinom definisanja funkcija, dekodiranje dovodi do opisa funkcije koji koristi osnovne funkcije i operacije nad njima što omogućava izračunavanje funkcije za proizvoljne argumente. To nam pruža mogućnost da definišemo jednu parcijalno izračunljivu funkciju koja će simulirati rad svih parcijalno izračunljivih funkcija, o čemu će biti reči u odeljku 2.4.12.

Postupak dekodiranja ćemo sada opisati preciznije. Najpre, ponovimo da arnost funkcija ne figuriše u samom kodiranju, tako da isti kod dobijaju odgovarajuće funkcije sa različitim brojem argumenata, na primer $gb(P^i_k) = gb(P^i_{k+1})$. Pretpostavimo da su promenljive uvek obeležene sa $x_1, \ldots, x_k$ i da je funkcija sa $k \geq 1$ promenljivih čiji je indeks n označena sa f^k_n. Ako je:

- $duzina(n) = 0$, tj. n kodira prazni niz brojeva, reč je o nula-funkciji Z,
- $duzina(n) = 1$, tj. n kodira jednočlani niz brojeva $(n)_0$, reč je o funkciji naslednika S,
- $duzina(n) = 2$, tj. n kodira niz brojeva $((n)_0, (n)_1)$ i ako je:
  - $(n)_0 = 0$, ili se projektuje po koordinati $(n)_1 = 0$, reč je o nula-funkciji Z,
  - $(n)_0 \geq 1$ i projektuje se po koordinati $1 \leq (n)_1 \leq k$, reč je o funkciji $P^{(n)_1}_k$ i

  - $(n)_0 \geq 1$ i projektuje se po koordinati $(n)_1 > k$, reč je o funkciji P_k^k,

- $duzina(n) = 3$, tj. n kodira niz brojeva $((n)_0, (n)_1, (n)_2)$ i ako je:
  - kod funkcija argumenata $duzina((n)_1) = 0$, reč je o funkciji $f^k_{(n)_0}$ i
  - kod funkcija argumenata $duzina((n)_1) \geq 1$, reč je o funkciji dobijenoj kompozicijom $f^{duzina((n)_1)}_{(n)_0}(f^k_{((n)_1)_0}, \ldots, f^k_{((n)_1)_{\dot{-}(1,duzina((n)_1))}})$,

- $duzina(n) = 4$, tj. n kodira niz brojeva $((n)_0, (n)_1, (n)_2, (n)_3)$ i ako je
  - broj argumenata funkcije $k = 1$, tada je reč o funkciji $f^1_n(0) = (n)_0$, $f^1_n(x_1 + 1) = f^2_{(n)_1}(f^1_n(x_1), x_1)$ dobijenoj primitivnom rekurzijom i
  - broj argumenata funkcije $k > 1$, tada je reč o funkciji $f^k_n(0, x_2, \ldots, x_k) = f^{k-1}_{(n)_0}(x_2, \ldots, x_k)$, $f^k_n(x_1+1, x_2, \ldots, x_k) = f^{k+1}_{(n)_1}(f^k_n(x_1, x_2, \ldots, x_k), x_1, x_2, \ldots, x_k)$ dobijenoj primitivnom rekurzijom i

- $duzina(n) \geq 5$, tj. n kodira niz brojeva $((n)_0, (n)_1, (n)_2, (n)_3, (n)_4, \ldots)$ onda je $f^k_n(x_1, x_2, \ldots, x_k) = (\mu x_{k+1})(f^{k+1}_{(n)_0}(x_1, x_2, \ldots, x_{k+1}) = 0)$.

Konačno, možemo zaključiti da parcijalno rekurzivnih funkcija ima prebrojivo mnogo i da postoji postupak njihovog nabrajanja koji je primitivno rekurzivan.

2.4.12 Univerzalni predikat i univerzalna funkcija

U ovom odeljku ćemo pokazati da postoji univerzalni predikat pomoću kojeg možemo proveriti za proizvoljnu funkciju f i neke fiksirane argumente da li je funkcija definisana i njena vrednost jednaka nekom fiksiranom broju. Sa $UP(n, gb(a_1, \ldots, a_k), b, y)$ ćemo označiti da je funkcija f^k_n definisana za argumente $a_1, \ldots, a_k$, da je $f^k_n(a_1, \ldots, a_k) = b$ i da je najveći broj koji se koristi u tom izračunavanju manji do jednak od y.

Teorema 2.4.34 Postoji primitivno rekurzivni predikat UP arnosti 4 tako da je za sve n i k, $f^k_n(a_1, \ldots, a_k) = b$ ako i samo ako $(\exists y)UP(n, gb(a_1, \ldots, a_k), b, y)$.

Dokaz. Predikat UP ćemo definisati po slučaju koristeći pri tome postupak kodiranja opisan u odeljku 2.4.11. Pošto to kodiranje garantuje da je svaki prirodan broj kod neke funkcije, predikat koga razmatramo će biti uvek definisan. Praveći paralelu sa slučajevima koje smo tom prilikom razmatrali, predikat $UP(n, a, b, y)$ važi u sledećim slučajevima:

- ako $duzina(n) = 0$ i $b = 0$ (slučaj nula funkcije),
- ako $duzina(n) = 1$ i $b = (a)_0 + 1$ (slučaj funkcije naslednika),
- ako $duzina(n) = 2$ (slučaj funkcije projekcije) i
  - ako je $(n)_0 = 0$ ili $(n)_1 = 0$ i $b = 0$,
  - ako je $(n)_0 \geq 1$ i $1 \leq (n)_1 \leq duzina(a)$ i $b = (a)_{\dot{-}(1,(n)_1)}$,
  - ako je $(n)_0 \geq 1$ i $k < (n)_1$ i $b = (a)_{\dot{-}(1,duzina(a))}$,

- ako d$uzina(n) = 3$ (slučaj kompozicije) i
  - ako je d$uzina((n)_1) = 0$ i važi $UP((n)_0, a, b, y)$,
  - ako je d$uzina((n)_1) \geq 1$ i postoji $d \leq y$ koje kodira rezultate funkcija argumenata kompozicije za koje je d$uzina(d) =$ d$uzina((n)_1)$ i važe predikati $UP(((n)_1)_i, a, (d)_i, y)$, $0 \leq i \leq \dot{-}(1,$ d$uzina((n)_1))$, i $UP((n)_0, d, b, y)$,
- ako d$uzina(n) = 4$ (slučaj primitivne rekurzije) i
  - ako je d$uzina(a) = 1$ i $(a)_0 = 0$ i $b = (n)_0$[21],
  - ako je d$uzina(a) = 1$ i $(a)_0 \geq 1$ i postoji e, $0 < e \leq y$ i važi $UP(n, \frac{a}{2}, e, y)$ i $UP((n)_1, gb(e, \dot{-}(1, (a)_0)), b, y)$[22],
  - ako je d$uzina(a) > 1$ i $(a)_0 = 0$ i važi predikat $UP((n)_0, gb((a)_1, \ldots, (a)_{\dot{-}(1,$ d$uzina(a))}), b, y)$[23],
  - ako je d$uzina(a) > 1$ i $(a)_0 \geq 1$ i postoji e, $0 < e \leq y$ tako da je $UP(n, \frac{a}{2}, e, y)$ i $UP((n)_1, gb(e, \dot{-}(1, (a)_0), (a)_1, \ldots, (a)_{\dot{-}(1,$ d$uzina(a))}), b, y)$[24],
- ako d$uzina(n) \geq 5$ i važi $UP((n)_0, a \cdot p($d$uzina(a))^{1+b}, 0, y)$[25] i za svaki $i < b$ postoji e, $0 < e < y$ tako da važi $UP((n)_0, a \cdot p($d$uzina(a))^{1+i}, e, y)$ (slučaj neograničene minimizacije).

Razmotrimo detaljnije nekoliko od prethodnih slučajeva. Najpre se podsetimo da argumenti predikata $UP(n, a, b, y)$ označavaju redom indeks funkcije, kod argumenata, rezultat i jedno gornje ograničenje brojeva korištenih u izračunavanju. Ako je d$uzina(n) = 0$ radi se o funkciji Z pa predikat UP važi za $b = 0$, bez obzira na vrednosti ostalih argumenata. Ako je d$uzina(n) = 1$ radi se o funkciji S pa predikat UP važi za $b = (a)_0 + 1$ bez obzira na vrednosti ostalih argumenata, pri čemu je $(a)_0$ vrednost prve koordinate u dekompoziciji broja a. Ako je d$uzina(n) = 2$ radi se o funkciji projekcije P_k^i koja se kodira sa $n = gb(1, i) = gb((n)_0, (n)_1)$ i mogući su sledeći slučajevi. Ako je $(n)_0 = 0$ ili se projektuje po koordinati $(n)_1 = 0$ reč je o degenerisanom slučaju koji predstavlja funkciju Z, pa je rezultat $b = 0$. Ako je $(n)_0 \geq 1$ i projektuje se po koordinati $1 \geq (n)_1 \geq$ d$uzina(a)$, rezultat je argument a_i za $i = \dot{-}(1, (n)_1)$. Konačno, ako je $(n)_0 \geq 1$ i za koordinatu projekcije važi $k < (n)_1$, rezultat je $b = (a)_{\dot{-}(1,$ d$uzina(a))}$. Ako je d$uzina(n) \geq 5$, radi se o neograničenoj minimizaciji. Vrednost $(n)_0$ prve koordinate dekompozicije broja n je indeks funkcije po kojoj se radi minimizacija, $a \cdot p($d$uzina(a))^{1+b}$ je kod argumenata funkcije $f_{(n)_0}$ koja za te argumente ima vrednost 0, a y je ograničenje brojeva korištenih u izračunavanju. Slično je i u preostalim slučajevima.

[21] Odnosi se za slučaj funkcije oblika $f(0) = c$.

[22] Odnosi se na slučaj funkcije oblika $f(x + 1) = h(f(x), x)$. Izraz $\frac{a}{2}$ znači da se za 1 smanjuje argument.

[23] Odnosi se za slučaj funkcije oblika $f(0, \ldots) = g(\ldots)$.

[24] Odnosi se na slučaj funkcije oblika $f(x + 1, \ldots) = h(f(x, \ldots), x, \ldots)$. Izraz $\frac{a}{2}$ znači da se za 1 smanjuje argument po kome se sprovodi indukcija.

[25] funkcija $p(x)$ je primitivno rekurzivna funkcija koja izračunava x-ti prost broj.

Na osnovu definicije lako se uočava da je predikat UP primitivno rekurzivan. Preostaje da se pokaže da je $f_n^k(a_1, \ldots, a_k) = b$ ako i samo ako je $(\exists y)UP(n, gb(a_1, \ldots, a_k), b, y)$.

Dokaz sa leva na desno se sprovodi indukcijom po n i po $a = gb(a_1, \ldots, a_k)$. Za slučajeve $duzina(n) \leq 2$ radi se o osnovnim funkcijama i tvrđenje je istinito. Neka za svako $l < n$ tvrđenje važi, kao i za n i sve $x < a$. Neka je $f_n^k(a_1, \ldots, a_k) = b$ i $duzina(n) = 5$, odnosno radi se o funkciji dobijenoj neograničenom minimizacijom. Tada je $f_{(n)_0}^{k+1}(a_1, \ldots, a_k, b) = 0$ i pošto je $(n)_0 < n$ po indukcijskoj hipotezi postoji neko v tako da važi $UP((n)_0, gb(a_1, \ldots, a_k, b), 0, v)$. Dalje, za svako $i < b$ je $f_{(n)_0}^{k+1}(a_1, \ldots, a_k, i)$ definisano i veće od nule. Prema indukcijskoj hipotezi postoje $u_0, \ldots, u_{b-1}$ i $v_0, \ldots, v_{b-1}$ takvi da za svako $i \leq b-1$ važi $UP((n)_0, gb(a_1, \ldots, a_k, i), u_i, v_i)$, gde je $u_i > 0$. Definišimo $y = \max\{u_0, \ldots, u_{b-1}, v_0, \ldots, v_{b-1}, b\} + 1$. Zaključujemo da važi $UP(n, gb(a_1, \ldots, a_k), b, y)$. Slično se analiziraju i ostali slučajevi.

Dokaz sa desna na levo je direktna posledica definicije predikata UP. ■

Predikat UP ćemo nazvati *univerzalni predikat*.

Posmatrajmo sada funkciju $f_U(n, a) = ((\mu z)(UP(n, a, (z)_0, (z)_1)))_0$ čiji argumenti su redom kod funkcije n i kod niza argumenata $a = gb(a_1, \ldots, a_k)$ koja je definisana ako postoji $z = gb(b, y)$, kod dvočlanog niza sastavljenog od rezultata b funkcije f_n i ograničenja y za maksimalno dozvoljeni broj koji se koristi u izračunavanju funkcije f_n. Rezultat funkcije je prva koordinata u dekompoziciji broja z, tj. b. Ova funkcija je parcijalno rekurzivna pošto je UP primitivno rekurzivni predikat i naziva se *univerzalna funkcija*. Na osnovu prethodne teoreme, za svaku parcijalno rekurzivnu funkciju f arnosti k, njen kod n i sve k-torke argumenata važi $f_n(a_1, \ldots, a_k) \simeq ((\mu z)(UP(n, a, (z)_0, (z)_1)))_0$. Na osnovu definicije je jasno da je univerzalna funkcija parcijalno rekurzivna, ali nije totalna. Ona zavisi od izbora kodiranja pomoću koga nabrajamo parcijalno rekurzivne funkcije, ali, prema teoremi 2.4.34, za svako fiksirano kodiranje postoji univerzalna funkcija. Primetimo analogiju između ove funkcije i univerzalne Tjuringove mašine UTM: funkcija izračunava sve ostale funkcije kao što UTM simulira izvršavanje svih ostalih Tjuringovih mašina.

Teorema 2.4.35 Svaka parcijalno rekurzivna funkcija se može definisati korištenjem najviše jednog operatora neograničene minimizacije.

Dokaz. Tvrđenje je direktna posledica postojanja univerzalne funkcije u čijoj definiciji figuriše samo primitivno rekurzivni predikat UP, odnosno njegova karakteristična funkcija, u čijoj definiciji nema operatora neograničene minimizacije, a na koji je je primenjen tačno jedan operator μ. ■

Teorema 2.4.35 se po svom autoru naziva Klinijeva teorema o normalnoj formi.

2.4.13 s-m-n-teorema i teorema o fiksnoj tački

Ako posmatramo neku parcijalno rekurzivnu funkciju $f(x, y)$ i pretpostavimo da fiskiramo vrednost prvog argumenta tako da bude jednaka nekom parametru a

možemo posmatrati unarnu funkciju $f(a, y)$ čiji jedini argument je y. I ta nova funkcija je parcijalno rekurzivna, čak se može izračunati i njen indeks. U sledećoj teoremi iskazano je uopštenje ovog tvrđenja.

Teorema 2.4.36 (s-m-n-teorema) Neka su prirodni brojevi $m, n \geq 1$. Tada postoji $m+1$-arna rekurzivna funkcija S_n^m takva da za proizvoljnu $m+n$-arnu parcijalno rekurzivnu funkciju f_e važi

$$f_e(x_1, \ldots, x_m, y_1, \ldots, y_n) \simeq f_{S_n^m(e, x_1, \ldots, x_m)}(y_1, \ldots, y_n).$$

Dokaz. Neka je funkcija f nastala iz funkcije f_e zamenom prvih m argumenata konstantama $a_1, \ldots, a_m$. Preciznije, funkcija f je nastala kompozicijom funkcije f_e, konstantnih funkcija i funkcija projekcije tako da je

$$f(y_1, \ldots, y_n) \simeq f_e \quad (\quad K_n^{a_1}(y_1, \ldots, y_n), \ldots, K_n^{a_m}(y_1, \ldots, y_n), \\ P_n^1(y_1, \ldots, y_n), \ldots, P_n^n(y_1, \ldots, y_n)).$$

Za fiksirane e, i $a_1, \ldots, a_m$, Gedelov broj funkcije f je

$$S_n^m(e, a_1, \ldots, a_m) = gb(e, gb(gb(K_n^{a_1}), \ldots, gb(K_n^{a_m}), gb(P_n^1), \ldots, gb(P_n^n)), 0).$$

Funkcija S_n^m koja izračunava taj broj je primitivno rekurzivna, pa i rekurzivna, jer je definisana kao kompozicija primitivno rekurzivnih funkcija. ■

s-m-n-teorema se naziva i teorema o parametrizaciji ili teorema iteracije. Imajući u vidu ekvivalentnost pojmova izračunljivih funkcija i programa, u teoremi se zapravo kaže da se podaci mogu efektivno ugraditi u program. A kako se programi mogu kodirati podacima, teorema u suštini uvodi postupak koji se može interpretirati kao pozivanje potprograma.

Teorema 2.4.37 (O fiksnoj tački) Ako je h unarna rekurzivna funkcija, postoji prirodan broj n tako da je $f_n \simeq f_{h(n)}$, gde su f_n i $f_{h(n)}$ parcijalno rekurzivne funkcije sa indeksima n odnosno $h(n)$.

Dokaz. Posmatrajmo parcijalno rekurzivnu funkciju $g(x, y) \simeq f_{h(f_x(x))}(y)$. Prema s-m-n-teoremi postoji rekurzivna funkcija $S(x)$ koja daje indeks tako da je $g(x, y) \simeq f_{S(x)}\ (y)$[26], pa je i $f_{h(f_x(x))}(y) \simeq f_{S(x)}(y)$. I sama funkcija S ima svoj indeks m. Fiksirajući $x = m$ i $S(m) = n$ dobijamo $f_{h(f_m(m))}(y) \simeq f_{S(m)}(y)$, odnosno $f_{h(n)}(y) \simeq f_n(y)$. ■

Teorema 2.4.37 se naziva i *teorema o rekurziji*. Primetimo da se ni u njenoj formulaciji, ni u dokazu, ne insistrira da je $n = h(n)$, već da su jednake parcijalne funkcije čiji su n i $h(n)$ indeksi. I pored toga broj n iz teoreme se naziva *fiksna tačka* funkcije h.

[26] Primetimo da u formulaciji s-m-n-teoreme funkcija S ima i argument koji je indeks funkcije. Ovde se, međutim, razmatra fiksna funkcija g, pa je taj indeks takođe fiksiran i ne smatra se argumentom funkcije.

Pored teoreme 2.4.37 postoji još jedna teorema rekurzije koja ima sličnu formulaciju, ali se odnosi na operatore, tj. funkcije koje kao domene i kodomene imaju klase funkcija. Na primer, jedan operator h^* indukovan funkcijom h preslikava funkcije u funkcije, tako da je $h^*(f_x) = f_{h(x)}$. U toj teoremi se dokazuje da svaki rekurzivni operator koji preslikava klasu n-arnih parcijalno rekurzivnih funkcija u samu sebe ima najmanju nepokretnu tačku. Analogno možemo posmatrati i operatore koji preslikavaju programe, recimo za Tjuringovu mašinu, u programe. Recimo, neka je $h^*(P_x) = P_{h(x)}$, gde su P_x i $P_{h(x)}$ programi koji redom izračunavaju funkcije f_x i $f_{h(x)}$. Druga teorema rekurzije se odnosi i na takve operatore što ima primenu u semantičkoj analizi programskih jezika jer omogućava da se odredi značenje rekurzivnih programa.

Do kraja odeljka ćemo proanalizirati neke posledice teoreme 2.4.37.

Teorema 2.4.38 Svaka unarna rekurzivna funkcija ima beskonačno mnogo fiksnih tačaka.

Dokaz. Pokazaćemo da za svaku unarnu rekurzivnu funkciju h i za svaki prirodan broj m postoji prirodan broj $n > m$ tako da je $f_n \simeq f_{h(n)}$. Posmatrajmo skup $M = \{f_1, \ldots, f_m\}$ prvih m parcijalno rekurzivnih funkcija. Pošto postoji beskonačno mnogo međusobno različitih parcijalno rekurzivnih funkcija, postoji i parcijalno rekurzivna funkcija $f_c \notin M$. Definišimo funkciju

$$g(x) = \begin{cases} c & \text{za } x \leq m \\ h(x) & \text{za } x > m \end{cases}$$

I ova funkcija je rekurzivna i ima bar jednu fiksnu tačku n_0 koja je veća od m jer za $n \leq m$ je $f_{g(n)} = f_c \not\simeq f_n$. Prema definiciji funkcije g, n_0 je istovremeno i fiksna tačka funkcije h. ■

Teorema 2.4.39 Ako je $f(x, y)$ parcijalno rekurzivna funkcija, onda postoji prirodan broj n takav da je $f(n, y) \simeq f_n(y)$.

Dokaz. Neka je indeks n fiksna tačka funkcije $S_1^1(x)$ iz teoreme 2.4.36, tj. $f_{S_1^1(n)} \simeq f_n$. Tada je $f(x, y) \simeq f_{S_1^1(x)}(y)$ i $f(n, y) \simeq f_{S_1^1(n)}(y) \simeq f_n(y)$. ■

Primer 2.4.40 Primenjujući teoremu 2.4.39 na funkciju $f(y, x) = x^y$ zaključujemo da postoji broj $n \in \mathbb{N}$ takav da je $f(n, x) \simeq f_n(x)$, odnosno da je $f_n(x) \simeq x^n$. Slično postoji broj $n \in \mathbb{N}$ takav da je $Dom(f_n) = \{n\}$, jer za funkciju

$$g(x, y) = \begin{cases} 0 & \text{za } x = y \\ \uparrow & \text{za } x \neq y \end{cases}$$

postoji $n \in \mathbb{N}$ tako da je $f(n, y) \simeq f_n(y)$ i pri tome je $Dom(f_n) = \{n\}$. ■

Funkcija f je *samoreprodukujuća* ako važi $f(x) \downarrow \text{g}b(f)$, tj. bez obzira na argument funkcija daje svoj sopstveni kod, tj. indeks funkcije.

Teorema 2.4.41 Postoji samoreprodukujuća funkcija.

Dokaz. Posmatrajmo parcijalno rekurzivnu funkciju $f(y, x) = y$. Prema teoremi 2.4.39 postoji prirodan broj n takav da je $f(n, x) = f_n(x) = n$. ■

Zanimljivo je da se analogno razmatranje može primeniti i na programe i pokazati da postoji *samoreprodukujući program* koji kao izlaz daje svoj sopstveni kod. Ako analiziramo značenje ovakvog tvrđenja videćemo da je taj program upravo ono što se u savremenom računarstvu naziva *programski virus*, koji se razmnožava šireći svoje kopije tokom izvršavanja. Pojava samoreprodukujućih mašina donekle protivreči intuiciji da mašina mora biti složenija od svog proizvoda jer mora sadržati bar nacrt konstrukcije tog proizvoda, ali je očigledno i teorijski i praktično potvrđena.

2.5 Tjuring-izračunljive i parcijalno rekurzivne funkcije

Iako su međusobno različiti, modeli izračunavanja definisani Tjuringovom mašinom i klasom parcijalno rekurzivnih funkcija međusobno su ekvivalentni, tj. određuju jednu te istu klasu izračunljivih funkcija.

Teorema 2.5.1 Svaka parcijalno rekurzivna funkcija je Tjuring-izračunljiva.

Dokaz. Primetimo najpre da za sve osnovne funkcije (Z, S, P_k^j) postoje odgovarajuće Tjuringove mašine, što je i pokazano u primerima 2.3.8, 2.3.9 i 2.3.10. Ostaje da se pokaže da je Tjuringovim mašinama moguće simulirati operacije kompozicije, primitivne rekurzije i neograničene minimizacije.

Neka su $h_1, \dots h_m$ parcijalno rekurzivne funkcije arnosti k i g parcijalno rekurzivna funkcija arnosti m za koje postoje odgovarajuće Tjuringove mašine. Opisaćemo Tjuringovu mašinu M_{komp} koja izračunava funkciju koja odgovara kompoziciji $g(h_1\ (x_1, \dots, x_k), \dots, h_m(x_1, \dots, x_k))$. Mašina M_{komp} će biti dobijena komponovanjem mašina M_g, M_{h_1}, ..., M_{h_m} koje izračunavaju funkcije g, h_1, ..., h_m. Na početku izvršavanja mašine M_{komp} traka će sadržati unarni zapis brojeva x_1, $\dots x_k$. U prvoj fazi rada mašina pravi kopiju ulaznih podataka i na njihovo mesto smešta rezultat izvršavanja mašine M_{h_1} koja izračunava vrednost $h_1(x_1, \dots, x_k)$. Postupak se ponavlja m puta, nakon čega traka sadrži m unarnih zapisa rezultata mašina M_{h_1}, ..., M_{h_m}. Sada se na tu traku primeni program Tjuringove mašine M_g, nakon čega mašina M_{komp} završava rad.

Primetimo da, ako za neko i funkcija $h_i(x_1, \dots, x_k)$ nije definisana, odgovarajuća Tjuringova mašina M_{h_i} se neće zaustaviti, pa će se isto dogoditi i mašini M_{komp}. Slično važi i u preostala dva slučaja čija analiza sledi, pa ovu situaciju nećemo posebno pominjati.

Dalje ćemo opisati Tjuringovu mašinu M_{pr} koja za date parcijalno rekurzivne funkcije g arnosti m i h arnosti $m+2$, tj. odgovarajuće Tjuringove mašine M_g i M_h, izračunava funkciju f definisanu primitivnom rekurzijom nad h sa bazom g. Na početku izvršavanja mašine M_{pr} na traci su smešteni unarni zapisi argumenata n i x_1, ..., x_k funkcije f. U jednoj petlji mašina M_{pr} proverava da li je argument n po kome se radi primitivna rekurzija jednak 0. Ako nije, pomera udesno sadržaj trake i na oslobođeno mesto smešta argumente rekurzivnog poziva, tako da nakon završetka petlje, kada argument po kome se radi rekurzija ima vrednost 0, izgled trake

$$101^{x_1+1}0 \dots 01^{x_k+1}01101^{x_1+1}0 \dots 01^{x_k+1}0 \dots 01^{n+1}01^{x_1+1}0 \dots 01^{x_k+1}$$

podseća na stek na kome se nalaze argumenti za pozivanje funkcija. Sada se redom koriste najpre Tjuringova mašina M_g i odgovarajući broj puta Tjuringova mašina M_h, nakon čega se prekida izvršavanje mašine M_{pr}.

Konačno, neka je data parcijalno rekurzivna funkcije f arnosti $k+1$, tj. odgovarajuća Tjuringova mašina M_f. Opisaćemo Tjuringovu mašinu M_μ koja izračunava funkciju $(\mu y)(f(x_1, \ldots, x_k, y) = 0)$ definisanu neograničenom minimizacijom u odnosu na funkciju f. Na početku izvršavanja mašine M_μ na traci su smešteni unarni zapisi argumenata $x_1, \ldots, x_k, 0$ funkcije f. Ovaj sadržaj se kopira na desno od krajnje desne ćelije koja sadrži znak 1 i nad tom kopijom se izvrši izračunavanje primenom mašine M_f. Ako je dobijeni rezultat unarni zapis broja 0, brišu se svi zapisi brojeva $x_1, \ldots, x_k$ i kao rezultat prikazuje zapis argumenta po kome se vrši minimizacija. Ako rezultat izvršavanja mašine M_f nije nula, on se briše, uvećava se vrednost argumenta po kome se vrši minimizacija i postupak ponavlja. ■

Teorema 2.5.2 Svaka Tjuring-izračunljiva funkcija je parcijalno rekurzivna.

Dokaz. Neka je f funkcija koju izračunava Tjuringova mašina M_f. Pretpostavimo da je f unarna funkcija, a ako nije posmatrajmo Tjuringovu mašinu koja kao ulazni podatak dobija Gedelov broj niza ulaznih podataka koji najpre raspakuje čime dobija sve ulazne podatke i zatim primenjuje program za M_f. U dokazu teoreme izvršićemo kodiranje programa i konfiguracija Tjuringove mašine M_f i opisati parcijalno rekurzivne funkcije koje manipulišući tim kodovima simuliraju izvršavanje mašine M_f.

Najpre ćemo svim stanjima i operacijama pridružiti kod, odnosno prirodan broj. Na primer, neka kod stanja q_z bude 0, a za sva ostala stanja q_i neka je kod $i+1$. Operaciju upisivanja 0, odnosno 1, kodirajmo sa 0, odnosno 1, operaciju pomeranja glava ulevo L sa 2, a operaciju pomeranja glave udesno R sa 3. Program mašine M_f se može opisati funkcijom oblika $h(x) = a$, gde je x kod para (q_i, s), a a kod para (o, q_j), za stanja q_i i q_j, znak s i operaciju o. Kako je program konačni niz naredbi, funkcija h se definiše po slučajevima, pa je primitivno rekurzivna. Sa $s(x,t)$ označićemo funkciju čija vrednost je 1 ako se nakon t koraka glava mašine M_f koja je započela izvršavanje sa ulaznim podatkom x nalazi iznad ćelije u kojoj je upisan znak 1, inače je $s(x,t) = 0$. Sa $l(x,t)$ označićemo Gedelov broj niza znaka koji se nakon t koraka izvršavanja mašine M_f koja je započela rad sa ulaznim podatkom x nalaze levo od aktuelne ćelije zaključno sa najlevljom ćelijom u kojoj se nalazi znak 1. Analogno, neka je $d(x,t)$ Gedelov broj niza znaka sa desne strane aktuelne ćelije nakon t koraka izvršavanja mašine M_f koja je započela rad sa ulaznim podatkom x, zaključno sa krajnjom desnom ćelijom koja sadrži znak 1. Sa $q(x,t)$ označimo broj tekućeg stanja mašine M_f koja je započela izvršavanje sa ulaznim podatkom x. Konfiguracija mašine M_f koja je započela izvršavanje sa ulaznim podatkom x nakon t koraka rada je jednoznačno opisana vrednostima $l(x,t)$, $s(x,t)$, $d(x,t)$ i $q(x,t)$, što je ilustrovano slikom 2.5. Funkcije $l(x,t)$, $s(x,t)$, $d(x,t)$ i $q(x,t)$ se definišu induktivno. Za $t = 0$, što odgovara početku izvršavanja mašine, ako su na traku smeštene $x+1$ jedinica koje čine unarni zapis ulaznog podatka x, biće: $q(x,0) = 1$, $l(x,0) = 0$, $s(x,0) = 1$ i $d(x,0) = gb(1^x)$. Neka je $t > 0$, vrednost $h(gb(q(x,t-1), s(x,t-1)) = a$ i $a = gb(o, q(x,t))$ kod para koji čine operacija i naredno stanje, onda je $q(x,t) = (a)_1$. Zavisno od vrednosti $o = (a)_0$, određuju se vrednosti za $l(x,t)$, $s(x,t)$ i $d(x,t)$. Na primer, ako kod o

$$\begin{array}{lll} l(x,t) & s(x,t) & d(x,t) \\ & q(x,t) & \end{array}$$

Slika 2.5. Funkcije s, l, d i q opisuju konfiguraciju Tjuringove mašine.

operacije odgovara upisu znaka 1, biće $l(x,t) = l(x,t-1)$, $s(x,t) = 1$ i $d(x,t) = d(x,t-1)$. Slično se postupa i u ostalim slučajevima. Pošto je funkcija h primitivno rekurzivna, a funkcije l, s, d i q se na osnovu nje definišu po slučajevima i one su primitivno rekurzivne.

Tjuringova mašina M_f se zaustavlja nakon t koraka u standardnoj konfiguraciji ako i samo ako je ispunjen uslov SK da je $(\forall i < t)(q(x,i) \neq 0)$, $q(x,t) = 0$, $l(x,t) = 0$, $s(x,t) = 1$ i $(\forall i < d(x,0)+t)(\neg \dot{-}(p(i)^2, d(x,t)) \rightarrow \neg \dot{-}(p(i+1)^2, d(x,t)))$, gde je $p(i)$ i-ti prosti broj. Uslov SK je primitivno rekurzivan. Pojasnimo poslednji deo uslova SK. Nakon t koraka izvršavanja broj korištenih ćelija je manji do jednak od $d(x,0) + t$. Ako se mašina nađe u standardnoj konfiguraciji znači da na traci postoji tačno jedan neprekidni blok znakova 1, odnosno kada se pojavi prvi znak 0, onda se desno od njega nalaze samo znaci 0. U kodu koji opisuje zauzete ćelije prisustvo znaka 1 u ćeliji i se prikazuje prisustvom faktora $p(i)^2$ u proizvodu, odnosno time što je vrednost koda deljiva sa $p(i)^2$. Prema tome, ako se u standardnoj konfiguraciji pojavi ćelija i koja sadrži znak 0, tj. $p(i)^2$ ne deli $d(x,t)$, isto će važiti i za sve ostale ćelije $i+1$, $i+2$, ...

Ako se mašina M_f zaustavi u standardnoj konfiguraciji, rezultat y je predstavljen rečju unarnog alfabeta 1^{y+1}, odnosno blokom neprekidnih jedinica dužine $duzina(d(x,t))$. Prema tome biće $f(x) = duzina(d(x,(\mu t)SK))$. ■

Posledica ovih teorema je da se tvrđenje o postojanju univerzalne Tjuringove mašine može dokazati indirektno, kao što je spomenuto u odeljku 2.3.9. Naime, Tjuringovu mašinu koja interpretira univerzalnu parcijalno rekurzivnu funkciju f_U označimo sa UTM. Izvršavanje proizvoljne Tjuringove mašine možemo simulirati pomoću neke parcijalno rekurzivne funkcije, koju opet simuliramo pomoću f_U, a zatim mašinom UTM. Analogno se, na osnovu postojanja mašine UTM i prethodnih teorema pokazuje postojanje univerzalne parcijalno rekurzivne funkcije f_U.

2.6 Čerčova teza

Čuvena Čerčova teza je iskaz da

svaki algoritam definiše funkciju koja
pripada jednoj dobro definisanoj klasi funkcija

(klasa parcijalno rekurzivnih funkcija, klasa Tjuring-izračunljivih funkcija, klasa λ-definabilnih funkcija ili neka druga ekvivalentna klasa), odnosno da se klasa intuitivno izračunljivih funkcija poklapa sa svakom od nabrojanih klasa. Iako je više istraživača skoro u isto vreme imalo slične ideje, tezu je prvi formulisao Čerč, pa je po njemu i dobila ime.

Intuitivni pojam algoritma je zasnovan na iskustvenom znanju o ljudskim umnim sposobnostima, dok su klase izračunljivih funkcija precizno definisane u odgovarajućim formalnim modelima izračunavanja. Čerčova teza izjednačava neformalni i formalni pristup pojmu efektivne izračunljivosti, te se ne može u strogom smislu smatrati matematičkim tvrđenjem, već je sličnija formulacijama raznih fizičkih zakona. Teza se ne može dokazati u okviru neke formalne teorije, ali može biti opovrgnuta ako bi bila pronađena funkcija koja jeste intuitivno izračunljiva, a nije, recimo, Tjuring-izračunljiva. Činjenica da se tako nešto nije dogodilo od njenog formulisanja govori u prilog tezi. Drugi važan argument u korist teze je međusobna ekvivalentnost raznorodnih formalnih modela izračunavanja do koje teško da bi došlo da neka od intuitivnih karakteristika algoritama nije njima obuhvaćena. Zbog svega toga, pošto višedecenijsko istraživanje nije uspelo da je obori, Čerčova teza se može prihvatiti i kao definicija izračunljivosti.

Tokom svih tih decenija razvijane su moćne tehnike za dokazivanje međusobne ekvivalentnosti formalnih modela izračunavanja i proučavanje široke klase intuitivno izračunljivih funkcija. Nagomilano iskustvo omogućava relativno lako prepoznavanje da li nekom neformalno opisanom postupku odgovara parcijalno izračunljiva funkcija i rutinski prelazak sa intuitivnog na strogi opis algoritma. To za posledicu ima primenu Čerčove teze u nešto širem smislu nego što je prethodno naznačeno. Naime, da bi se u raznim dokazima istakle suštinske ideje i izbegli tehnički detalji često se pribegava formulaciji oblika: 'funkcija je intuitivno izračunljiva, pa je prema Čerčovoj tezi parcijalno izračunljiva'. Time se dokazi skraćuju i čine preglednijim, no treba skrenuti pažnju da ovakvo pozivanje na Čerčovu tezu ne znači suštinski gubitak strogosti jer za svaki takav korak mora postojati formalno opravdanje koje se i iznosi u slučaju potrebe.

Čerčova teza se koristi i kao argument pri objašnjavanju zašto neki problem nije rešiv. Naime, ako pokažemo da se postupak za rešavanje problema ne nalazi u nekoj od formalizovanih klasa izračunljivih funkcija, na osnovu Čerčove teze zaključujemo i da ne postoji efektivni postupak za rešavanje tog problema. U poglavlju 2.8 će ovaj tip obrazloženja biti osnova za razdvajanje problema na one na koje smo u stanju da odgovorimo i one kod kojih to nije slučaj, tako da često neće ni biti eksplicitno istaknut.

Za kraj ovog odeljka spomenimo i jedan aspekt intuitivne izračunljivosti kome do sada nije bila posvećena pažnja. Naime, razmotrimo i šta se intuitivno podrazumeva pod algoritamskom izračunljivošću. Kao što će se videti u glavi 10, postoje rekurzivne funkcije za čije izračunavanje je potrebno vreme duže od vremena proteklog od pretpostavljenog nastanka kosmosa, i/ili se zahteva veći broj memorijskih registara nego što je broj atoma od kojih je sastavljena naša planeta. Postavlja se pitanje da li su takve funkcije zaista izračunljive, jer je očigledno da se bar neke njihove vrednosti praktično ne mogu izračunati. Ako se pod intuitivnom izračunljivošću podrazumeva ono što se stvarno može izračunati, uglavnom se prihvata da su izračunljive funkcije za čije izračunavanje je potrebno ne više od broja koraka koji je polinomijalna funkcija dužine ulaznih podataka, što je očigledno prava potklasa klase rekurzivnih funkcija. U tom slučaju, Čerčova teza predstavlja korisnu granicu klase funkcija izvan koje sigurno nema izračunljivih funkcija.

2.7 Relativna izračunljivost

Klasa funkcija koje nisu parcijalno rekurzivne je daleko veća od klase parcijalno rekurzivnih funkcije pošto ovih drugih ima samo prebrojivo mnogo dok svih aritmetičkih funkcija ima neprebrojivo mnogo. *Relativna izračunljivost* predstavlja odnos između dvaju funkcija, recimo f i g, kojim izražavamo da, recimo, funkciju f možemo izračunati pomoću osnovnih funkcija i funkcije g primenom kompozicije, primitivne rekurzije i minimizacije. Tada kažemo da je f relativno izračunljiva u odnosu na g. Jasno je da, ako funkcije g jeste parcijalno rekurzivna to isto važi i za funkciju f, dok to prestaje da važi ako g nije parcijalno rekurzivna. Na primer, u teoremi 2.4.33 je pokazano da unarna funkcija $g(x)$ čija vrednost je 1 ukoliko $f_x(x) \downarrow$, a inače 0, nije parcijalno rekurzivna, pa nisu parcijalno rekurzivne ni funkcije koje se dobijaju korištenjem ove funkcije.

Neka je g proizvoljna totalna funkcija. Funkcije izračunljive pomoću *orakla*[27] za funkciju g su funkcije koje se relativno izračunavaju u odnosu na g, odnosno formiraju klasu koja kao osnovnu funkciju ima i funkciju g. Analogno se definišu i programi za neku mašinu sa oraklom za funkciju g koji kao deo programa imaju naredbu koja izračunava funkciju g. Ako je, recimo, funkcija f izračunljiva programom P sa oraklom za funkciju g, onda je u nekom smislu f lakša za izračunavanje od g, jer znajući da izračunamo g to umemo da uradimo i za f. Opisivanje klase funkcija koju dobijamo uključujući i neke posebne neizračunljive funkcije među osnovne funkcije se sprovodi analogno postupku za opis parcijalno rekurzivnih funkcija. Oznaku $f \leq g$ koristimo da kažemo da je funkcija f izračunljiva pomoću orakla za g.

Primer 2.7.1 Neka je $g(x)$ funkcija čija vrednost je 1 ukoliko $f_x(x) \downarrow$, a inače 0. U teoremi 2.4.33 je pokazano da ona nije parcijalno rekurzivna. Neka je C_g klasa funkcija koja pored parcijalno rekurzivnih sadrži i funkcije koje su relativno izračunljive u odnosu na g. Jasno je da klasa C sadrži i funkcije koje nisu parcijalno rekurzivne, jer je $g \leq g$. Ipak, postoje i funkcije koje nisu u klasi C pošto je C prebrojivo beskonačna, dok svih aritmetičkih funkcija ima neprebrojivo mnogo. Neka je $f_1, f_2, \ldots$ jedno nabrajanje funkcija klase C. Analogno postupku iz teoreme 2.4.33 se pokazuje da funkcija h definisana sa

$$h(x) = \begin{cases} 1 & \text{ako } f_x(x) \downarrow \\ 0 & \text{inače} \end{cases}$$

ne pripada klasi C. ■

Klasa funkcija C sa osobinom da za svake dve funkcije $f, g \in C$ važi da je $f \leq g$ i $g \leq f$ naziva se *Tjuringov stepen*. Tjuringovi stepeni čine jednu hijerarhiju klasa funkcija koja se intenzivno proučava.

2.8 Odlučivost

U odeljku 2.2 je kao razlog uvođenja formalnih modela izračunavanja navedeno utvrđivanje da li za neki problem postoji algoritam koji ga rešava. Pošto defini-

[27] Orakl je rupa koja se nalazila u proročištu u Delfima u antičkoj Grčkoj iz koje su dobijani čudotvorni odgovori.

cije Tjuring-izračunljivih i parcijalno rekurzivnih funkcija određuju jasnu granicu dosega algoritama u nastavku ovog poglavlja analiziraćemo više problema i videti da za neke od njih u opštem slučaju ne postoji rešenje. U problemima koje ćemo sretati se uglavnom traži izračunavanje posebne vrste funkcija koje uvek imaju vrednost ili 0 ili 1, što se može tumačiti i kao *ne* i *da*. U ovakvim slučajevima problemi se obično formulišu sa 'da li je ...' ili 'da li važi ...' i sl. Ako smo u stanju da na takvo pitanje uvek odgovorimo problem je rešiv. Jasno je da probleme možemo shvatiti kao predikate, pa se u terminologiji koju koristimo za rešive probleme kaže da su *odlučivi*.

2.8.1 Odlučivi i neodlučivi predikati i skupovi

Kao što je u odeljku 2.4.4 rečeno, pod predikatima se podrazumevaju relacije nad $\mathbb{N}^k$ za $k = 1, 2, \ldots$, dok je karakteristična funkcija predikata totalna funkcija koja za k-torku argumenata ima vrednost 1 ako za tu k-torku predikat važi, inače je vrednost funkcije 0. U definiciji 2.4.30 predikat je nazvan *rekurzivnim* ili *odlučivim* ako je njegova karakteristična funkcija rekurzivna, tj. totalna parcijalno rekurzivna funkcija. Ako taj uslov za karakteričnu funkciju nije ispunjen, predikat je *neodlučiv*. Imajući u vidu dokazanu ekvivalentnost klasa parcijalno rekurzivnih funkcija i Tjuring-izračunljivih funkcija, lako se uočava da je neki predikat odlučiv ako i samo ako postoji program koji izračunava karakterističnu funkciju predikata, tj. definisan je za svaki ulaz odgovarajuće arnosti i daje rezultat 1 (ili završava rad u stanju q_{da}) ako predikat važi za ulazne podatke, a u suprotnom slučaju daje rezultat 0 (ili završava rad u stanju q_{ne}). Pošto je relacija nekakav podskup skupa nad kojim je definisana, prirodno je da se razmatra i odlučivost skupova. Zapravo, analogno se kaže za neki skup A da je rekurzivan (odlučiv), odnosno neodlučiv, ako mu je karakteristična funkcija ($C_A(x) = 1$ ako je $x \in A$, inače je $C_A(x) = 0$) rekurzivna, odnosno nije rekurzivna. U nastavku ćemo ravnopravno upotrebljavati reč skup i predikat u razmatranju (ne)odlučivosti.

Primeri odlučivih skupova su skup $\mathbb{N}$, svaki njegov konačan podskup[28], skup parnih i skup neparnih brojeva itd. Klasa odlučivih skupova (a time i predikata) je zatvorena za osnovne operacije komplementiranja (u odnosu na skup $\mathbb{N}^k$), unije, preseka i razlike, što trivijalno proizilazi iz razmatranja karakterističnih funkcija za skupove dobijene tim operacijama.

U teoremi 2.4.33 susreli smo se sa jednim neodlučivim problemom – pokazano je da problem utvrđivanja da li je $f_x(x) \downarrow$, tj. da li je unarna parcijalno rekurzivna funkcija sa indeksom x definisana za argument x nije odlučiv. Znači da predikat P definisan tako da $P(x)$ važi ako i samo ako je $f_x(x) \downarrow$ nije odlučiv. Sledi još nekoliko primera neodlučivih predikata. Do kraja ovog odeljka koristićemo oznake $Dom(f)$ za domen funkcije f, $Im(f)$ za kodomen funkcije f, f_x za funkciju sa indeksom x u nekom nabrajanju parcijalno rekurzivnih funkcija i $\mathbb{C}$ za komplement nekog skupa, odnosno relacije, u odnosu na $\mathbb{N}^k$ za neko $k > 0$.

Primer 2.8.1 Neka je C_{Tot} karakteristična funkcija unarnog predikata $Tot(x)$ koji važi ako i samo ako je f_x totalna funkcija. Koristeći metodu dijagonalizacije

[28] Za konačan skup $A = \{a_1, \ldots, a_n\}$ prirodnih brojeva karakteristična funkcija $x = a_1 \vee \ldots \vee x = a_n$ je očigledno rekurzivna.

pokazaćemo da predikat Tot nije odlučiv. Posmatrajmo najpre funkciju h definisanu sa:

$$h(0) = (\mu y)(C_{tot}(y) = 1)$$

$$h(n+1) = (\mu y)(y > h(n) \wedge C_{Tot}(y) = 1)$$

Ako bi funkcija C_{Tot} bila rekurzivna, isto bi važilo i za funkciju h, jer totalnih funkcija ima neograničeno mnogo, recimo sve konstantne funkcije su totalne. $Im(h)$ čine svi kodovi totalnih funkcija. Neka je funkcija f definisana sa:

$$f(x) = f_{h(x)}(x) + 1$$

Ako je h rekurzivna, onda je i funkcija f rekurzivna, znači i totalna, pa postoji njen indeks e takav da za neko $n_0 \in \mathbb{N}$ važi $h(n_0) = e$. Tada bi važilo $f(n_0) = f_{h(n_0)}(n_0) + 1 = f(n_0) + 1$, što je očigledna kontradikcija, pa funkcija C_{tot} nije rekurzivna, tj. predikat Tot nije odlučiv. Dakle, problem 'da li je funkcija totalna' nije odlučiv. ■

Primer 2.8.2 Neka je C_{Halt} karakteristična funkcija binarnog predikata $Halt$ (x, y) koji važi ako i samo ako je $f_x(y)$ definisano[29] i predstavlja uopštenje problema da li $f_x(x) \downarrow$, tj. da li je funkcija f_x definisana za argument x za koji je u teoremi 2.4.33 pokazano da nije odlučiv i što ćemo neposredno iskoristiti. Ako bi funkcija $C_{Halt}(x, y)$ bila rekurzivna, to bi bila i funkcija $f(x) = C_{Halt}(x, x)$. Međutim, funkcija $f(x)$ je ustvari karakteristična funkcija problema $f_x(x) \downarrow$ i nije rekurzivna, pa to nije ni funkcija C_{Halt}. Dakle, predikat $Halt$, odnosno problem 'da li je funkcija definisana', nije odlučiv. ■

Na osnovu dokazane ekvivalentnosti klasa Tjuring-izračunljivih i parcijalno rekurzivnih funkcija činjenica da predikat $Halt$ nije odlučiv se može protumačiti i na sledeći način: u opštem slučaju ne možemo odgovoriti da li neki program za dati ulaz završava rad u konačnom broju koraka[30].

Primer 2.8.3 Neka je $C_{\mathbb{O}}$ karakteristična funkcija predikata $\mathbb{O}(x)$ koji važi ako i samo ako je f_x nula funkcija, tj. za sve argumente ima vrednost nula. Pretpostavimo da je $C_{\mathbb{O}}$ rekurzivna funkcija. Napravićemo takvu funkciju za koju će se pitanje da li je jednaka nula-funkciji svesti na jedan drugi neodlučivi problem. Nula funkciju ćemo obeležiti sa $f\mathbb{O}$. Posmatrajmo funkciju f definisanu sa:

$$f(x, y) = f\mathbb{O}(f_x(y))$$

Funkcija f je parcijalno rekurzivna, pa postoji indeks e u nekom nabrajanju parcijalno rekurzivnih funkcija tako da je $f \simeq f_e$. Po s-m-n-teoremi (teorema 2.4.36), biće $f(x, y) \simeq f_{S_1^1(e,x)}(y)$. Lako se vidi da je $f_{S_1^1(e,x)}$ nula funkcija, odnosno da je $C_{\mathbb{O}}(S_1^1(e, x)) = 1$ ako i samo ako je $C_{Halt}(x, y) = 1$, tj. da bi iz odlučivosti predikata $\mathbb{O}$ sledila odlučivost predikata $Halt$. Pošto predikat $Halt$ nije odlučiv, isto važi i za predikat $\mathbb{O}$, pa problem 'da li je funkcija nula-funkcija', nije odlučiv. ■

[29] Ovaj problem se naziva i *halting*-problem, odnosno *problem zaustavljanja*.

[30] Kao što je rečeno u poglavlju 1, Tjuring je neodlučivost ovog problema iskoristio za davanje negativnog odgovora na Hilbertov Entscheidungsproblem.

Primer 2.8.4 Problem jednakosti dve parcijalno rekurzivne funkcije, odnosno da li je $f_x \simeq f_y$, nije odlučiv. Ovo je direktna posledica prethodnog primera. Pošto je nula-funkcija rekurzivna, a time i parcijalno rekurzivna, ona ima svoj indeks e, pa ako bi problem $f_x \simeq f_y$ bio odlučiv, to bi bio i problem $f_x \simeq f_e$, što nije slučaj. ■

Teorema 2.8.5 (Rajsova teorema) Neka je $\mathbb{B}$ neprazna prava potklasa klase svih parcijalno rekurzivnih funkcija. Problem da li je $f_x \in \mathbb{B}$ nije odlučiv.

Dokaz. Neka je h unarna funkcija koja nije definisana ni za jedan $n \in \mathbb{N}$. Pretpostavimo da $h \notin \mathbb{B}$ (u suprotnom, ako $h \in \mathbb{B}$ posmatraćemo komplement klase $\mathbb{B}$, $\mathbb{C}\mathbb{B}$, koji je odlučiv ako i samo ako je odlučiva klasa $\mathbb{B}$). Neka je $g \in \mathbb{B}$ funkcija različita od h. Takva funkcija uvek postoji pošto je $\mathbb{B} \neq \emptyset$, a pretpostavili smo da $h \notin \mathbb{B}$. Takođe, ako je klasa $\mathbb{B}$ odlučiva, takvu funkciju uvek možemo pronaći proveravajući redom da li se neka od funkcija iz bilo kog nabrajanja svih parcijalno rekurzivnih funkcija nalazi u $\mathbb{B}$. Zbog osobina klase $\mathbb{B}$, taj postupak mora, pre ili posle, dovesti do tražene funkcije g. Neka je funkcija $K_1^1(x)$ totalna konstanta funkcija koja uvek ima vrednost 1. Posmatrajmo funkciju $f(x, y)$ definisanu sa $g(y) \cdot K_1^1(f_x(x))$ koja je parcijalno rekurzivna. Znači da postoji njen indeks e u nekom nabrajanju parcijalno rekurzivnih funkcija. Po s-m-n-teoremi (teorema 2.4.36) biće $f(x, y) \simeq f_{S_1^1(e,x)}(y)$, pri čemu je funkcija S_1^1 rekurzivna. Kako je e konstanta, zapravo je $f(x, y) \simeq f_{S_1^1(x)}(y)$. Ako je $x \in Dom(f_x)$ biće $f_{S_1^1(x)} \simeq g \in \mathbb{B}$, a ako $x \notin Dom(f_x)$ biće $f_{S_1^1(x)} \simeq h \notin \mathbb{B}$. Prema tome $f_{S_1^1(x)} \in \mathbb{B}$ važi akko $x \in Dom(f_x)$. Poslednje pitanje, kao što je ranije pokazano, nije odlučivo, pa to ne može biti ni problem pripadanja funkcije klasi $\mathbb{B}$. ■

Teorema 2.8.5 zapravo kaže da su svi netrivijalni skupovi parcijalno rekurzivnih funkcija neodlučivi. Na primer:

Teorema 2.8.6 Sledeći problemi nisu odlučivi:

- domen funkcije je konačan,
- domen funkcije je beskonačan,
- kodomen funkcije je konačan i
- kodomen funkcije je beskonačan.

Dokaz. Tvrđenja su direktna posledica teoreme 2.8.5 pošto je reč o klasama funkcija koje nisu ni prazne niti se poklapaju sa klasom svih parcijalno rekurzivnih funkcija. ■

Među neodlučive spadaju i sledeći poznati problemi:

- problem reči za grupe, tj. ako je grupa G sa jediničnim elementom e generisana skupom elemenata $Gen_G = \{g_1, g_2, \ldots\}$, da li za proizvoljan izraz t_1 sastavljan od elemenata iz Gen_G (recimo $t_1 = g_2^3 g_1^{-1} g_5$) važi $t_1 = e$,
- rešivost diofantskih jednačina,
- problemi zadovoljivosti i valjanosti formula predikatskog računa prvog reda,

- problem pokrivanja[31] ravni u kome je dat konačan broj proizvoljnih oblika poligona, a postavlja se pitanje da li je moguće u potpunosti, bez preklapanja, pokriti ravan poligonima samo tih oblika itd.

dok su problemi zadovoljivosti i valjanost iskaznih formula odlučivi. Za neku teoriju T kažemo da je odlučiva ako postoji rekurzivna karakteristična funkcija problema 'formula α je teorema teorije T'. Primeri odlučivih teorija su pored iskaznog računa i teorija Bulovih algebri, teorija množenja prirodnih brojeva, teorija Abelovih grupa, teorija realno zatvorenih polja, elementarna euklidska geometrija itd. Sa druge strane, pored predikatskog računa, neodlučive su i teorije Peanova aritmetika, teorija grupa, teorija prstena, teorija polja, ZF teorija skupova itd.

2.8.2 Klasifikacija neodlučivih predikata i skupova

U odeljku 2.8.1 je prikazano više primera neodlučivih predikata i skupova. Zapravo se može reći da je neodlučivost pravilo, a odlučivost izuzetak imajući u vidu brojnost jednih i drugih slučajeva. U ovom odeljku ćemo razmotriti jednu klasifikaciju neodlučivih predikata.

Parcijalno odlučivi predikati i skupovi

Definicija 2.8.7 Predikat R je *parcijalno odlučiv* ako je njegova karakteristična funkcija parcijalno rekurzivna, tj. oblika[32]

$$C_R(x_1,\ldots,x_n) = \begin{cases} 1 & \text{ako važi } R(x_1,\ldots,x_n) \\ \text{nedefinisano} & \text{inače.} \end{cases}$$

Drugačiji naziv za parcijalno odlučive predikate je *rekurzivno nabrojivi predikati*. Analogno predikatima definišu se i parcijalno odlučivi, odnosno rekurzivno nabrojivi, skupovi i problemi. Kao i ranije, ove pojmove ćemo ravnopravno koristiti.

Teorema 2.8.8 Skup P je parcijalno odlučiv ako i samo ako postoji parcijalno rekurzivna funkcija f čiji je domen skup P.

Dokaz. Ako je P parcijalno odlučiv skup, funkcija f je njegova karakteristična funkcija. Obrnuto, ako postoji parcijalno rekurzivna funkcija f čiji domen je jednak P, onda je $C_P(x_1,\ldots,x_n) = K_1^1(f(x_1,\ldots,x_n))$ karakteristična funkcija za P. C_P je parcijalno rekurzivna funkcija, pa je po definiciji P parcijalno odlučiv. ■

Na osnovu definicije 2.8.7 je jasno da je svaki odlučiv predikat istovremeno i parcijalno odlučiv pošto je svaka rekurzivna funkcija istovremeno i parcijalno rekurzivna, dok obrnuto ne mora važiti. Sledeći primer prikazuje jedan takav predikat.

Primer 2.8.9 Predikat $Halt$ je, kao što je pokazano u primeru 2.8.3, neodlučiv. Kako je $C_{Halt}(x,y) = K_1^1(f_x(y))$ parcijalno rekurzivna funkcija, predikat $Halt$ je parcijalno odlučiv. ■

[31] Tiling problem.

[32] Ova definicija se može oslabiti tako što se dozvoli da za neke, ali ne nužno sve, $x_1,\ldots,x_n$ za koje $R(x_1,\ldots,x_n)$ ne važi, bude $C_R(x_1,\ldots,x_n) = 0$.

Teorema 2.8.10 Predikat $P(x_1, \ldots, x_n)$ je parcijalno odlučiv ako i samo ako postoji odlučiv predikat $R(x_1, \ldots, x_n, y)$ tako da važi $P(x_1, \ldots, x_n)$ ako i samo ako važi $(\exists y)R(x_1, \ldots, x_n, y)$.

Dokaz. Neka je R traženi predikat. Definišimo funkciju $C_P(x_1, \ldots, x_n)$ tako da bude jednaka $K_1^1((\mu y)(R(x_1, \ldots, x_n, y)))$. Funkcija C_P je očigledno parcijalno rekurzivna, pa kako predstavlja karakterističnu funkciju za P, sam predikat P je parcijalno odlučiv. Obrnuto, neka je predikat P parcijalno odlučiv i neka je C_P njegova parcijalno rekurzivna karakteristična funkcija. Neka je njen kod u nekom nabrajanju parcijalno rekurzivnih funkcija e. Posmatrajmo predikat $R(x_1, \ldots, x_n, y)$ definisan kao $UP(e, \mathrm{g}b(x_1, \ldots, x_n), (y)_0, (y)_1)$. UP je univerzalni predikat uveden u teoremi 2.4.34, gde je dokazano da je on primitivno rekurzivan, a time i odlučiv, pa je to i predikat R. Pri tome važi $P(x_1, \ldots, x_n)$ ako i samo ako $C_P(x_1, \ldots, x_n) \downarrow$ ako i samo ako $(\exists y)R(x_1, \ldots, x_n, y)$, što je i trebalo dokazati. ■

Očigledno je, dakle, da parcijalna odlučivost ne povlači i odlučivost predikata, ali ako su i predikat i njegov komplement parcijalno odlučivi, onda je i sam predikat odlučiv, što se dokazuje sledećom teoremom.

Teorema 2.8.11 Predikat P je odlučiv ako i samo ako su predikati P i $\mathbb{C}P$ parcijalno odlučivi.

Dokaz. Ako je predikat odlučiv, odlučiv je i njegov komplement, pa su oba i parcijalno odlučivi. Obrnuto, pretpostavimo da su P i $\mathbb{C}P$ parcijalno odlučivi predikati arnosti n. Prema teoremi 2.8.10 postoje odlučivi predikati $R(x_1, \ldots, x_n, y)$ i $S(x_1, \ldots, x_n, y)$ takvi da važi $P(x_1, \ldots, x_n)$ ako i samo ako $(\exists y)R(x_1, \ldots, x_n, y)$ i $\mathbb{C}P(x_1, \ldots, x_n)$ ako i samo ako $(\exists y)S(x_1, \ldots, x_n, y)$. Primetimo da za svaku fiksiranu n-torku $x_1, \ldots, x_n$ važi tačno jedan od predikata $(\exists y)R(x_1, \ldots, x_n, y)$, odnosno $(\exists y)S(x_1, \ldots, x_n, y)$. Uočimo funkciju $g(x_1, \ldots, x_n, y) = C_R(x_1, \ldots, x_n, y) + C_S(x_1, \ldots, x_n, y)$, gde su C_R i C_S rekurzivne karakteristične funkcije predikata R i S. Očigledno je da je funkcija g rekurzivna. S obzirom na vezu predikata R i S sa predikatima P i $\mathbb{C}P$, za svaku fiksiranu n-torku $x_1, \ldots, x_n$, jedna od funkcija C_R i C_S je nula-funkcija za svako y, dok ona druga za neke y ima takođe vrednost 0, ali i za neke y ima vrednost 1. Prema tome, kodomen $Im(g)$ funkcije g je skup $\{0, 1\}$. Posmatrajmo sada funkciju $f(x_1, \ldots, x_n) = (\mu y)(g(x_1, \ldots, x_n, y) = 1)$. Na osnovu prethodnog i funkcija f je rekurzivna, dok je za fiksirane $x_1, \ldots, x_n$ njena vrednost minimalno y za koje je ispunjeno da važi bilo $R(x_1, \ldots, x_n, y)$, bilo $S(x_1, \ldots, x_n, y)$. Konačno, definišimo funkciju:

$$C_P(x_1, \ldots, x_n) = \begin{cases} 1 & \text{ako } C_R(x_1, \ldots, x_n, f(x_1, \ldots, x_n)) = 1 \\ 0 & \text{inače.} \end{cases}$$

Ova funkcija je rekurzivna i ima vrednost 1 ako i samo ako $C_R(x_1, \ldots, x_n, f(x_1, \ldots, x_n)) = 1$ ako i samo ako važi $(\exists y)R(x_1, \ldots, x_n, y)$, odnosno ako i samo ako važi $P(x_1, \ldots, x_n)$, pa je to karakteristična funkcija predikata P. Odatle je predikat P odlučiv. ■

Intuitivno rečeno, predikat $P(x)$ je parcijalno odlučiv ako postoji program koji odgovara potvrdno u slučaju da predikat važi za argumente programa, inače program ne mora da se zaustavi. Ako bi postojali takvi programi $Prog_P$ za predikat

P i $Prog_{\mathbb{C}P}$ za njegov komplement $\mathbb{C}P$, mogli bismo na dva računara da ih pokrenemo paralelno. Pošto za svaki x važi ili $P(x)$ ili $\mathbb{C}P(x)$ jedan od programa će se posle izvesnog vremena zaustaviti. Ako se zaustavi program $Prog_P$ odgovor bi bio 'važi $P(x)$', a ako se zaustavi program $Prog_{\mathbb{C}P}$ odgovor bi bio 'ne važi $P(x)$', pa bi predikat P bio odlučiv.

Primer 2.8.12 Imajući u vidu dokazano u teoremi 2.4.33 predikat P takav da važi $P(x)$ ako i samo ako $x \in Dom(f_x)$ nije odlučiv. Kako je karakteristična funkcija predikata $C_P(x) = K_1^1(f_x(x))$ parcijalno rekurzivna, on je parcijalno odlučiv. Zbog toga i rezultata teoreme 2.8.11, njegov komplement $x \notin Dom(f_x)$ ne može biti parcijalno odlučiv, jer da jeste, predikat $x \in Dom(f_x)$ bi bio odlučiv. ■

Teorema 2.8.13 Neka je $A \subset \mathbb{N}$. Tada je ekvivalentno:

1. A je parcijalno odlučiv,

2. $A = \emptyset$ ili A je kodomen neke rekurzivne funkcije i

3. postoji indeks e i broj n $(e, n \in \mathbb{N})$ takvi da je A kodomen n-arne parcijalno rekurzivne funkcije f_e.

Dokaz. $(1 \Rightarrow 2)$ Prema teoremi 2.8.8 skup A je domen neke parcijalno rekurzivne funkcije f_e. Ako ta funkcija nije nigde definisana skup A je prazan. Pretpostavimo zato da je $A \neq \emptyset$. Tada postoji $a \in A$. Posmatrajmo predikat $T(e, x, t)$ definisan analogno univerzalnom predikatu UP iz teoreme 2.4.34 sem što ne vodi računa o rezultatu funkcije. Jasno je i da je predikat T primitivno rekurzivan. Definišimo funkciju $g(x, t) = x \cdot C_T(e, x, t) + a \cdot C_{\neg T}(e, x, t)$ koja je trivijalno parcijalno rekurzivna i totalna i ima vrednost x ako je $x \in A$, odnosno a ako $x \notin A$. Prema tome, $A = Im(g)$.
$(2 \Rightarrow 3)$ Trivijalno, jer je svaka rekurzivna funkcija i parcijalno rekurzivna.
$(3 \Rightarrow 1)$ Ako je $A = Im(f_e)$ za neku n-arnu parcijalno rekurzivnu funkciju f_e, karakteristična funkcija skupa A definisana sa $C_A(x) = K_1^1((\mu y) f_e((y)_0, \ldots, (y)_{n-1}) = x))$ je parcijalno rekurzivna, pa je i skup A parcijalno odlučiv. ■

Primer 2.8.14 Skup $Tot = \{x : f_x \text{ je totalna funkcija}\}$ nije parcijalno odlučiv. Ako bi to bio, prema teoremi 2.8.13, bio bi jednak i kodomenu neke rekurzivne funkcije h. Niz funkcija $f_{h(n)}$ sadržao bi sve rekurzivne funkcije, što bi dovelo do kontradikcije kao i u primeru 2.8.1. ■

Primer 2.8.15 Neka je $p(x_1, \ldots, x_m, y_1, \ldots, y_n)$ polinom sa celobrojnim koeficijentima i promenljivim $x_1, \ldots, x_m, y_1, \ldots, y_n$. Predikat P, takav da važi $P(x_1, \ldots, x_m)$ ako i samo ako je $(\exists y_1) \ldots (\exists y_n) p(x_1, \ldots, x_m, y_1, \ldots, y_n) = 0$, naziva se *diofantovski*. Jurij Matijaševič je pokazao da je svaki parcijalno odlučiv predikat ekvivalentan nekom diofantovskom predikatu odakle direktno sledi da problem rešivosti diofantovskih jednačina nije odlučiv. ■

Aritmetička hijerarhija

Kao što smo videli u primeru 2.8.14, postoje problemi koji ne samo da nisu odlučivi već nisu ni parcijalno odlučivi. Zapravo postoji čitava jedna hijerarhija složenosti skupova prirodnih brojeva koja se naziva *aritmetička hijerarhija*. Kao i ranije, smatraćemo da su skupovi i predikati jedno te isto. Klase odlučivih i parcijalno odlučivih skupova su na samom dnu ove hijerarhije, odnosno predstavljaju najjednostavnije klase. Skupovi koji pripadaju klasama iz aritmetičke hijerarhije nazivaju se *aritmetički skupovi*. Hijerarhija se definiše induktivno:

Definicija 2.8.16 Skup prirodnih brojeva $A \subset \mathbb{N}$ je u klasi Σ_1^0 ako i samo ako postoji odlučivi binarni predikat B takav da $x \in A$ ako i samo ako $(\exists y)B(x, y)$.

Skup prirodnih brojeva $A \subset \mathbb{N}$ je u klasi Π_1^0 ako i samo ako postoji Σ_1^0 skup A' takav da je $A = \mathbb{C}A'$.

Skup prirodnih brojeva $A \subset \mathbb{N}$ je u klasi Δ_1^0 ako i samo ako pripada klasi $\Sigma_1^0 \cap \Pi_1^0$.

Skup prirodnih brojeva $A \subset \mathbb{N}$ je u klasi Σ_{n+1}^0 ako i samo ako postoji Π_n^0 binarni predikat B takav da $x \in A$ ako i samo ako $(\exists y)B(x, y)$.

Skup prirodnih brojeva $A \subset \mathbb{N}$ je u klasi Π_{n+1}^0 ako i samo ako postoji Σ_{n+1}^0 skup A' takav da je $A = \mathbb{C}A'$.

Skup prirodnih brojeva $A \subset \mathbb{N}$ je u klasi Δ_{n+1}^0 ako i samo ako pripada klasi $\Sigma_{n+1}^0 \cap \Pi_{n+1}^0$.

Za podskup A skupa $\mathbb{N}^k$, za $k > 1$, odnosno predikat veće arnosti, smatraćemo da pripada onoj klasi kojoj pripada skup kodova nizova dužine k koje čine k-torke iz A. Primetimo da su, na osnovu teorema 2.8.10 i 2.8.11, klase Δ_1^0 i Σ_1^0 u stvari klase odlučivih i parcijalno odlučivih skupova. Skupovi iz klase Π_1^0 se nazivaju *koparcijalno odlučivi*, odnosno *korekurzivno nabrojivi*. .

Može se pokazati da je ova hijerarhija prava, tj. da za svako n važi: $\Delta_n^0 \subset \Sigma_n^0$, $\Delta_n^0 \neq \Sigma_n^0$, $\Delta_n^0 \subset \Pi_n^0$, $\Delta_n^0 \neq \Pi_n^0$, $\Pi_n^0 \neq \Sigma_n^0$ i $\Sigma_n^0, \Pi_n^0 \subset \Delta_{n+1}^0$. Recimo, u primeru 2.8.12 je pokazano da problem $x \in Dom(f_x)$ nije odlučiv, a jeste parcijalno odlučiv, dok njegov komplement $x \notin Dom(f_x)$ nije parcijalno odlučiv. Može se pokazati i da su sve ove klase zatvorene za konačne unije i preseke.

Konačno treba reći i da postoje skupovi prirodnih brojeva koji nisu u ovoj hijerarhiji.

2.9 Teorija izračunljivosti i programski jezici

Opisivanja izračunljivih funkcija u različitim formalizmima uticalo je i na razvoj programskih jezika. Navešćemo nekoliko primera za to.

Zapisivanje algoritamskih šema u vidu nizova naredbi je dovelo do uvođenja labela, tj. adresa, naredbi i naredbi uslovnog i bezuslovnog skoka[33] koje su tipične za nestruktuirane programske jezike. Primetimo takođe da naredbe Tjuringove mašine u sebi sadrže ideju programskog skoka u formi promene stanja. Sa druge strane, jezgro programskog jezika PASCAL je nastalo tako što je formalizam *while*-programa obogaćen proizvoljnim logičkim izrazima.

[33] *if S then go to A* i *go to A*.

Formulacija izračunljivih funkcija na bazi λ-računa, proširena numeričkim i funkcijama za rad sa rečima predstavlja osnovu za programski jezik LISP[34], iz koga su proizašli i mnogi drugi savremeni funkcionalni jezici. Za izvršavanje funkcionalnih programskih jezika su razvijene posebne vrste i idealnih[35] i stvarnih mašina koje se značajno razlikuju od standardnih računara jer dozvoljavaju direktnu i visoko paralelizovanu redukciju objekata koji predstavljaju funkcije.

Programski jezik SNOBOL je nastao iz ideje pravila kod Markovljevih algoritama kao jezik specijalizovan za operacije obrade reči. SNOBOL sadrži naredbe dodeljivanje, zamene podreči i uslovnog skoka.

Osnova programskog jezika PROLOG[36] se može shvatiti kao deduktivno zasnovan mehanizam rešavanja sistema jednačina kojima se definišu funkcije. Sistem zamišljamo kao da se sastoji od jednačina u formi Hornovih klauza $R_1 \wedge \ldots \wedge R_n \rightarrow Q$, gde su R_i i Q oblika $f(x_1, \ldots, x_m) = y$, što ne predstavlja ograničenje. Na primer, jednačina $f(x) = g(h(x))$ se može zapisati u obliku $h(x) = y \wedge g(y) = z \rightarrow f(x) = z$.

2.10 Drugi formalni modeli izračunavanja

U narednim odeljcim ćemo ukratko prikazati još nekoliko formalnih modela izračunavanja. Za sve njih je pokazano da su jednako izražajni.

2.10.1 Postova mašina i normalni sistemi

Postova mašina je sistem razvijen nezavisno i pre nego što je objavljen rad koji opisuje Tjuringovu mašinu, ali je publikovan nešto kasnije, pa je prosto iznenađujuća njihova sličnost. I kod Postove mašine postoji beskonačna traka podeljena na ćelije u koje može biti opisan blanko znak ili znak crtice, odnosno znaci 0 i 1, i glava koja se pomera, čita sadržaj ćelija i u njih upisuje jedan od znakova.

Post se bavio i istraživanjima takozvanih normalnih sistema, posmatrajući najpre pravila izvođenja oblika $\alpha_0 v_1 \alpha_1 v_2 \ldots v_n \alpha_n \rightarrow \beta_0 v'_1 \beta_1 v'_2 \ldots v'_m \beta_m$ gde su α_i, β_j reči nekog alfabeta i v_i, v'_j promenljive za reči. Pokazao je da je dovoljno koristiti samo produkcije oblika $\alpha v \rightarrow v\beta$ da bi se opisao svaki skup parova reči u unarnoj notaciji oblika $(1^n, 1^{f(n)})$ za sve prirodne brojeve n. Ovako definisane funkcije se nazivaju generisane i njihova klasa se poklapa sa klasom rekurzivnih funkcija.

2.10.2 λ-račun

λ-račun je formalni model izračunavanja koji je tridesetih godina dvadesetog veka razvio Čerč zajedno sa svojim studentima Klinijem i John-om Rosser-om (Roser, 1907 – 1989) pokušavajući da da predlog definicije efektivne izračunljivosti. Vremenom, λ-račun je postao osnova za implementiranje funkcionalnih programskih jezika. Jedine sintaksne celine u λ-računu su λ-izrazi koji mogu biti:

- konstante,

[34] Ime dolazi od prvih slova reči *List Processing*.
[35] SECD mašina, SK mašina.
[36] Ime dolazi od prvih slova reči *Programming in Logic*.

- imena promenljivih,
- aplikacije oblika $\lambda - \text{izraz}\ \ \lambda - \text{izraz}$ i
- apstrakcije oblika λ ime promenljive.λ-izraz

U izrazima se koriste i zagrade kako bi se povećala čitljivost.

λ-apstrakcijom se definišu funkcije, recimo $\lambda x.+\ x\ 1$ se shvata kao funkcija sa argumentom x u kojoj se vrednost argumenta uvećava za 1. Promenljiva koja neposredno sledi znak λ je formalni parametar funkcije, a λ-izraz iza tačke je telo funkcije. Ovakva definicija funkcija je slična sa onom u standardnim programskim jezicima, izuzimajući da je ovde funkcija neimenovana i da poseduje samo jedan argument. Funkcije sa više argumenata, recimo $f(x, y) = x+y$, uvode se uzastopnim apstrakcijama[37] oblika $\lambda x.(\lambda y.+\ y)\ x$.

Aplikaciji u programskim jezicima odgovara poziv funkcije, recimo $(\lambda x.+\ x\ 1)\ 4$. Ovaj izraz bi, u programerskom žargonu, vratio vrednost 5 pošto se vrednost argumenta funkcije fiksira da bude 4 i zatim se uveća za 1. Argument aplikacije sme biti i drugi λ-izraz.

U svom najčistijem obliku λ-račun ne sadrži ugrađene funkcije, poput sabiranja, ali se imajući u vidu primene, često proširuje aritmetičkim i logičkim funkcijama ($+$, $-$, $*$, :, *not*, *and*, *or*), konstantama (0, 1, ..., *true*, $false$, 'a', 'b', ..., NIL[38]), uslovnom $if - then - else$ funkcijom, funkcijama koje barataju sa listama i koje se definišu sa $HEAD(CONSab) = a$ i $TAIL(CONSab) = b$ itd. Sve ove funkcije i konstante su zapravo skraćeni zapisi odgovarajućih λ-izraza i koriste se kako bi se zapis učinio preglednijim i ne povećavaju izražajnost jezika. Recimo, definiše se:

$$CONS = (\lambda a.\lambda b.\lambda f.f\ a\ b)$$

$$HEAD = (\lambda c.c(\lambda a.\lambda b.a)) \text{ i}$$

$$TAIL = (\lambda c.c(\lambda a.\lambda b.b)).$$

Slično, ako definišemo $TRUE = (\lambda x.\lambda y.x)$ i $FALSE = (\lambda x.\lambda y.y)$, onda se uslovna funkcija definiše sa

$$if\ E_1\ then\ E_2\ else\ E_3 = (\lambda E_1.\lambda E_2.\lambda E_3.E_1\ E_2\ E_3).$$

Proširivanje jezika ovim elementima inspirisano je zahtevima za efikasnošću pošto se spominjane konstante i funkcije direktno reprezentuju u računaru.

U λ-računu su precizno definisane transformacije jednih u druge λ-izraze. Postupak transformacije odgovara izračunavanju vrednosti izraza. Pravila kojima se vrši transformacija iz jednog u drugi, ekvivalentni, λ-izraz nazivaju se *konverzije*.

Definicija 2.10.1 Pojava promenljive u izrazu može biti *slobodna* ili *vezana*. Promenljiva se javlja kao slobodna u sledećim situacijama:

- promenljiva x je slobodna u x,
- promenljiva x je slobodna u λ-izrazu (EF) ako je x slobodna u λ-izrazu E ili u λ-izrazu F i

[37] Carrying.
[38] Oznaka kraja liste.

- promenljiva x je slobodna u λ-izrazu $\lambda y.E$ ako su x i y različite promenljive pri čemu je x slobodna u λ-izrazu E.

Promenljiva se javlja kao vezana u sledećim situacijama:

- promenljiva x je vezana u λ-izrazu (EF) ako je x vezana u izrazu E ili u izrazu F i
- promenljiva x je vezana u λ-izrazu $\lambda y.E$ ako su x i y iste promenljive ili je x vezana u E.

Ni jedna promenljiva nije vezana u izrazu koji se sastoji od samo jedne promenljive, ili konstante. U λ-izrazu $(\lambda x.+\ x\ y)\ 4$ promenljiva x je vezana, a promenljiva y slobodna.

U narednoj definiciji koristićemo oznaku $E[M/x]$ koja znači da se u izrazu E slobodne pojave promenljive x zamenjuju sa M.

Definicija 2.10.2 *Konverzije* u λ-računu su:

- α-konverzija, ako y nije slobodno u E,

$$(\lambda x.E) \leftrightarrow_\alpha (\lambda y.E[y/x])$$

- β-konverzija,

$$(\lambda x.E)M \leftrightarrow_\beta E[M/x]$$

- η-konvrezija, ako x nije slobodno u λ-izrazu E koji označava funkciju,

$$(\lambda x.E\ x) \leftrightarrow_\eta E.$$

Kada se konverzije izvode sa leva na desno, β i η konverzije nazivaju se *redukcije* i označavaju sa $\rightarrow$.

Nije teško proveriti da α-konverzija predstavlja jednostavno preimenovanje formalnog parametra funkcije, na primer:

$$(\lambda x.+\ x\ 1) \leftrightarrow_\alpha (\lambda y.+\ y\ 1)$$

β-konverzijom, ili preciznije β-redukcijom, se vrši prenos parametara pri nekoj aplikaciji, odnosno pozivu funkcije. Efikasnost funkcionalnog jezika zavisi pre svega od kvaliteta realizacije β-redukcije. Rezultat ove transformacije primenjene na neku λ-aplikaciju je instancijacija tela funkcije, tj. λ-apstrakcije, koja učestvuje u aplikaciji, pri čemu se sve slobodne pojave formalnog parametra u telu apstrakcije zamenjuju argumentom aplikacije:

$$(\lambda x.+\ x\ 1)4 \rightarrow_\beta +\ 4\ 1$$

Slično, redukcija uslovne funkcije se izvodi na sledeći način:

$$if\ TRUE\ then\ M\ else\ N$$

$$= (\lambda p.\lambda q.\lambda r.p\ q\ r)TRUE\ M\ N$$

$$\rightarrow_\beta (\lambda q.\lambda r.TRUE\ q\ r)M\ N$$

$$\rightarrow_\beta (\lambda r.TRUE\ M\ r)N$$

$$\rightarrow_\beta TRUE\ M\ N$$

$$= (\lambda x.\lambda y.x)M\ N$$

$$\rightarrow_\beta (\lambda y.M)N$$

$$\rightarrow_\beta M$$

Prilikom primene β-redukcije postoji skrivena opasnost kolizije stvarnog parametra sa formalnim parametrom neke unutrašnje λ-apstrakcije. Na primer, neka je λ-izraz

$$E = (\lambda f.\lambda x.f\ (f\ x))$$

Posmatrajmo sada izraz $E\ E$ i pokušajmo sprovesti β-redukciju:

$$E\ E$$

$$= (\lambda f.\lambda x.f\ (f\ x))\ E$$

$$\rightarrow_\beta (\lambda x.E\ (E\ x))$$

$$= (\lambda x.E\ ((\lambda f.\underline{\lambda x.f\ (f\ x))}\ x))$$

U podvučenom delu izraza ne bi bilo korektno izvršiti zamenu formalnog parametra f sa x, pri čemu bi se dobilo

$$\rightarrow_\beta (\lambda x.E\ (\lambda x.x\ (x\ x)))$$

jer je x takođe formalni parametar unutrašnje λ-apstrakcije. U takvim situacijama se problem rešava preimenovanjem formalnog parametra, odnosno α-konverzijom

$$(\lambda x.E\ ((\lambda f.\underline{\lambda x.f\ (f\ x))}\ x))$$

$$\leftrightarrow_\alpha (\lambda x.E\ ((\lambda f.\underline{\lambda y.f\ (f\ y))}\ x))$$

$$\rightarrow_\beta (\lambda x.E\ (\lambda y.x\ (x\ y)))$$

U λ-računu je dopuštena i obrnuta transformacija koja se naziva β-apstrakcija i označava sa $\leftarrow_{beta}$.

η-konverzija izražava intuitivnu ekvivalentnost izraza, na primer

$$(\lambda x.+\ 1\ x) \leftrightarrow_\eta (+1)$$

dok nisu η-ekvivalentni izrazi $(\lambda x.(+\ x)\ x)$ i $(+\ x)$ jer je u prvom izrazu x slobodno u izrazu $(+\ x)$. Očigledno je da se pri η-redukciji eliminišu nepotrebne apstrakcije.

Radi potpunosti navedenim konverzijama treba dodati i δ-konverziju (δ-pravilo) koja predstavlja zajedničko ime za skup transformacija nad ugrađenim funkcijama i konstantama. Recimo,

$$+\ 4\ 1 \leftrightarrow_\delta 5$$

odgovara pravilu za redukciju operacije sabiranja itd.

Postupak koji se izvodi redukcijama i/ili konverzijama odgovara izračunavanju u nekom standardnom programskom jeziku. U preostalom delu ovog odeljka opisaćemo šta predstavlja rezultat takvog izračunavanja.

Definicija 2.10.3 *Redeks* je λ-izraz koji se može redukovati β-redukcijom. λ-izraz koji ne sadrži ni jedan redeks je u *normalnoj formi*.

Normalna forma izraza je oblik izraza do kojega pokušavamo da dovedemo svaki izraz. Normalna forma ne postoji za svaki izraz, recimo:

$$(\lambda x.x\ x)(\lambda x.x\ x) \rightarrow_\beta (\lambda x.x\ x)(\lambda x.x\ x)$$

dok kod nekih izraza redosled redukovanja redeksa utiče na (ne)dobijanje normalne forme. Na primer, izraz

$$(\lambda x.3)(\lambda x.xx)(\lambda x.xx)$$

bi se redukovao na 34, ako bi se prvo primenila redukcija na apstrakciju $(\lambda x.3)$ bez izračunavanja argumenata, a u suprotnom bismo upali u beskonačni ciklus redukcije.

Može se pokazati da neki redosledi redukovanja uvek dovode polazni izraz u normalnu formu, ukoliko normalna forma za taj izraz postoji. U vezi sa tim su naredne definicije i teoreme koje dajemo bez dokaza.

Definicija 2.10.4 *Normalni redosled redukcije* određuje da se uvek prvo redukuje najlevlji spoljašnji redeks. U *aplikativnom redosledu redukcije* se najpre redukuju argumenti aplikacije.

Teorema 2.10.5 (Prva Čerč – Roser-ova teorema) Ako za izraze E_1 i E_2 važi $E_1 \leftrightarrow E_2$ postoji izraz E takav da se oba izraza E_1 i E_2 mogu redukovati na njega ($E_1 \rightarrow E$ i $E_2 \rightarrow E$). Normalna forma je jedinstvena do na α-konverziju.

Teorema 2.10.6 (Druga Čerč – Roser-ova teorema) Ako za izraze E i F važi da je $E \rightarrow F$ i da je F u normalnoj formi, onda postoji niz redukcija iz E u F u normalnom redosledu.

Ove teoreme garantuju postojanje najviše jedne normalne forme i preciziraju redosled redukcija kojima se dolazi do normalne forme, ukoliko ona postoji. Normalni redosled je uz to veoma pogodan, jer je u nekim sistemima za redukciju i najkraći po dužini - broju pojedinačnih redukcija.

2.10.3 Markovljevi algoritmi

Sistem Markovljevih algoritama, nastao pedesetih godina dvadesetog veka, predstavlja vrstu produkcionih sistema. Neka je $A = \{1\}$ unarni alfabet i neka se blanko znak označava sa 0. *Algoritmom nad alfabetom A* nazivamo svaku efektivno izračunljivu funkciju čiji domen čini niz od fiksiranog broja reči nad A (razdvojenih znakom 0), a kodomen reči nad A. Algoritamska šema se sastoji od konačnog niza prostih pravila oblika $P \rightarrow Q$ i/ili završnih pravila oblika $P \rightarrow \cdot Q$, gde su P i Q reči nad A. Oznaku $P \rightarrow (\cdot)Q$ ćemo koristiti da označimo bilo koje od ove dve vrste pravila. Za datu reč P i algoritamsku šemu $\mathcal{A}$ oblika

$P_1 \to (\cdot)Q_1$

$\ldots$

$P_k \to (\cdot)Q_k$

koristimo oznaku $\mathcal{A} : P!$ ako ni jedna od reči P_i nije podreč reči P. Ako je m $(1 \leq m \leq k)$ najmanji broj takav da je P_m podreč reči P i ako je R reč dobijena iz P zamenom najlevlje pojave reči P_m rečju Q_m, onda pišemo $P \vdash_{\mathcal{A}} R$, ako je pravilo $P_m \to (\cdot)Q_m$ prosto, odnosno $P \vdash_{\mathcal{A}} \cdot R$, ako je pravilo završno. Sa $P \models_{\mathcal{A}} (\cdot)R$ označavamo da postoji niz reči $P = R_0, R_1, \ldots, R_n = R$ takva da je $R_i \vdash_{\mathrm{A}} R_{i+1}$ za $0 \leq q \leq n-2$ i $R_{n-1} \vdash_{\mathrm{A}} (\cdot)R_n$. Kažemo da je $\mathcal{A}(P) = R$ ako je

- $P \models_{\mathcal{A}} \cdot R$ ili
- $P \models_{\mathcal{A}} R$ i $\mathcal{A} : R!$.

Ovako definisano preslikavanje $\mathcal{A}$ je normalni Markovljev algoritam. Činjenica da opisani uslovi za izvođenje jednoznačno određuju redosled koraka transformacije opravdava upotrebu reči algoritam.

Primer 2.10.7 Algoritamska šema

$11 \to 1$

$1 \to \cdot 1$

definiše normalni algoritam $\mathcal{A}$ za koji važi $\mathcal{A}(n) = 0$ za svaki prirodan broj n. ■

2.10.4 Neograničena registarska mašina

Neograničena registarska mašina, ili URM, opisana je šestdesetih godina dvadesetog veka. Sastoji se od beskonačne trake podeljene u registre, tj. ćelije, numerisane prirodnim brojevima. U svaki registar se može upisati bilo koji prirodni broj, a naredbe su:

- $Z(m)$ postavlja nulu u registru m,
- $S(m)$ uvećava sadržaj registra m za 1,
- $T(m, n)$ u registar n upisuje sadržaj registra m i
- $J(m, n, q)$ u slučaju da su sadržaji registara m i n različiti prelazi se na sledeću naredbu, inače se prelazi na q-tu naredbu.

Program je niz numerisanih naredbi čije izvršavanje započinje od prve naredbe i nastavlja se redom, sem eventualno u slučaju naredbe J kada može doći do skoka, a prekida se prelaskom na nepostojeću naredbu. Rezultat se na kraju rada nalazi u registru 0. Sledi primer jednog programa za URM.

Primer 2.10.8 Pod pretpostavkom da su u prva dva registra smešteni brojevi x i y, a da su svi ostali registri sadrže nulu, program:

I_1 $J(1,2,5)$

I_2 $S(0)$

I_3 $S(2)$

I_4 $J(0,0,1)$

izračunava vrednost $x+y$. Prva naredba proverava da li su sadržaji registara 1 i 2 jednaki i ako jesu prelazi na naredbu 5, odnosno završava rad u kom slučaju je rezultat u registru 0. Kako se u registru 2 na početku nalazi nula, ako se posle prvog poređenja u naredbi 1 završi rad, to znači da je i sadržaj registra 1 takođe nula, pa je rezultat $x+0$ jednak x, tj. sadržaju registra 0. Preostale tri naredbe redom za po jedan povećavaju sadržaje registara 0 i 2 i vraćaju izvršavanje na naredbu 1, zapravo y puta za jedan uvećavaju x. ■

2.10.5 Predstavljivost u aritmetici

Ovaj sistem je uveo Gedel u prvoj polovini tridesetih godina dvadesetog veka. Posmatrao je formalnu, takozvanu Robinsonovu, aritmetiku, teoriju prvog reda čiji jezik sadrži promenljive, logičke veznike, konstantu 0 i operacijske simbole $'$ za unarnu operaciju naslednik i binarne $+$ i $\cdot$. *Numerali* su konstantni termi 0, $0'$, $0''$, ... koji odgovaraju prirodnim brojevima 0 1, 2, ... Numerali se označavaju i sa $\underline{0}$, $\underline{1}$, $\underline{2}$ ... Aksiome teorije su aksiome predikatskog računa prvog reda sa jednakošću i uobičajene aksiome za aritmetičke operacije, dok za potrebe predstavljanja aksioma indukcije nije neophodna. Formula $\varphi(x,y)$ ove teorije predstavlja unarnu funkciju $f:\mathbb{N}\to\mathbb{N}$ ako za svako m, $n\in\mathbb{N}$ važi $f(m)=n$ ako i samo ako se u formalnoj aritmetici može dokazati $\vdash \varphi(\underline{m},y)\leftrightarrow(y=\underline{n})$.

2.10.6 Sistemi jednačina

Ovaj formalni sistem izračunavanja je predložio Erbran, a do kraja razvio Gedel do 1934. godine. Pristup se zasniva na ideji da, ako su ψ_1, ..., ψ_k poznate, a ϕ nepoznata funkcija i postoji jedinstveno rešenje po ϕ sistema funkcionalnih jednačina u kojima su funkcije ψ_i i ϕ argumenti jedne drugima, onda je ϕ definabilna sistemom. Gedel je pokazao da se klasa definabilnih funkcija poklapa sa klasom rekurzivnih funkcija.

2.10.7 Algoritamske šeme, *while*-programi i *for*-programi

Algoritamske šeme[39] su kao sredstvo prikazivanja programa, odnosno funkcija, predložili su Goldštajn i Fon Nojman krajem četrdesetih godina dvadesetog veka. Šeme imaju samo jedan ulaz i konačan broj izlaza, a izgrađuju se povezivanjem ulaznih i izlaznih ivica osnovnih blokova. Svaki osnovni blok sadrži neku od naredbi dodeljivanja oblika $X:=0$, $X:=X+1$ ili $X:=X-1$, ili predstavlja uslovnu naredbu oblika $X=0?$. Blok dodeljivanja ima jednu ulaznu i jednu izlaznu ivicu, dok uslovni blok ima jednu ulaznu i dve izlazne ivice koje se odnose na odgovor 'da', tj. 'ne'. Izlazne ivice blokova se mogu spojiti sa ostatkom šeme u proizvoljnoj

[39] Flowchart.

tački. Pokazuje se da je funkcija parcijalno rekurzivna i samo ako je izraziva algoritamskom šemom.

while-programi su zapisi algoritamskih šema pri čemu se ne koriste naredbe skoka, već samo naredbe dodeljivanja, nizanja naredbi i naredba while. I za klasu *while*-programa je dokazano da je jednaka klasi parcijalno rekurzivnih funkcija. Jedno slabljenje *while*-programa je u vidu *for*-programa u kojima se umesto naredbe while koristi naredba oblika *for* Y *do* S koja neki fiksirani broj (Y) puta izvršava naredbu S. Pokazuje se da je klasa *for*-programa jednaka klasi primitivno rekurzivnih funkcija.

2.10.8 Pristup sa totalnim algoritmima

U literaturi je prisutan i nešto drugačiji pristup modelima izračunavanja od prethodno opisanog. U tom modelu se zahteva da je jedan od kriterijuma koje algoritmi ispunjavaju i to da algoritam uvek završava rad, odnosno da su izračunljive funkcije totalne. U slučaju Tjuringovih mašina razmatrali bi se samo programi koji uvek konvergiraju, dok bi se u definiciji 2.4.27 zahtevalo da je funkcija f na koju se primenjuje operacija neograničene minimizacije *regularna*, tj. da u izrazu $(\mu y)(f(x_1, \ldots, x_k, y) = 0)$ funkcija $f(x_1, \ldots, x_k, y)$ bude totalna i da jednačina $f(x_1, \ldots, x_k, y) = 0$ ima uvek rešenje po y. Međutim, na pitanje da li $f(x_1, \ldots, x_k, y) = 0$ ima rešenje po y ne možemo uvek odgovoriti, pa se tako definisane funkcije ne mogu efektivno nabrojati. A kada bi se mogle nabrojati, metodom dijagonalizacije lako bismo konstruisali totalnu, intuitivno izračunljivu, funkciju koja ne bi bila u toj klasi. U ovde predstavljenom pristupu čini se ustupak time što se u klasi razmatranih funkcija nalaze i parcijalne funkcije, ali se zato blokira da se postupkom dijagonalizacije dođe do intutivno izračunljive funkcije koja nije obuhvaćena klasom. Takođe, zahtevajući da se pod izračunljivim funkcijama shvataju samo totalne funkcije došli bismo do neobičnog, možda čak neprihvatljivog, zaključka da mnogi postupci koje karakterišemo kao efektivne nisu obuhvaćeni definicijom. Na primer, takav bi bio postupak ispitivanja zadovoljivosti formula predikatske logike prvog reda, odnosno svi oni postupci kojima se izjašnjavamo o parcijalno odlučivim predikatima.

U vezi sa ovom primedbom prodiskutujmo operaciju $(\min y)(f(x_1, \ldots, x_k, y) = 0)$ definisanu kao najmanje rešenje jednačine $f(x_1, \ldots, x_k, y) = 0$, ako takvo postoji, inače nedefinisano koja primenjena na funkciju f arnosti $k+1$ daje funkciju arnosti k. U slučaju da se zahteva da je funkcija f regularna, tj. da je totalna i da uvek postoji rešenje jednačine $f(x_1, \ldots, x_k, y) = 0$, dobijena funkcija bi uvek bila totalna. Na taj način, polazeći od primitivno rekurzivnih funkcija dodavanjem operacije $\min y$ dobili bismo klasu totalnih parcijalno rekurzivnih funkcija[40]. Primetimo da, ako se u definisanju ograničimo na regularne funkcije, operacije $\min y$ i μy imaju isti efekat. Sa druge strane, ako odbacimo zahtev za regularnošću, operacije nisu jednake. Razmotrimo sledeće funkcije:

$h(x) = \frac{0}{x}$ koja je nedefinisana za $x = 0$ i ima vrednost 0 za $x > 0$,

$g_\mu(x) = (\mu x)(h(x) = 0)$ i

[40] Ova klasa se ponekad naziva i klasa *generalno rekurzivnih funkcija*.

$$g_{\min}(x) = (\min x)(h(x) = 0).$$

Funkcija g_μ nije definisana ni za jedno x pošto $h(0)$ nije definisano, dok je funkcija $g_{\min}$ konstantna i uvek ima vrednost 1. Sa stanovišta intuitivne izračunljivosti, opredeljenje za operaciju neograničene minimazicije ima svoje prednosti. Kao što smo ilustrovali u odeljku 2.4.8, funkcije dobijene neograničenom minimizacijom su intuitivno izračunljive. U slučaju operacije min problem nastaje u tome što reći da je $(\min y)(f(x, y) = 0) = 1$ znači da ili je $f(x, 0) > 0$ ili da $f(x, 0)$ nije definisano. Kako nije uvek moguće utvrditi da li je $f(x, 0)$ definisano, to određivanje vrednosti $(\min y)(f(x, y) = 0)$ nekada podrazumeva okonačanje jednog beskonačnog procesa.

2.11 Za dalje proučavanje

Teorija izračunljivih funkcija je detaljno opisana u [20, 23, 29]. U [29] pristup je zasnovan na Tjuringovim mašinama, a u [20] na neograničenim registarskim mašinama. Pregledi raznih sistema izračunavanja su dati u [77, 87]. Klinijev pristup primitivno rekurzivnim funkcijama je opisan u [65]. Veći broj zadataka iz teorije izračunljivosti je rešen u [120]. [110] je programerski orijentisan uvod u lambda račun i implementaciju funkcionalnih programskih jezika. Opširan prikaz značaja Hilbertovih problema na razvoj matematike, a u okviru toga i tekst o neodlučivosti diofantovskih jednačina, dat je u [81].

3

Klasična iskazna logika

Prema opšte prihvaćenoj klasifikaciji savremena matematička logika obuhvata: teoriju modela, teoriju izračunljivosti, teoriju skupova, teoriju dokaza, konstruktivnu matematiku, algebarsku logiku itd. U ovom poglavlju bavićemo se onim delom matematičke logike koji izučava formalni matematički jezik, teorije (skupove formula) izgrađene na njemu, modele tih teorija i postupke dokazivanja tvrđenja o tim teorijama. Pri tome ćemo se ograničiti na slučaj (klasične) iskazne logike. Naredna poglavlja biće posvećena klasičnoj predikatskoj logici i neklasičnim logikama.

Matematička logika se često koristi u računarstvu što ćemo ilustrovati raznim primerima, a takođe ćemo razmotriti i neke primene računarstva u logici. Klasična (i iskazna i predikatska) logika prvog reda je pogodna za predstavljanje mnogih iskaza koji se standardno pojavljuju u matematici, pa i u teorijskom računarstvu. Na primer, funkcije koje su reprezentovane integrisanim elektronskim kolima mogu se izraziti i analizirati sredstvima klasične iskazne logike. Takođe, mnoge primena matematičke logike počivaju na činjenici da se zaključivanje, koje se pripisuje čoveku, ponekad može sasvim prihvatljivo opisati u terminima automatskog dokazivanja teorema koje se bazira na formalnim, sintaksnim, osnovama. U vezi sa tim razmotrićemo dve metode automatskog dokazivanja: rezoluciju i analitičke tabloe. Zadovoljivosti iskaznih formula je jedan od centralnih problema kojima se bavi iskazna logika. Njegova složenost je dovela do razvoja heurističkih metoda rešavanja, od kojih ćemo jedan prikazati na kraju poglavlja.

3.1 Iskazi i iskazne formule

Među rečenicama koje se upotrebljavaju u svakodnevnom komuniciranju, posebno mesto zauzimaju one koje imaju svojstvo da su bilo tačne, bilo netačne. Takve rečenice se nazivaju *iskazima*.

Primer 3.1.1 Iskazi su: 'Sneg je beo', 'Kuća je žuta', ili 'Ja se zovem Nikola', dok iskazi nisu: 'Dobar dan', ili 'Doviđenja'. ■

U primeru 3.1.1 navedeni su iskazi koji se ne mogu rastavljati na prostije rečenice. Takvi iskazi se nazivaju atomskim. U formalnom radu sa iskazima

uobičajeno je da se iskazna slova p, q, r, s, p_1, p_2, q_1, ... koriste za označavanje atomskih iskaza.

Složeni iskazi se grade od atomskih iskaza upotrebom logičkih veznika: negacije, konjunkcije, disjunkcije, implikacije i ekvivalencije. Logički veznici se mogu primeniti i na složene iskaze čime se dobija bogat skup rečenica koji ćemo na dalje koristiti.

Negacija iskaza p je iskaz 'Nije p'. Negacija iskaza je tačna ako iskaz koji se negira nije tačan. *Konjunkcija* iskaza p i q je iskaz 'p i q'. Konjunkcija dva iskaza je tačna samo ako su tačna oba iskaza. *Disjunkcija* iskaza p i q je iskaz 'p ili q'. Da bi disjunkcija dva iskaza bila tačna dovoljno je da je samo jedan od iskaza tačan. *Implikacija* iskaza p i q je iskaz 'Ako p onda q'. Implikacija dva iskaza je tačna uvek, sem ako je prvi iskaz (p) tačan, a drugi iskaz (q) netačan. *Ekvivalencija* iskaza p i q je iskaz 'p ako i samo ako q'. Da bi ekvivalencija dva iskaza bila tačna, potrebno je da su oba iskaza bilo tačna, bilo netačna. Za složeni iskaz se kaže da je netačan, ukoliko u skladu sa rečenim nije tačan.

Primer 3.1.2 Složeni izkazi su: 'Ako je to Ivanov pas, onda je pas sive boje', 'Nije kuća žuta', 'Sneg je beo i ja se zovem Nikola', 'Idem kući ako i samo ako je kuća žuta ili je njen vlasnik Nikola' itd. ■

Kao što iskazna slova označavaju atomske iskaze, znaci $\neg$, $\wedge$, $\vee$, $\rightarrow$ i $\leftrightarrow$ služe za označavanje logičkih veznika, i to redom negacije, konjunkcije, disjunkcije, implikacije i ekvivalencije. Pojam iskazne formule dat sledećom definicijom, precizno i pregledno uvodi pojam složenih iskaza.

Definicija 3.1.3 *Iskazne formule* (ili kratko: formule) se definišu na sledeći način:

- Iskazna slova su *atomske formule*.
- Atomske formule su iskazne formule.
- Ako su A i B iskazne formule, onda su i $(\neg A), (A \wedge B), (A \vee B), (A \rightarrow B)$ i $(A \leftrightarrow B)$ iskazne formule.
- Iskazne formule se grade samo konačnom primenom prethodnih pravila.

Zagrade, koje uokviruju iskazne formule, ili njihove podformule, se ne pišu kad god to ne unosi zabunu. Asocijativnost konjunkcije i disjunkcije (videti tabelu 3.4) omogućava izvesnu slobodu u pisanju formula. Tako formula $A_1 \vee A_2 \vee A_3 \vee A_4$ označava bilo koju od sledećih formula $((A_1 \vee A_2) \vee A_3) \vee A_4$, $(A_1 \vee A_2) \vee (A_3 \vee A_4)$, $A_1 \vee (A_2 \vee A_3) \vee A_4$, $A_1 \vee ((A_2 \vee A_3) \vee A_4)$ ili $A_1 \vee (A_2 \vee (A_3 \vee A_4))$.

Primer 3.1.4 Neka p i q označavaju redom iskaze. 'Sneg je beo' i 'Ja se zovem Nikola'. Formula $p \wedge q$ označava složeni iskaz 'Sneg je beo i ja se zovem Nikola'. ■

U definiciji 3.1.3, a i na dalje, često se koriste slova A, B, C, itd. da označe iskazne, odnosno, predikatske formule. Formule u kojima figurišu ovakve oznake su *shema-formule* i predstavljaju veći broj 'pravih' formula. Te 'prave' formule se od shema dobijaju takozvanom sistematskom zamenom. Ova zamena podrazumeva da se u shema-formuli isto slovo zamenjuje uvek istom formulom sastavljenom od iskaznih simbola. Tako dobijene formule su primerci[1] shema formula.

[1] Instance.

Primer 3.1.5 Iskazna formula $p \wedge q$ je primerak shema formule $A \wedge B$, pri čemu je A zamenjeno sa p, a B sa q. Iskazna formula $(p \rightarrow \neg q) \leftrightarrow (q \vee r)$ je primerak sheme $A \leftrightarrow (B \vee C)$, pri čemu je A zamenjeno sa $(p \rightarrow \neg q)$, B sa q, a C sa r. ■

Definicija 3.1.6 *Dužina iskazne formule* A, u oznaci $|A|$, je funkcija za koju je:

- ako je A iskazno slovo, onda je $|A| = 1$,
- ako je A oblika $\neg B$, onda je $|A| = 1 + |B|$ i
- ako je A oblika $B \wedge C$, $B \vee C$, $B \rightarrow C$ ili $B \leftrightarrow C$, onda je $|A| = |B| + 1 + |C|$.

Iskazne formule A i B su *podformule* odgovarajućih iskaznih formula uvedenih u drugom koraku definicije 3.1.3, kao i sopstvene podformule. Takođe, sve podformule od A i B su takođe i podformule iskaznih formula čije su A i B podformule. Skup svih podformula formule A se označava sa $Sub(A)$.

Primer 3.1.7 Posmatrajmo formulu $p \wedge (q \wedge r)$. Tada je $Sub(p \wedge (q \wedge r)) = \{p \wedge (q \wedge r), p, q \wedge r, q, r\}$. ■

Teorema 3.1.8 Broj podformula neke iskazne formule nije veći od dužine te formule, tj. $|Sub(A)| \leq |A|$.

Dokaz. Dokaz ćemo sprovesti indukcijom po složenosti formule. Ako je formula A iskazno slovo, onda je $Sub(A) = \{A\}$, pa je $|Sub(A)| = 1 \leq |A|$. Ako je formula A oblika $\neg B$, onda je $Sub(A) = \{\neg B\} \cup Sub(B)$, pa je $|Sub(A)| = 1 + |Sub(B)|$ što je po indukcijskoj pretpostavci manje do jednako od $1 + |B| = |A|$. Ako je formula A oblika $B \wedge C$, onda je $Sub(A) = \{B \wedge C\} \cup Sub(B) \cup Sub(C)$, pa je $|Sub(A)| \leq 1 + |Sub(B)| + |Sub(C)|$. Nejednakost važi jer može biti $Sub(B) \cap Sub(C) \neq \emptyset$. Prema indukcijskoj pretpostavci dalje je $1 + |Sub(B)| + |Sub(C)| \leq 1 + |B| + |C| = |A|$. Preostali slučajevi se analogno ispituju. ■

3.2 Interpretacija i tačnost iskaznih formula

U radu sa iskaznim formulama nas pre svega zanima njihova istinitosna vrednost, a ne stvarne rečenice koje predstavljaju. Upravo glavni zadatak iskazne logike je određivanje načina kako da, polazeći od logičkih vrednosti atomskih iskaza, dođemo do istinitosne vrednosti složenih iskaza.

U klasičnoj logici istinitosna vrednost iskaza je jedna od dve: tačno ili netačno. Prethodno je objašnjeno kada su složeni iskazi tačni, a kada netačni, u zavisnosti od istinitosti iskaza od kojih su formirani. Formalnije se vrednost iskaznih formula uvodi sledećom definicijom, pri čemu simboli $\top$ i $\bot$ redom označavaju istinitosne vrednosti tačno i netačno.

Definicija 3.2.1 *Interpretacija* I je preslikavanje koje iskaznim slovima pridružuje vrednosti iz skupa $\{\top, \bot\}$. *Vrednost formule* A pri interpretaciji I (u oznaci $I(A)$) je sledeća:

- ako je A iskazno slovo p, onda je $I(A) = I(p)$,

- ako su vrednosti $I(A)$ i $I(B)$ formula A i B već izračunate, onda su vrednosti formula $\neg A$, $A \wedge B$, $A \vee B$, $A \to B$ i $A \leftrightarrow B$ redom, kao u tabeli 3.1, $I(\neg A)$, $I(A \wedge B)$, $I(A \vee B)$, $I(A \to B)$ i $I(A \leftrightarrow B)$.

Iskazna formula A je *tačna pri interpretaciji* I (u oznaci: $I \models A$) ako je, polazeći od istinitosne vrednosti pri interpretaciji I iskaznih slova koja se javljaju u A, izračunato $I(A) = \top$. Skup iskaznih formula $T = \{A_1, \ldots, A_n, \ldots\}$ je tačan pri interpretaciji I (u oznaci: $I \models T$) ako je za svaku formulu A_i tog skupa $I(A_i) = \top$.

$I(A)$	$I(B)$	$I(\neg A)$	$I(A \wedge B)$	$I(A \vee B)$	$I(A \to B)$	$I(A \leftrightarrow B)$
$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\top$

Tabela 3.1. Istinitosna tablica osnovnih operacija.

Primetimo da, ako je skup formula $T = \{A_1, \ldots, A_n\}$ konačan, onda $I \models T$ ako i samo ako $I \models A_1 \wedge \ldots \wedge A_n$. Neposrednom proverom u tabeli 3.1 lako se pokazuje da važi $I(A \wedge B) = I(\neg(A \to \neg B))$, $I(A \vee B) = I((\neg A) \to B)$ i $I(A \leftrightarrow B) = I((A \to B) \wedge (B \to A))$. Takođe, očigledno je da se istinitosna vrednost neke formule može u konačnom broju koraka jednoznačno izračunati na osnovu istinitosne vrednosti iskaznih slova koja ulaze u tu formulu.

Primer 3.2.2 Ako je $I(p) = \top$ i $I(q) = \bot$, onda je $I \models \neg q$, $I \models p \leftrightarrow \neg q$ i $I \models (p \wedge q) \to (p \leftrightarrow \neg q)$, a nije $I \models q$, niti $I \models p \wedge q$. Slično, ako je $I(p) = \bot$ i $I(q) = \top$, onda je $I \models q$, $I \models p \leftrightarrow \neg q$ i $I \models (p \wedge q) \to (p \leftrightarrow \neg q)$, a nije $I \models p \wedge q$, niti $I \models p$. ■

Umesto 'formula A je tačna pri interpretaciji I', kaže se i formula A je *zadovoljena pri interpretaciji* I, formula A *važi pri interpretaciji* I, I *zadovoljava* A, ili I je *model* za A. Ako nije $I \models A$, interpretacija I je *kontramodel* za formulu A, što se označava sa $I \not\models A$.

Definicija 3.2.3 Formula A je *zadovoljiva* ako postoji interpretacija I takva da $I \models A$. Ako formula nije zadovoljiva, ona je nezadovoljiva. Skup formula T je *zadovoljiv* ako postoji interpretacija I takva da $I \models T$, u suprotnom skup je nezadovoljiv.

Ovde je veoma važno obratiti pažnju da se o značenju formule, odnosno o istinitosnoj vrednosti formule može govoriti samo nakon interpretiranja. Interpretacija daje semantiku iskaznim slovima i formulama.

Problem zadovoljivosti iskaznih formula, u oznaci[2] SAT, je prvi problem za koji je pokazano da je NP-kompletan (videti odeljak 10.6.4) i kao takav veoma detaljno se obrađuje. O jednom novom pristupu ovom problemu biće više reči u odeljku 3.9.

[2]Od engleskog termina *satisfiability*.

3.2.1 Tautologije, kontradikcije i istinitosne tablice

Na istinitosnu vrednost formule utiču samo istinitosne vrednosti iskaznih slova koja se u njoj pojavljuju (videti primer 3.2.4). Ako u nekoj formuli ima n različitih iskaznih slova, različitih interpretacija ovih iskaznih slova ima ukupno 2^n.

Primer 3.2.4 Neka su A iskazna formula, a I_1 i I_2 dve interpretacije koje se poklapaju na iskaznim slovima koja se javljaju u formuli A (tj. za svako iskazno slovo p koje se pojavljuje u formuli A je $I_1(p) = I_2(p)$). Koristeći indukciju po složenosti formule, pokazujemo da je $I_1(A) = I_2(A)$. Slučaj kada je formula iskazno slovo je trivijalan. Ako je $A = B \wedge C$, onda je $I_1(A) = I_1(B \wedge C) = I_1(B) \wedge I_1(C) = I_2(B) \wedge I_2(C) = I_2(B \wedge C) = I_2(A)$ i slično za ostale slučajeve. ■

Primer 3.2.5 U tabeli 3.2 su u četiri reda prikazane sve $(4 = 2^2)$ različite interpretacije formule koja od iskaznih slova sadrži samo iskazna slova p i q. ■

	p	q
I_1	$\top$	$\top$
I_2	$\top$	$\bot$
I_3	$\bot$	$\top$
I_4	$\bot$	$\bot$

Tabela 3.2. Sve interpretacije iskaznih slova p i q.

Definicija 3.2.6 Formula A je *tautologija* (kao sinonim se često upotrebljava i termin *valjana formula*), u oznaci $\models A$, ako je tačna pri svakoj interpretaciji. Formula A je *kontradikcija* ako nije tačna ni pri jednoj interpretaciji.

Primer 3.2.7 Formula $((p \to q) \wedge p) \to q$ je jedna tautologija. Ovo se neposredno proverava ispisivanjem sve 4 različite interpretacije iskaznih slova p i q, računanjem istinitosnih vrednosti podformula formule i, konačno, računanjem istinitosne vrednosti same formule koju razmatramo, kao što je prikazano u tabeli 3.3. Sličnim razmatranjem se ustanovljava da je formula $(p \to q) \wedge (p \wedge \neg q)$ kontradikcija. ■

p	q	$p \to q$	$(p \to q) \wedge p$	$((p \to q) \wedge p) \to q$
$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$	$\bot$	$\top$
$\bot$	$\bot$	$\top$	$\bot$	$\top$

Tabela 3.3. Istinitosna tablica formule.

Postupak koji je korišten u primeru 3.2.7 se naziva *metoda istinitosnih tablica*. Istinitosne tablice se koriste za utvrđivanje da li je neka formula tautologija, zadovoljiva ili kontradikcija, tj. nezadovoljiva. Primetimo da, pošto je broj iskaznih

slova u svakoj formuli konačan, konačan je i broj različitih intepretacija tih iskaznih slova, pa i broj redova istinitosne tablice formule. Dalje, i svaki red se izračunava u konačnom broju koraka, tako da zaključujemo da važi teorema

Teorema 3.2.8 Problemi ispitivanja da li je iskazna formula tautologija, zadovoljiva ili kontradikcija su odlučivi.

Neke od važnih tautologija su (sa $\top$ su označene uvek tačne formule, dok $\bot$ označava uvek netačne formule) prikazane su u tabeli 3.4.

Tautologije su vrsta formula koje imaju veliku primenu. To su formule koje su tačne na osnovu svoje forme, odnosno načina na koje su izgrađene, a ne na osnovu neke posebne vrednosti iskaznih slova od kojih su sastavljene. Tautologije su zato obrasci ispravnog zaključivanja, što će biti kasnije ilustrovano primerima. Pošto je negacija svake tautologije kontradikcija, i te, uvek netačne, formule se obilato koriste kao pokazatelji stranputica do kojih se došlo u zaključivanju.

Primer 3.2.9 Iskaz 'ako niz nije ograničen, onda nije konvergentan' predstavlja kontrapoziciju poznatog tvrđenja iz realne analize 'ako je niz konvergentan, onda je ograničen', zbog čega su iskazi ekvivalentni. ■

U primeru 3.2.10 su opisani generalizovani slučajevi nekih od navedenih tautologija. U vezi tautologija važna su sledeća tvrđenja. Prvo od njih se odnosi na postupak zaključivanja *modus ponens*.

Primer 3.2.10 Indukcijom se mogu dokazati generalisani zakoni De Morgana i distributivni zakoni: $\models \neg(\wedge_{i=1}^{n} A_i) \leftrightarrow (\vee_{i=1}^{n} \neg A_i)$, $\models \neg(\vee_{i=1}^{n} A_i) \leftrightarrow (\wedge_{i=1}^{n} \neg A_i)$, $\models ((\vee_{i=1}^{m} A_i) \wedge (\vee_{j=1}^{n} B_j)) \leftrightarrow (\vee_{i=1}^{m}(\vee_{j=1}^{n}(A_i \wedge B_j)))$, $\models ((\wedge_{i=1}^{m} A_i) \vee (\wedge_{j=1}^{n} B_j)) \leftrightarrow (\wedge_{i=1}^{m}(\wedge_{j=1}^{n}(A_i \vee B_j)))$. ■

Teorema 3.2.11 Ako su formule A i $A \to B$ tautologije, tautologija je i formula B.

Dokaz. Razmotrimo bilo koju interpretaciju I skupa formula $\{A, B\}$. Ako je $I(B) = \bot$, zbog $I(A \to B) = \top$ mora biti $I(A) = \bot$, što je kontradikcija, jer je $I(A) = \top$. Dakle, mora biti $I(B) = \top$. Kako je I proizvoljna interpretacija, sledi da je $\models B$. ■

Tvrđenjem 3.2.12 se uvodi metod *ekvivalentnih transformacija*.

Teorema 3.2.12 Neka je formula $A \leftrightarrow B$ tautologija i $F(A)$ formula čija je A podformula i neka je formula $F(B)$ nastala zamenom jedne ili više pojava formule A formulom B. Tada je $F(A) \leftrightarrow F(B)$ tautologija.

Dokaz. Razmotrimo bilo koju interpretaciju I skupa formula $\{A, B, F(A), F(B)\}$. Pošto je $\models A \leftrightarrow B$ to znači da je $I(A) = I(B)$. Dokaz se sprovodi indukcijom po složenosti formule $F(A)$. Ako je formula $F(A)$ jednaka formuli A, onda je $F(B)$ formula B, pa je $I(F(A)) = I(A) = I(B) = I(F(B))$. Pretpostavimo dalje da $F(A)$ nije isto što i formula A. Ako je $F(A)$ oblika $\neg C$, onda je $I(F(A)) = I(\neg C(A))$. Po indukcijskoj pretpostavci je $I(C(A)) = I(C(B))$, pa je $I(F(A)) = I(\neg C(B)) = I(F(B))$. Slično se analiziraju i ostali slučajevi. Kako ovo važi za proizvoljnu interpretaciju, sledi da je $\models F(A) \leftrightarrow F(B)$. ■

$A \to A$	(zakon refleksivnosti implikacije)
$A \vee \neg A$	(zakon isključenja trećeg)
$\neg(A \wedge \neg A)$	(zakon neprotivrečnosti)
$\neg\neg A \leftrightarrow A$	(zakon dvojne negacije)
$((A \to B) \wedge (B \to C)) \to (A \to C)$	(zakon tranzitivnosti za implikaciju)
$((A \leftrightarrow B) \wedge (B \leftrightarrow C)) \to (A \leftrightarrow C)$	(zakon tranzitivnosti za ekvivalenciju)
$(A \to B) \leftrightarrow (\neg A \vee B)$	(zakon uklanjanja implikacije)
$(A \leftrightarrow B) \leftrightarrow ((A \to B) \wedge (B \to A))$	(zakon uklanjanja ekvivalencije)
$(\neg A \to (B \wedge \neg B)) \to A$	(zakon svođenja na apsurd)
$(A \wedge A) \leftrightarrow A$	(zakon idempotencije za $\wedge$)
$(A \vee A) \leftrightarrow A$	(zakon idempotencije za $\vee$)
$(A \wedge B) \leftrightarrow (B \wedge A)$	(zakon komutativnosti za $\wedge$)
$(A \vee B) \leftrightarrow (B \vee A)$	(zakon komutativnosti za $\vee$)
$(A \wedge (B \wedge C)) \leftrightarrow ((A \wedge B) \wedge C)$	(zakon asocijacijativnosti za $\wedge$)
$(A \vee (B \vee C)) \leftrightarrow ((A \vee B) \vee C)$	(zakon asocijacijativnosti za $\vee$)
$(A \wedge (A \vee B)) \leftrightarrow A$	(zakon apsorpcije)
$(A \vee (A \wedge B)) \leftrightarrow A$	(zakon apsorpcije)
$(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$	(zakon distribucije $\wedge$ prema $\vee$)
$(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$	(zakon distribucije $\vee$ prema $\wedge$)
$(\neg(A \wedge B)) \leftrightarrow (\neg A \vee \neg B)$	(zakon De Morgana)
$(\neg(A \vee B)) \leftrightarrow (\neg A \wedge \neg B)$	(zakon De Morgana)
$(A \wedge (A \to B)) \to B$	(modus ponens)
$((A \to B) \wedge \neg B) \to \neg A$	(modus tolens)
$(\neg B \to \neg A) \leftrightarrow (A \to B)$	(zakon kontrapozicije)
$(A \vee \top) \leftrightarrow \top$	(zakon disjunkcije sa tautologijom)
$(A \wedge \top) \leftrightarrow A$	(zakon konjunkcije sa tautologijom)
$(A \vee \bot) \leftrightarrow A$	(zakon disjunkcije sa kontradikcijom)
$(A \wedge \bot) \leftrightarrow \bot$	(zakon konjunkcije sa kontradikcijom)
$\neg\top \leftrightarrow \bot$	(zakon negacije tautologije)

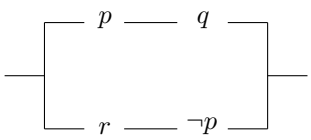
Tabela 3.4. Spisak nekih tautologija.

3.2.2 Druge interpretacije iskaznih formula

Ovde ćemo navesti još dve vrste interpretacija iskaznih formula u kojima se ne interpretiraju znaci $\to$ i $\leftrightarrow$. Imajući u vidu izloženo u odeljku 3.3 ovo nije stvarno ograničenje pošto se sve formule mogu transformisati u takav oblik.

U prvoj interpretaciji iskaznim slovima i njihovim negacijama odgovaraju prekidači koji su u stanju 0 ili 1 (pri tome ako je iskazno slovo p u stanju 0, njegova negacija $\neg p$ je u stanju 1 i obrnuto), dok se složenije formule u kojima se negacija primenjuje samo na iskazna slova interpretiraju kao logičke mreže koje provode struju. Konjunkciji odgovara redno povezivanje podformula, a disjunkciji paralelno (tj. razgranato) povezivanje. Na slici 3.1 je dat primer jedne mreže kojoj odgovara formula $(p \wedge q) \vee (r \wedge \neg p)$. Može se uočiti da tautologijama odgovaraju mreže (i obrnuto) kroz koje uvek protiče struja.

Iskazna slova se mogu interpretirati i kao elementi partitivnog skupa $\mathbb{P}(S)$ nekog nepraznog skupa S, dok se znaci $\neg$, $\wedge$ i $\vee$ interpretiraju kao skupovne operacije komplementa, preseka i unije. Ako je skup S jednočlan, ova interpretacija se svodi



Slika 3.1. Logička mreža.

na istinitosnu interpretaciju iz odeljka 3.2.1, pri čemu se $\emptyset$ poistovećuje sa $\bot$, a sam skup S sa $\top$. Međutim, ako skup S ima bar dva člana, vrednost formule se finije gradira između krajnosti koje predstavljaju $\emptyset$ i S, pa se ovakva interpretacija naziva *viševrednosna* ili *polivalentna*.

3.3 Normalne forme za iskaznu logiku

Povremeno se javlja potreba da se neka formula transformiše iz jednog oblika u drugi, naročito u oblik *normalne forme*.

U jednoj vrsti ovih transformacij ključnu ulogu imaju ranije navedene valjane formule. Pomoću njih, polazeći od neke formule A, u nizu koraka redom se dobijaju međusobno ekvivalentne formule A_1, A_2, ..., što se završava dobijanjem željene normalne forme, u oznaci $NF(A)$. Zbog zakona tranzitivnosti ekvivalencije (videti tabelu tautologija) važi da su formule A i $NF(A)$ ekvivalentne, tj. A je valjana (odnosno zadovoljiva, ili kontradikcija) ako i samo ako je $NF(A)$ valjana (zadovoljiva, ili kontradikcija).

Definicija 3.3.1 *Literal* je atomska formula ili njena negacija. Formula je u *konjunktivnoj normalnoj formi*[3], ako je oblika

$$D_1 \wedge D_2 \wedge \ldots \wedge D_n$$

pri čemu je svaki od D_i-ova disjunkcija literala oblika $D_i = L_{i,1} \vee \ldots \vee L_{i,m_i}$. Formula je u *disjunktivnoj normalnoj formi*[4], ako je oblika

$$K_1 \vee K_2 \vee \ldots \vee K_n$$

pri čemu je svaki od K_i-ova konjunkcija literala oblika $K_i = L_{i,1} \wedge \ldots \wedge L_{i,m_i}$.

Primer 3.3.2 Formula $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$ je konjunktivnoj normalnoj formi, dok je formula $(p \wedge \neg q \wedge r) \vee (\neg p \wedge q)$ u disjunktivnoj normalnoj formi. ■

Konjunktivna normalna forma će imati važnu ulogu u proceduri rezolucije u odeljku 3.7, a jedna posebna vrsta konjunktivne normalne forme, takozvane Hornove formule, u odeljku 5.13.1. Svaka formula se može transformisati u njoj

[3]Ova forma se naziva i sastavna formula, a svaki član u konjunkciji sastavak. Takođe se koriste i oznake CNF i POS (*product of sums*).

[4]Ova forma se naziva i rastavna formula, a svaki član u konjunkciji rastavak. Takođe se koristi i oznaka SOP (*sum of products*).

ekvivalentnu konjunktivnu, odnosno disjunktivnu normalnu formu, što se dokazuje u teoremi 3.3.3. Za neku formulu A konjunktivna normalna forma se označava sa $KNF(A)$, a disjunktivna normalna forma sa $DNF(A)$.

Teorema 3.3.3 Za svaku formulu A postoje njoj ekvivalentne formule $KNF(A)$ i $DNF(A)$ u konjunktivnoj, odnosno u disjunktivnoj normalnoj formi.

Dokaz. Na osnovu zakona o uklanjanju implikacije i ekvivalencije i prema ekvivalentnoj transformaciji, formula A je ekvivalentna formuli u kojoj nema znakova $\rightarrow$ i $\leftrightarrow$, zbog čega ćemo bez gubitka opštosti pretpostaviti da se u formuli A nalaze samo logički veznici $\neg$, $\wedge$ i $\vee$. Dokaz tvrđenja ćemo sprovesti indukcijom po složenosti formule A. Ako je A iskazno slovo, tvrđenje trivijalno važi.

Neka je A oblika $\neg A_1$ i neka za A_1 tvrđenje važi, tj. $KNF(A_1)$ i $DNF(A_1)$ su redom konjunktivna i disjunktivna normalna forma ekvivalentne formuli A_1. Neka je $KNF(A_1) = \wedge_{i=1}^{n}(\vee_{j=1}^{m_i} L_{i,j})$ za neke literale $L_{i,j}$. Tada je, prema uopštenim zakonima De Morgana (primer 3.2.10) A ekvivalentno redom sa $\neg \wedge_{i=1}^{n} (\vee_{j=1}^{m_i} L_{i,j})$, $\vee_{i=1}^{n} \neg(\vee_{j=1}^{m_i} L_{i,j})$, $\vee_{i=1}^{n}(\wedge_{j=1}^{m_i} \neg L_{i,j})$ i $\vee_{i=1}^{n}(\wedge_{j=1}^{m_i} \overline{L_{i,j}})$, gde je $\overline{L_{i,j}}$ oznaka za negaciju literala $L_{i,j}$ u kojoj je, eventualno, prema zakonu dvojne negacije uklonjena dvostruka negacija[5]. Poslednja formula je u disjunktivnoj normalnoj formi i ekvivalentna je sa A, pa je to tražena $DNF(A)$. Analogno, polazeći od $\neg DNF(A_1)$, dobija se $KNF(A)$.

Neka je A oblika $A_1 \wedge A_2$, i neka za A_1 i A_2 tvrđenje važi, tj. $KNF(A_1)$, $DNF(A_1)$, $KNF(A_2)$ i $DNF(A_2)$ su redom konjunktivne i disjunktivne normalne forme ekvivalentne formulama A_1, odnosno A_2. Odmah se vidi da je formula A ekvivalentna formuli $KNF(A_1) \wedge KNF(A_2)$ koja je u konjunktivnoj normalnoj formi, pa je to $KNF(A)$. Takođe, formula A je ekvivalentna formuli $DNF(A_1) \wedge DNF(A_2)$. Neka su $DNF(A_1) = \vee_{i=1}^{m} K(A_1)_i$ i $DNF(A_2) = \vee_{j=1}^{n} K(A_2)_i$, gde su $K(A_1)_i$ i $K(A_2)_j$ odgovarajući konjunkti literala. Prema uopštenim distributivnim zakonima (primer 3.2.10), formula A je ekvivalentna sa $\vee_{i=1}^{m}(\vee_{j=1}^{n}(K'(A_1)_i \wedge K'(A_2)_j))$. Sa K' smo označili da su u $K'(A_1)_i \wedge K'(A_2)_j$ prema zakonima idempotencije eliminisane dvostruke pojave literala, kao i da su prema zakonu disjunkcije sa kontradikcijom eliminisani svi konjunkti u kojima se našao i neki literal i njegova negacija. Dakle, poslednja formula je u disjunktivnoj normalnoj formi i predstavlja traženu formulu $DNF(A)$.

Analogno se sprovodi dokaz slučaja za $A = A_1 \vee A_2$. ■

U proceduri NomalnaForma je opisan jedan od postupaka kojim se dobijaju konjunktivna, odnosno disjunktivna, normalna forma.

procedure NormalnaForma
begin
1 Koristeci zakone o uklanjanju ekvivalencije i implikacije
dobiti formule u kojima nema simbola $\leftrightarrow$ i $\rightarrow$.
2 Ponavljati primenu
2a De Morganovih zakona i
2b zakona dvojne negacije

[5]Ako je $L_{i,j}$ negacija iskaznog slova p, tada je $\overline{L_{i,j}}$ jednako samom p.

da bi se simboli $\neg$ uklonili, ili spustili neposredno do iskaznih slova.
3 Ako se želi konjunktivna normalna forma, ponavljati primenu zakona distributivnosti $\vee$ prema $\wedge$.
4 Ako se želi disjunktivna normalna forma, ponavljati primenu zakona distributivnosti $\wedge$ prema $\vee$.
end

Primer 3.3.4 Primenimo proceduru NormalnaForma za dobijanju disjunktivne normalne forme formule $(p \vee \neg q) \rightarrow r$. Polazeći od same formule $(p \vee \neg q) \rightarrow r$, primenom pravila (1) dobija se formula $\neg(p \vee \neg q) \vee r$. Primenom pravila (2a) dobija se formula $(\neg p \wedge \neg\neg q) \vee r$. Primenom pravila (2b) dobija se formula $(\neg p \wedge q) \vee r$, koja je u disjunktivnoj normalnoj formi. U ovom izvođenju nije korišteno pravilo (4), ali primena dualnog pravila (3) je neophodna da bi se formula $(p \wedge (q \rightarrow r)) \rightarrow s$ transformisala u konjuktivnu normalnu formu $(s \vee \neg p \vee q) \wedge (s \vee \neg p \vee \neg r)$. ■

Normalne forme se direktno primenjuju u ispitivanju valjanosti, zadovoljivosti i nezadovoljivosti iskaznih formula. Recimo, neka je $DNF(A) = K_1 \vee K_2 \vee \ldots \vee K_n$. Ako svaki od K_i-ova sadrži istovremeno i neko iskazno slovo i njegovu negaciju, prema navedenim tautologijama, A je pri svakoj interpretaciji netačna. Ako bar jedan među K_i-ovima ne sadrži istovremeno nijedno iskazno slovo i njegovu negaciju, formula A je tačna pri interpretaciji koja zadovoljava upravo taj K_i. Slično, neka je $KNF(A) = D_1 \wedge D_2 \wedge \ldots \wedge D_n$. Ako svaki od D_i-ova sadrži istovremeno i neko iskazno slovo i njegovu negaciju, onda je A tautologija.

Primer 3.3.5 Sve formule čija je konjunktivna normalna forma oblika $(p \vee q \vee \neg q) \wedge (p \vee \neg q \vee r \vee \neg r)$ su valjane. Sve formule čija je disjunktivna normalna forma oblika $(p \wedge q \wedge \neg q) \vee (p \wedge \neg q \wedge r \wedge \neg r)$ su kontradikcije. ■

Primetimo da za neku formulu A njene $KNF(A)$ i $DNF(A)$ ne moraju biti jedinstvene. Recimo, formula $A = p \vee q \vee \neg q$ je već i sama u disjunktivnoj normalnoj formi, ali jedna njena $DNF(A)$ je i oblika $q \vee \neg q$. Takođe, različite međusobno ekvivalentne formule imaju iste konjunktivne i disjunktivne normalne forme.

U mnogim primenama, poput dizajniranja digitalnih kola, interesantne su što kraće normalne forme. Međutim, primene zakona distributivnosti pri pravljenju normalnih formi često dovode do eksponencijalnog rasta dužine što u principu nije moguće izbeći. Zbog toga se normalne forme optimizuju, recimo metodom Karnuaugh (Karnu). Iako međusobno različite, metode optimizacije se svode na skraćivanje na osnovu distributivnih i drugih iskaznih zakona, kao u primeru 3.3.6.

Primer 3.3.6 Skraćivanjem na osnovu distributivnog zakona i zakona disjunkcije sa tautologijom, polazeći od formule $(p \vee q) \wedge (p \vee \neg q)$ dobijaju se ekvivalentne formule $p \vee (q \wedge \neg q)$, $p \vee \bot$ i p. ■

Primer 3.3.7 Razmotrimo formulu $A = (p_1 \vee q_1) \wedge \ldots \wedge (p_n \vee q_n)$. Jednu $DNF(A)$ čine svi disjunkti oblika $x_1 \wedge \ldots \wedge x_n$ gde je x_i bilo p_i, bilo q_i, pa ona očigledno ima eksponencijalnu dužinu u odnosu na broj iskaznih slova polazne formule. Kakvi su zahtevi za proizvoljnu $DNF(A)$? Sledeće razmatranje se oslanja na ekvivalentnost formula A i $DNF(A)$. Najpre procenimo dužinu disjunkata iz $DNF(A)$. Pošto

je A zadovoljiva formula, to je i $DNF(A)$, pa $DNF(A)$ sadrži bar jedan disjunkt K_l koji nije kontradiktoran. Neka taj disjunkt ne sadrži iskazna slova p_i i q_i za neko i i neka je I interpretacija za koju je $I \models K_l$. Za tu interpretaciju mora biti i $I \models A$. Neka je I' interpretacija koja se od I eventualno razlikuje po tome što je $I'(p_i) = I'(q_i) = \bot$. Prema primeru 3.2.4 važi $I' \models K_l$, ali je $I'(A) = \bot$, pa A i $DNF(A)$ ne bi bile ekvivalentne. Odatle, za svako i svaki disjunkt iz $DNF(A)$ mora sadržati bar jedno od, eventualno negiranih, iskaznih slova p_i, q_i. Sada procenimo broj disjunkata u $DNF(A)$. Posmatrajmo interpretaciju I takvu da je za svako i bilo $I \models p_i$, bilo $I \models q_i$, ali nije $I \models p_i \wedge q_i$. Interpretacija ovog oblika, ako razmatramo samo iskazna slova koja se javljaju u A, ima 2^n. Neka je, recimo, za svako i, $I(p_i) = \top$ i $I(q_i) = \bot$. Pošto $I \models A$, mora postojati i disjunkt K_l iz $DNF(A)$ takav da je $I \models K_l$. Prema prethodnom, da bi bilo $I(K_l) = \top$, K_l mora za svako i sadržati bar jednu od formula p_i i $\neg q_i$, a ne sme sadržati ni jednu od formula $\neg p_i$ i q_i. Ni jedna od interpretacija razmatranog oblika koja se razlikuje od I ne zadovoljava K_l. Dakle, za svaku od 2^n interpretaciju posmatranog oblika postoji poseban disjunkt u $DNF(A)$ koga zadovoljava samo ta interpretacija, pa je broj disjunkata u $DNF(A)$ bar 2^n. Iz navedenog sledi da je svaka $DNF(A)$ eksponencijalno duža od A. ■

Ako u primeru 3.3.7 želimo da sačuvamo disjunktivni oblik, nikakva optimizacija ne bi suštinski skratila rezultujuću normalnu formu. Takođe, u mnogim slučajevima u kojima se normalna forma i može uprostiti nije pogodno najpre generisati dugačku normalnu formu, pa je potom skraćivati, jer to zahteva puno računarskih resursa. Zato se razvijaju postupci koji bilo direktno prave kraću normalnu formu, bilo odbacuju zahtev za ekvivalentnošću formule i njene normalne forme. U odeljku 4.3 ćemo razmotriti postupak nalaženja kompaktnih i kratkih normalnih formi koje nisu obavezno u disjunktivnom, odnosno konjunktivnom, obliku. U odeljku 3.3.1 ćemo prikazati postupak u kome se vrši transformacija u jednu vrstu konjunktivne normalne forme, pri čemu se očuvava jedino svojstvo zadovoljivosti polazne formule.

3.3.1 Transformacija u definicione forme

Normalna forma u konjunktivnom obliku čiji postupak dobijanja će biti naveden naziva se *definiciona forma*[6]. Njenu prednost u odnosu na konjunktivnu normalnu formu dobijenu ekvivalentnim transformacijama predstavlja to što joj je dužina najviše polinomijalno, a ne i eksponencijalno, ograničena dužinom polazne formule. Definiciona forma neke formule je zadovoljiva ako i samo ako je zadovoljiva polazna formula.

Ako se u polaznoj formuli nalazi podformula $H \leftrightarrow J$, nju treba pre transformacije prepisati u ekvivalentan oblik $(H \rightarrow J) \wedge (J \rightarrow H)$. Ideja na kojoj se bazira transformacija formule F u definicionu formu je da se za svaku formulu G koja je podformula of F uvede novo iskazno slovo L_G i formula C_G u konjunktivnoj normalnoj formi koja će čuvati informaciju o strukturi formule G. Preciznije:

- ako je G atomska formula, onda je $C_G = (L_G \vee \neg G) \wedge (\neg L_G \vee G)$,
- ako je $G = \neg H$, onda je $C_G = (L_G \vee L_H) \wedge (\neg L_G \vee \neg L_H)$,

[6] Definitional form.

- ako je $G = H \wedge J$, $C_G = (L_G \vee \neg L_H \vee \neg L_J) \wedge (\neg L_G \vee L_H) \wedge (\neg L_G \vee L_J)$,
- ako je $G = H \vee J$, $C_G = (L_G \vee \neg L_H) \wedge (L_G \vee \neg L_J) \wedge (\neg L_G \vee L_H \vee L_J)$ i
- ako je $G = H \rightarrow J$, $C_G = (L_G \vee L_H) \wedge (L_G \vee \neg L_J) \wedge (\neg L_G \vee \neg L_H \vee L_J)$.

Ako je formula G oblika $H \rho J$, gde je ρ neki binarni logički veznik, formula C_G je u konjuktivni oblik transformisana formula $L_G \leftrightarrow (L_H \rho L_J)$, a slična situacija je i ako je G oblika $\neg H$. Odatle, formula C_G uvodi iskazno slovo L_G kao neku vrstu skraćenice za formulu G. Ovde treba obratiti pažnju da je formula F sopstvena podformula, pa se uvođenje iskaznog slova L_F i formule C_F odnosi i na nju.

Definicija 3.3.8 Neka je $C(F) = \wedge_{G \in Sub(F)} C_G$ konjunkcija svih formula C_G takvih da je G podformula od F. *Definiciona forma* F' formule F je formula $C(F) \wedge L_F$.

Osnovno tvrđenje koje se odnosi na definicionu formu je:

Teorema 3.3.9 Formula F je zadovoljiva ako i samo ako je zadovoljiva njena definiciona forma.

Dokaz. ($\Leftarrow$) Pretpostavimo da je formula F zadovoljiva. Tada postoji interpretacija I takva da je $I \models F$. Posmatrajmo interpretaciju I' takvu da je $I'(L_G) = I(G)$ za svaku podformulu G formule F i $I'(G) = I(G)$ za svako iskazno slovo G koje se javlja u F. Pošto je $I(F) = \top$, važi da je $I'(L_F) = \top$. Pokazaćemo i da je $I'(C_G) = \top$ za svaku formulu $G \in Sub(F)$. Ako je G iskazno slovo, onda je $C_G = (L_G \vee \neg G) \wedge (\neg L_G \vee G)$, pa tvrđenje važi po definiciji interpretacije I'. Ako je $G = \neg H$, onda je $C_G = (L_G \vee L_H) \wedge (\neg L_G \vee \neg L_H)$. Kako je $I(G) = \neg I(H)$, to je i $I'(L_G) = \neg I'(L_H)$, pa je $I'(C_G) = \top$. Ako je $G = H \wedge J$, onda je $C_G = (L_G \vee \neg L_H \vee \neg L_J) \wedge (\neg L_G \vee L_H) \wedge (\neg L_G \vee L_J)$. Kako je $I(G) = I(H) \wedge I(J)$, to je $I'(L_G) = I'(L_H) \wedge I'(L_J)$, pa je $I'(C_G) = \top$. Slično se pokazuju i preostali slučajevi. Prema tome, $I'(C(F)) = I'(L_F) = \top$, pa je $I'(F') = \top$, odnosno formula F' je zadovoljiva.

($\Rightarrow$) Pretpostavimo da je formula F' zadovoljiva, tj. da postoji interpretacija I za koju je $I \models F'$. Tada je $I \models L_F$ i $I \models C(F)$, tj. $I \models C_G$ za svaku formulu $G \in Sub(F)$. Posmatrajmo interpretaciju I' takvu da je $I'(G) = I(L_G)$ za svako iskazno slovo G koje se javlja u F i pokažimo da to isto važi i za svaku formulu $G \in Sub(F)$. Najpre, tvrđenje važi za iskazna slova na osnovu definicije interpretacije I'. Posmatrajmo formulu $G = \neg H$. Tada je $C_G = (L_G \vee L_H) \wedge (\neg L_G \vee \neg L_H)$. Pošto važi $I(C_G) = \top$, to je $I(L_G) = \neg I(L_H)$. Na osnovu indukcijske pretpostavke se dobija $I(L_G) = \neg I'(H)$, odakle direktno sledi $I'(G) = I(L_G)$. Neka je $G = H \wedge J$. Tada je $C_G = (L_G \vee \neg L_H) \wedge (L_G \vee \neg L_J) \wedge (\neg L_G \vee L_H \vee L_J)$. Pošto važi $I(C_G) = \top$, to je $I(L_G) = I(L_H) \wedge I(L_J)$. Na osnovu indukcijske pretpostavke se dobija $I(L_G) = I'(H) \wedge I'(J)$, odakle direktno sledi $I'(G) = I(L_G)$. Slično se dokazuju i preostali slučajevi. Prema tome je $I'(F) = I(L_F) = \top$, pa je F zadovoljiva. ■

Primetimo da se definiciona forma neke formule nalazi u konjunktivnoj normalnoj formi. Takođe, njena dužina $|F'|$ je ograničena linearnom funkcijom broja podformula, pa i dužine, formule F.

Primer 3.3.10 Ako je formula F iskazno slovo p, njena definiciona forma je $F' = (L_p \vee \neg p) \wedge (\neg L_p \vee p) \wedge L_p$. Ako je F oblika $p \vee q$, onda su formule $C_p = (L_p \vee \neg p) \wedge (\neg L_p \vee p)$, $C_q = (L_q \vee \neg q) \wedge (\neg L_q \vee q)$ i $C_F = (L_F \vee \neg L_p) \wedge (L_F \vee \neg L_q) \wedge (\neg L_F \vee L_p \vee L_q)$, dok je $F' = C_F \wedge C_p \wedge C_q \wedge L_F$.

3.3.2 Potpune normalne forme

Da bi se za neku formulu A dobila $DNF(A)$ moguće je iskoristiti i istinitosnu tablicu formule A tako što svaki red tablice u kome formula ima vrednost $\top$ odgovara jednom konjunktu. Posmatrajmo jedan red tablice koji odgovara interpretaciji I za koji je $I(A) = \top$ i opišimo odgovarajući konjunkt K_I. Literali koji se javljaju u konjunktu K_I se određuju na sledeći način. Ako je $I(p) = \top$ za neko iskazno slovo koje se nalazi u formuli A, u K_I se javlja literal p, a ako je $I(p) = \bot$, u K_I se javlja literal $\neg p$. Da bi se dobila $KNF(A)$ radi se dualno, tj. posmatraju se samo redovi u kojima je vrednost $I(A) = \bot$, dok se u D_I javlja literal p ako je $I(p) = \bot$, a $\neg p$ ako je $I(p) = \top$. Primetimo da se u svakom disjunktu, odnosno konjunktu, ovako dobijenih formi pojavljuju sva iskazna slova (ili njihove negacije) koja se nalaze i u samoj formuli A. Zbog toga se ovakve normalne forme nazivaju *potpunim*.

Definicija 3.3.11 Neki sistem prikazivanja objekata je *kanonski* ako su dva objekta jednaka (ekvivalentna) ako i samo ako imaju istu reprezentaciju u tom sistemu.

Potpuna disjunktivna normalna forma i potpuna konjunktivna normalna forma su ekvivalentne polaznoj formuli i jedinstvene, pa predstavljaju kanonske sisteme. Zbog toga se ispitivanje ekvivalentnosti formula može sprovesti upoređivanjem odgovarajućih parova ovih formi.

Primer 3.3.12 Za formulu $p \to q$ istinitosna tablica je data u tabeli 3.5. Prema opisanom postupku, potpuna disjunktivna normalna forma je oblika $(p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$, dok je potpuna konjunktivna normalna forma oblika $\neg p \vee q$. ■

p	q	$p \to q$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\top$
$\bot$	$\bot$	$\top$

Tabela 3.5. Istinitosna tablica formule $p \to q$.

Primer 3.3.13 Puni sabirač[7] je elektronsko kolo sa tri ulaza x, y i c, gde x i y odgovaraju bitovima koji se sabiraju, a c prenosu pri sabiranju prethodnog para bitova, i dva izlaza: r, bit rezultata, i s, bit prenosa koji predstavlja ulazni podatak

[7] *Full adder*. Drugi naziv je *binarni sabirač*.

za sabiranje sledećeg para bitova. Uvidom u istinitosnu tablicu 3.6, dobijaju se potpune disjunktivne normalne forme za r, $(\neg x \wedge \neg y \wedge c) \vee (\neg x \wedge y \wedge \neg c) \vee (x \wedge \neg y \wedge \neg c) \vee (x \wedge y \wedge c)$ i za s, $(\neg x \wedge y \wedge c) \vee (x \wedge \neg y \wedge c) \vee (x \wedge y \wedge \neg c) \vee (x \wedge y \wedge c)$. Ove forme se mogu optimizovati u $(\neg x \wedge ((\neg y \wedge c) \vee (y \wedge \neg c))) \vee (x \wedge ((\neg y \wedge \neg c) \vee (y \wedge c)))$, odnosno $(y \wedge c) \vee (x \wedge ((\neg y \wedge c) \vee (y \wedge \neg c)))$. Ako uvedemo veznik za ekskluzivnu disjunkciju $\veebar$ tako da je $p \veebar q$ skraćenica za $(\neg p \wedge q) \vee (p \wedge \neg q)$, onda se optimizovane forme mogu zapisati u obliku $x \veebar (y \veebar c)$ za r, odnosno $(y \wedge c) \vee (x \wedge (y \veebar c))$ za s. ■

x	y	c	r	s
$\top$	$\top$	$\top$	$\top$	$\top$
$\top$	$\top$	$\bot$	$\bot$	$\top$
$\top$	$\bot$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\top$	$\top$	$\bot$	$\top$
$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Tabela 3.6. Istinitosna tablica punog sabirača.

3.4 Semantičke posledice

U postupku zaključivanja često je potrebno utvrditi kada je neki iskaz posledica nekog skupa iskaza, odnosno da li iskaz sledi iz nekog skupa iskaza, tj. da li je iskaz semantička posledica skupa iskaza.

Definicija 3.4.1 Formula A je *semantička posledica* skupa formula $T = \{B_1, B_2, \ldots\}$, (u oznaci $\{B_1, B_2, \ldots\} \models A$) ako je A tačno pri svakoj intepretaciji I koja zadovoljava sve formule $B_i \in T$. Formule B_i su *premise* formule A.

Često se umesto $\{B_1, B_2, \ldots\} \models A$ piše bez zagrada $B_1, B_2, \ldots \models A$, odnosno umesto $\{B_1, B_2, \ldots\} \cup \{B\} \models A$ koristimo $B_1, B_2, \ldots, B \models A$.

Primer 3.4.2 Neka su iskazi 'Dan je sunčan', 'Ako je dan sunčan, onda ne pada kiša' i 'Ne pada kiša' predstavljeni formulama, redom p, $p \to q$ i q. Neposrednom proverom pomoću iskaznih tablica, kao u tabeli 3.3, utvrđuje se da je q tačno, kad god su tačne obe formule p i $p \to q$. Dakle, q je semantička posledica skupa formula $\{p, (p \to q)\}$. ■

Veza semantičkih posledica i nekih tautologija opisana je sledećim tvrđenjem.

Teorema 3.4.3 Formula A je semantička posledica skupa formula $\{B_1, \ldots, B_n\}$ ako i samo ako je formula $(B_1 \wedge \ldots \wedge B_n) \to A$ tautologija, odnosno ako i samo ako je formula $(B_1 \wedge \ldots \wedge B_n \wedge \neg A)$ kontradikcija.

Dokaz. Ako pretpostavimo da je $B_1, \ldots, B_n \models A$, to znači da pri svakoj interpretaciji I za koju je $I(B_i) = \top$, za $i = 1, n$, važi i $I(A) = \top$, odakle direktno sledi da kad god je $I(B_1 \wedge \ldots \wedge B_n) = \top$, onda je i $I(A) = \top$, odnosno da je $I((B_1 \wedge \ldots B_n) \rightarrow A) = \top$. Ako je pak $I(B_1 \wedge B_2 \wedge \ldots B_n) = \bot$, onda je trivijalno $I((B_1 \wedge \ldots B_n) \rightarrow A) = \top$. Prema tome, važi da je $\models (B_1 \wedge \ldots B_n) \rightarrow A$. Obrnuto, pretpostavimo da je $\models (B_1 \wedge \ldots B_n) \rightarrow A$. Tada, trivijalno, za svaku interpretaciju I ako je $I(B_1 \wedge \ldots \wedge B_n) = \top$, onda je i $I(A) = \top$, tj. važi $B_1, \ldots, B_n \models A$.

Drugo tvrđenja u kome se razmatra formula $(B_1 \wedge \ldots \wedge B_n \wedge \neg A)$ je jednostavna posledica zakona o uklanjanju implikacije i De Morganovih zakona. ■

Analogno se dokazuje i sledeća teorema.

Teorema 3.4.4 (Teorema dedukcije) Za formule A i B i skup formula T važi $T, B \models A$ ako i samo ako $T \models B \rightarrow A$.

Jedna od posledica teoreme 3.4.3 je i

Teorema 3.4.5 Za svaki konačan skup formula T i svaku formulu A odlučiv je problem da li je $T \models A$.

3.5 Formalni sistemi

U dosadašnjem razmatranju iskaznih formula korišten je postupak koji se naziva semantički. Iskaznim slovima je bilo pridruženo značenje (istinitosna vrednost) i onda, na osnovu njihove istinitosne vrednosti, određivana je istinitost jedne formule, ili skupa formula. Ova tehnika se koristila i u ispitivanju da li je neka formula posledica nekih drugih formula. U matematičkoj logici su razvijeni i drugačiji, sintaksni, postupci u kome se sa simbolima radi formalno, bez razmatranja njihove istinitosti, svodeći intuiciju na najmanju meru. Takvi sistemi se nazivaju *formalni sistemi* i njima se formalizuju odgovarajuće semantičke strukture, tj. teorije, u kojima je definisan pojam istinitosti. Formalni postupci se prirodno realizuju pomoću računara, gde se na mašinskom nivou manipuliše nizovima sastavljenim od cifara 0 i 1, kojima se ne daje značenje. Sintaksni postupci predstavljaju osnovu za takozvane inteligentne programske sisteme, kao što su programski jezik Prolog, razni ekspertni sistemi itd. u kojima se izvodi automatsko dokazivanje teorema.

Primer jedne teorije koja se formalizuje je logika iskaza koju smo već upoznali. U ovom trenutku možda nije jasno zašto je potrebno pored intuitivno sasvim jasnog semantičkog postupka istinitosnih tablica, koji je u slučaju iskazne logike sasvim zadovoljio jer su problemi ispitivanja tautologičnosti, zadovoljivosti i kontradiktornosti odlučivi, uvoditi neki drugi postupak. Odgovor na to pitanje dobija se razmatranjem složenijih logika, poput predikatske logike prvog reda, u kojima istinitosne tablice ili neki drugi semantički postupci nisu uvek primenljivi ili dovoljni.

Sintaksni postupak podrazumeva postojanje nekog formalnog sistema koga određuju:

- najviše prebrojiv skup elementarnih znaka (jezik[8]) od koga se grade složene konstrukcije (formule) formalnog sistema,
- pravila za izgradnju formula,
- skup aksioma, formula od kojih se polazi u zaključivanju (u nekim sistemima ovaj skup je prazan) i
- rekurzivan skup rekurzivnih pravila izvođenja pomoću kojih se od aksioma i već izvedenih formula izvode nove formule; pravila izvođenja su u stvari n-arne relacije nad skupom formula, pa ako za neko pravilo P i formule A_1, $\ldots$, A_{n-1}, A_n važi $(A_1, \ldots, A_{n-1}, A_n) \in P$, onda kažemo da se iz formula $A_1, \ldots, A_{n-1}$ primenom pravila P izvodi formula A_n[9].

Uobičajeno je da se u formalnim teorijama koristi jezik koji sadrži znake teorije koja se formalizuje. Pravila izvođenja se mogu prikazivati na razne načine, od kojih su neki:

$$P : \frac{A_1, A_2, \ldots, A_{n-1}}{A_n},$$

P : iz skupa formula $\{A_1, A_2, \ldots, A_{n-1}\}$ izvesti formulu A_n, itd.

U formalnim sistemima se definišu pojmovi dokaza, teoreme, sintaksne posledice itd. koji su pandani semantičkih pojmova kao što su ispitivanje istinitosti, valjanost, semantičke posledice itd. u odgovarajućim strukturama koje formalizuju posmatrani sistemi. Međusobne veze sintaksnih i semantičkih pojmova se postižu interpretiranjem formalnog jezika i dokazivanjem tvrđenja o korektnosti, kompletnosti i karakterizaciji.

Definicija 3.5.1 *Dokaz*, ili izvođenje, u formalnom sistemu je konačan niz formula A_1, A_2, $\ldots$, A_m, takvih da je svako A_i bilo aksioma, bilo formula dobijena primenom nekog od pravila izvođenja na prethodne formule u dokazu. Formula A je *teorema* formalnog sistema (u oznaci $\vdash A$) ako u formalnom sistemu postoji dokaz koji se završava formulom A.

Jedna formalna teorija je opisana u sledećem primeru.

Primer 3.5.2 Posmatrajmo unarni jezik $\{1\}$, skup formula koji se sastoji od svih nepraznih reči, jednočlani skup aksioma koji sadrži formulu 1 i skup pravila izvođenja koji sadrži samo pravilo 'iz B izvesti $B11$', gde je B proizvoljna formula. Formula A oblika 11111 je teorema ovog formalnog sistema jer za nju postoji dokaz koji sadrži sledeći niz formula: 1, 111, 11111, gde je prva formula askioma, druga formula se dobija kada se na prvu formulu primeni pravilo izvođenja, a slično se primenom pravila izvođenja na drugu formulu dobija treća formula, odnosno razmatrana formula B. Ovde se zapravo može lako dokazati i sledeće tvrđenje:

[8]Primetimo da se ovde, za razliku od preostalog dela knjige, termin jezik koristi u smislu alfabet.

[9]Zbog napomene o rekurzivnosti za neku n-torku formula uvek je moguće utvrditi da li je neka formula dobijena primenom nekog od pravila izvođenja na ostale formule i odrediti koje je to pravilo.

formula A je teorema formalnog sistema ako i samo ako je čini konačni niz sastavljen od neparnog broja jedinica. U jednom smeru dokaz počiva na činjenici da je jedina aksioma niz od jednog, dakle neparnog broja, znaka 1 i da pravilo izvođenja očuvava neparnost broja znaka u izvedenim formulama. Obrnuto, ako se formula A sastoji od $2n+1$ jedinica, za neko $n \in \mathbb{N}$, dokaz za formulu A čini niz formula koji počinje aksiomom 1 za kojom sledi n formula koje se iz prethodne dobijaju primenom jedinog pravila izvođenja. ■

Definicija 3.5.3 Formula A je *sintaksna posledica* skupa formula $T = \{B_1, B_2, \ldots\}$, (u oznaci $\{B_1, B_2, \ldots\} \vdash A$) ako u formalnom sistemu postoji konačan niz formula $A_1, A_2, \ldots, A_m$, takvih da je svako A_i bilo aksioma, bilo neka od formula iz skupa T, bilo formula dobijena primenom nekog od pravila izvođenja na prethodne formule u nizu i $A = A_m$. Ovaj niz formula se naziva *dokaz za A iz skupa T*.

Kao i kod semantičkih posledica, često se piše $B_1, B_2, \ldots \vdash A$, umesto $\{B_1, B_2, \ldots\} \vdash A$ i $T, B \vdash A$, umesto $T \cup \{B\} \vdash A$.

Primer 3.5.4 Posmatrajmo formalni sistem iz primera 3.5.2 i skup formula $T = \{11\}$. Lako je proveriti da su sve formule sintaksne posledice skupa T. ■

Definicija 3.5.5 Formalni sistem je *korektan* u odnosu na teoriju koju formalizuje ako su sve teoreme formalnog sistema valjane formule. Formalni sistem je *kompletan* u odnosu na teoriju koju formalizuje ako su sve valjane formule istovremeno i teoreme formalnog sistema. Formalni sistem *karakteriše* teoriju koju formalizuje ako je korektan i kompletan u odnosu na nju.

Definicija 3.5.6 Formalni sistem je *odlučiv* ako postoji efektivan postupak (tj. algoritam) kojim se za proizvoljnu formulu može u konačnom vremenu utvrditi da li je, ili nije, teorema sistema.

Imajući u vidu definiciju 2.4.30 i Čerčovu tezu, formalni sistem je *odlučiv*, ako postoji rekurzivna funkcija koja preslikava formule (zapravo kodove formula) formalnog sistema u skup $\{0, 1\}$, tako da svakoj teoremi pridružuje 1, a svakoj formuli koja nije teorema 0, tj. ako postoji program, recimo za Tjuringovu mašinu, koji izračunava tu funkciju tako da, ako na ulazu ima formulu koja je teorema daje rezultat 1, a ako je ulazni podatak formula koja nije teorema daje rezultat 0. Postoje teorije poput predikatske logike prvog reda koje nisu odlučive, odnosno za koje svaki program koji ispituje da li je formula teorema ponekad upada u beskonačnu petlju. Sa druge strane, postoje formalne teorije, kao što je iskazni račun, koje jesu odlučive. Formalni sistem je *rekurzivno aksiomatizabilan* ako mu je skup aksioma odlučiv, tj. postoji postupak kojim se za proizvoljnu formulu utvrđuje da li, ili ne, pripada skupu aksioma. Primetimo da je kod rekurzivno aksiomatizabilnih formalnih sistema moguće kodirati i nabrojati sve dokaze, tako da se u jednoj petlji za svaki prirodan broj proverava da li je kod nekog dokaza i, ako jeste, nalazi poslednja formula u dokazu. Time se definiše parcijalno rekurzivna karakteristična funkcija predikata kojim se za proizvoljnu formulu ispituje da li je teorema formalnog sistema. Odatle je problem da li je formula teorema rekurzivno aksiomatizabilnog sistema u najgorem slučaju parcijalno odlučiv.

Za opisivanje neke formalne teorije upotrebljava se jezik i neka matematička teorija koje nazivamo *meta-jezik*, odnosno *meta-teorija*, dok se jezik formalne teorije i ona sama nazivaju *objekt-jezik*, i *objekt-teorija*. Teoreme formalnog sistema sistema (definicija 3.5.1), takozvane *objekt-teoreme*, su formule na objekt-jeziku, a tvrđenja u meta-jeziku, poput iskaza o kompletnosti, koja govore o formalnom sistemu su *meta-teoreme*. Recimo, u primeru 3.5.2, jedna objekt-teorema je 11111, dok je rečenica 'formula je teorema ako i samo ako se sastoji od neparnog broja jedinica' meta-teorema.

U nastavku ćemo proanalizirati nekoliko formalnih sistema koji se odnose na iskaznu logiku: jedan sistem nazvan hilbertovskim u čast Davida Hilberta, sistem zasnovan na rezoluciji i sistem zasnovan na tabloima.

3.6 Iskazni račun

Jedan od najpoznatijih hilbertovskih formalnih sistema je iskazni račun L, označen tako po poljskom matematičaru Lukašieviču (Jan Lukasiewicz, 1878 – 1956). Razni hilbertovski sistemi za iskaznu logiku se međusobno razlikuju u jeziku, aksiomama i pravilima izvođenja, ali su međusobno ekvivalentni u smislu da svi karakterišu istu iskaznu logiku. Iskazni račun L je određen sledećim stavkama:

- Jezik sistema se sastoji od: prebrojivo mnogo iskaznih slova p, q, r, s, p_1, q_1, ... i znaka $\neg$, $\rightarrow$ i leve i desne male zagrade,
- Formule fomalnog sistema se uvode sa:
  1. iskazna slova su atomske formule,
  2. atomske formule su formule,
  3. ako su A i B formule, onda su i $(\neg A)$ i $(A \rightarrow B)$ formule i
  4. formule se dobijaju samo konačnom primenom pravila 1, 2 i 3.
- Skup aksioma sadrži sledeće tri shema-formule:
  1. $(A \rightarrow (B \rightarrow A))$
  2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
  3. $(\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$
- Skup pravila izvođenja sadrži samo jedno pravilo, modus ponens:
  1. Iz A i $(A \rightarrow B)$ izvesti B

Shema aksiome predstavljaju prebrojivo beskonačni skup primeraka, formula dobijenih sistematskom zamenom slova A, B i C iskaznim formulama. Pošto je jezik iskaznog računa prebrojiv, a formule konačni nizovi znaka, to i svih formula, pa i aksioma, ima prebrojivo mnogo. Uobičajeno je da se zagrade ne pišu kad god to ne unosi zabunu, a i da se skraćeno piše $(A \wedge B)$ umesto $(\neg(A \rightarrow \neg B))$, $(A \vee B)$ umesto $((\neg A) \rightarrow B)$ i $(A \leftrightarrow B)$ umesto $((A \rightarrow B) \wedge (B \rightarrow A))$. Primetimo da, na osnovu rečenog u odeljku 3.2, ovakvo skraćivanje je intuitivno opravdano pošto su vrednosti odgovarajućih formula pri proizvoljnoj interpretaciji jednake.

Primer 3.6.1 Sledeći niz formula:

1.	$(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$	(primerak aks. 2)
2.	$A \to ((A \to A) \to A)$	(primerak aks. 1)
3.	$(A \to (A \to A)) \to (A \to A)$	(iz 1 i 2 po pr. 1)
4.	$A \to (A \to A)$	(primerak aks. 1)
5.	$A \to A$	(iz 3 i 4 po pr. 1)

je dokaz za formulu $A \to A$ u računu L. ■

Do kraja ovog odeljka razmotrićemo odnos iskaznog računa i iskazne logike dokazujući više meta-tvrđenja.

Teorema 3.6.2 (Korektnost) Iskazni račun je korektan u odnosu na iskaznu logiku.

Dokaz. Lako se pokazuje da su sve aksiome valjane formule i da pravilo modus ponens očuvava svojstvo 'biti valjana formula' (teorema 3.2.11), tj. da primenjeno na valjane formule daje formulu koja je takođe valjana. Prema tome, sve teoreme jesu valjane formule, pa je iskazni račun korektan u odnosu na iskaznu logiku. ■

Sledeće tvrđenje je sintaksni pandan teoreme 3.4.4.

Teorema 3.6.3 (Teorema dedukcije) Za formule A i B i skup formula T važi
$T, B \vdash A$ ako i samo ako $T \vdash B \to A$.

Dokaz. Pretpostavimo najpre da je $T, B \vdash A$ i dokažimo da je $T \vdash B \to A$. U dokazu ćemo koristiti indukciju po dužini dokaza za A. Ako je dužina dokaza 1, formula A je bilo aksioma, bilo $A \in T$, bilo $A = B$. U prva dva slučaja je $T \vdash A$. Kako je $A \to (B \to A)$ aksioma, pa i teorema, to važi $T \vdash A \to (B \to A)$. Primenom pravila modus ponens dobija se $T \vdash B \to A$. U trećem slučaju je $\vdash A \to A$, pa i $T \vdash A \to A$, tj. $T \vdash B \to A$. Pretpostavimo da je dužina dokaza $k > 1$. Formula A može ponovo biti aksioma, pripadati skupu T ili biti jednaka formuli B u kom slučaju bi kao i malopre bilo $T \vdash B \to A$. Formula A može biti dobijena i primenom pravila izvođenja na prethodne formule u dokazu. Tada su njene pretpostavke oblika C i $C \to A$. Dokazi ovih pretpostavki su kraći od k, pa je na osnovu indukcijske pretpostavke $T \vdash B \to C$ i $T \vdash B \to (C \to A)$. Kako je formula $(B \to (C \to A)) \to ((B \to C) \to (B \to A))$ aksioma, pa i teorema, dvostrukom primenom pravila modus ponens se dobija $T \vdash B \to A$.

Obrnuto, pretpostavimo da je $T \vdash B \to A$ i dokažimo da je $T, B \vdash A$. Očigledno je da važi $T, B \vdash B \to A$, jer je $T \subset T \cup \{B\}$. Takođe je i $T, B \vdash B$ tako da se primenom pravila modus ponens odmah dobija $T, B \vdash A$. ■

U sledećem primeru je prikazana jedna primena teoreme dedukcije u dokazivanju teorema iskaznog računa.

Primer 3.6.4 Dokazaćemo da je u iskaznom računu $\vdash B \to (A \to A)$. Kako je $\vdash A \to A$, to je $B \vdash A \to A$, pa po stavu dedukcije $\vdash B \to (A \to A)$. ■

Tvrđenje da iskazni račun karakteriše neku iskaznu logiku se može kratko zapisati sa

Teorema 3.6.5 (Obična kompletnost) Za svaku iskaznu formulu A važi
$$\vdash A \text{ ako i samo ako } \models A.$$

tj., formula je teorema iskaznog računa ako i samo ako je tautologija. Za ovo tvrđenje postoji direktan dokaz na osnovu koga se može konstruisati dokaz za proizvoljnu teoremu, ali ćemo ovde prikazati jedan drugi postupak koji je uz izvesne izmene pogodan za slične dokaze kod nekih drugih logika, pre svega kod predikatske logike prvog reda i modalnih logika. Najpre ćemo definisati pojam konzistentnosti skupa formula u odnosu na formalni sistem.

Definicija 3.6.6 Skup formula T je *konzistentan*[10] u odnosu na iskazni račun L ako postoji formula A tako da nije $T \vdash A$. U suprotnom, ako za svaku formulu A važi $T \vdash A$, skup je *nekonzistentan*. Formula A je konzistentna, odnosno nekonzistentna, ako je skup $\{A\}$ konzistentan, odnosno nekonzistentan. Skup formula T je *maksimalno konzistentan* ako je konzistentan i svaki konzistentni nadskup od T se poklapa sa T.

Slede dokazi više teorema iskaznog računa koje će biti korištene u dokazu kompletnosti.

Teorema 3.6.7 Za bilo koje formule A, B, C važi:

1. $A, A \to B, B \to C \vdash C$ i $A \to B, B \to C \vdash A \to C$
2. $A, \neg A \vdash B$ i $\vdash \neg A \to (A \to B)$
3. $\vdash \neg\neg A \to A$
4. $\vdash A \to \neg\neg A$
5. $\vdash (A \to B) \to (\neg B \to \neg A)$
6. $A, B \vdash A \to B$
7. $\vdash \neg(A \wedge \neg A)$

Dokaz. 1. Kako je $A, A \to B, B \to C \vdash A$ i $A, A \to B, B \to C \vdash A \to B$, pravilom modus ponens dobijamo $A, A \to B, B \to C \vdash B$. Kako je zatim $A, A \to B, B \to C \vdash B \to C$, ponovo pomoću pravila modus ponens dobijamo $A, A \to B, B \to C \vdash C$. Na osnovu teoreme dedukcije sledi $A \to B, B \to C \vdash A \to C$.

2. Prvi deo tvrđenja se dokazuje sa: (1) $\neg A, A \vdash \neg A$, (2) $\neg A, A \vdash \neg A \to (\neg B \to \neg A)$ prema aksiomi 1, (3) $\neg A, A \vdash (\neg B \to \neg A)$ (iz 1,2, modus ponens), (4) $\vdash (\neg B \to \neg A) \to (A \to B)$ prema aksiomi 3, (5) $\neg A, A \vdash A \to B$ (iz 3, 4, modus ponens) (6) $\neg A, A \vdash A$ (7) $\neg A, A \vdash B$ (iz 5 i 6, modus ponens). Drugi deo tvrđenja sledi iz (7) po teoremi dedukcije.

[10] Koriste se i termin *neprotivrečan*.

3. Dokazaćemo da je $\neg\neg A \vdash A$, odakle po teoremi dedukcije sledi $\vdash \neg\neg A \to A$: (1) $\neg\neg A \vdash \neg\neg A$, (2) $\neg\neg A \vdash \neg\neg A \to (\neg A \to \neg\neg\neg A)$ prema teoremi 3.6.7.2, (3) $\neg\neg A \vdash \neg A \to \neg\neg\neg A$ (iz 1, 2, modus ponens) (4) $\vdash (\neg A \to \neg\neg\neg A) \to (\neg\neg A \to A)$ aksioma 3, (5) $\neg\neg A \vdash \neg\neg A \to A$ (iz 3, 4, modus ponens) (6) $\neg\neg A \vdash A$ (iz 1, 5, modus ponens).

4. Prema teoremi 3.6.7.3 važi $\vdash \neg\neg\neg A \to \neg A$, a kako po aksiomi 3 imamo $\vdash (\neg\neg\neg A \to \neg A) \to (A \to \neg\neg A)$, to primenom pravila modus ponens dobijamo $\vdash A \to \neg\neg A$.

5. Najpre se pokazuje da $A \to B, \neg\neg A \vdash \neg\neg B$: (1) $A \to B, \neg\neg A \vdash A \to B$, (2) $A \to B, \neg\neg A \vdash \neg\neg A$, (3) $\vdash \neg\neg A \to A$ (prema teoremi 3.6.7.3), (4) $A \to B, \neg\neg A \vdash A$ (iz 2, 3, modus ponens), (5) $A \to B, \neg\neg A \vdash B$ (iz 1, 4, modus ponens), (6) $\vdash B \to \neg\neg B$ (prema teoremi 3.6.7.4), (7) $A \to B, \neg\neg A \vdash \neg\neg B$ (iz 5, 6, modus ponens) Odavde, prema teoremi dedukcije dobijamo $\vdash (A \to B) \to (\neg\neg A \to \neg\neg B)$. Kako je po aksiomi 3 $(\neg\neg A \to \neg\neg B) \to (\neg B \to \neg A)$ koristeći teoremu 3.6.7.1 dobijamo $\vdash (A \to B) \to (\neg B \to \neg A)$.

6. Dokaz je izvođenje: (1) $A, B \vdash B$, (2) $\vdash B \to (A \to B)$ (aksioma 1), (3) $A, B \vdash A \to B$ (iz 1, 2, modus ponens).

7. Dokaz je izvođenje: (1) $\vdash A \to \neg\neg A$ (prema teoremi 3.6.7.4), (2) $\vdash (A \to \neg\neg A) \to \neg\neg(A \to \neg\neg A)$ (prema teoremi 3.6.7.4), (3) $\vdash \neg\neg(A \to \neg\neg A)$ (iz 1, 2, modus ponens) (4) $\neg(A \wedge \neg A)$ (prema definiciji $\wedge$). ■

Teorema 3.6.8 Neka je T skup formula i $\overline{T}$ skup svih sintaksnih posledica skupa T. Ako je T konzistentan, konzistentan je i $\overline{T}$.

Dokaz. Pretpostavmo da skup $\overline{T}$ nije konzistentan. Tada je svaka formula sintaksna posledica ovog skupa, tj. za svaku formulu A postoji dokaz iz skupa $\overline{T}$ za formulu A. U tom dokazu učestvuje samo konačno mnogo formula iz $\overline{T} \setminus T$. Za te formule postoje dokazi iz T. Proširujući dokaz $\overline{T} \vdash A$ tim dokazima, dobijamo dokaz za A iz skupa formula T. Dakle, za svaku formulu A bi bilo $T \vdash A$, pa bi i T bio nekonzistentan. ■

Teorema 3.6.9 Ako je T maksimalno konzistentan skup formula i važi $T \vdash A$, tada je $A \in T$.

Dokaz. Kako je trivijalno $T \subset \overline{T}$, T maksimalan konzistentan skup i, na osnovu teoreme 3.6.8, $\overline{T}$ konzistentan, po definiciji maksimalno konzistentnog skupa sledi da je $T = \overline{T}$. ■

Prethodno tvrđenje se može formulisati i na sledeći način: maksimalno konzistentan skup je *deduktivno zatvoren*, odnosno sadrži sve svoje posledice.

Teorema 3.6.10 Skup T je nekonzistentan ako i samo ako postoji iskazno slovo p tako da je $T \vdash p \wedge \neg p$.

Dokaz. ($\Rightarrow$) Ako je T nekonzistentan, onda po definiciji $T \vdash A$, za bilo koju formulu, pa i za $p \wedge \neg p$.

($\Leftarrow$) Ako je $T \vdash p \wedge \neg p$, kako je prema teoremi 3.6.7.7 $\vdash \neg(p \wedge \neg p)$, a s obzirom da prema teoremi 3.6.7.2 važi $\vdash \neg A \to (A \to B)$ za proizvoljnu formulu B, dvosturkom primenom pravila modus ponens za $A = p \wedge \neg p$ dobili bismo da je $T \vdash B$ za svaku formulu B, pa bi T bio nekonzistentan. ■

Teorema 3.6.11 Neka je skup T maksimalno konzistentan. Tada za svaku formulu A važi $A \in T$ ako i samo ako $\neg A \notin T$, odnosno za svaku formulu A je ili $A \in T$ ili $\neg A \in T$.

Dokaz. Najpre, ne mogu biti i A i $\neg A$ u skupu T jer bi prema teoremi 3.6.7.2 za svaku formulu B bilo $T \vdash B$, tj. skup T bi bio nekonzistentan. Ako bi bilo $A \notin T$, to znači da bi skup $T \cup \{A\}$ bio nekonzistentan, pa bi prema teoremi 3.6.10 bilo $T \cup \{A\} \vdash p \wedge \neg p$ za neko iskazno slovo p. Dalje je po teoremi dedukcije $T \vdash A \rightarrow (p \wedge \neg p)$, odnosno prema teoremi 3.6.7.5 $T \vdash \neg(p \wedge \neg p) \rightarrow \neg A$. Kako je prema teoremi 3.6.7.7 $\vdash \neg(p \wedge \neg p)$, dobijamo $T \vdash \neg A$. Pošto je T deduktivno zatvoren skup, prema teoremi 3.6.9 je $\neg A \in T$. Slično bi se pokazalo i da iz $\neg A \notin T$ sledi $A \in T$. Dakle, tačno jedna od formula A i $\neg A$ je u skupu T. ■

Teorema 3.6.12 Formula $\neg A$ je konzistentna ako i samo ako nije $\vdash A$.

Dokaz. ($\Leftarrow$) Pretpostavimo da $\neg A$ nije konzistentno. Prema teoremi 3.6.10 tada postoji iskazno slovo p tako da je $\neg A \vdash p \wedge \neg p$. Prema teoremi dedukcije tada je $\vdash \neg A \rightarrow (p \wedge \neg p)$, a prema teoremama 3.6.7.5, 3.6.7.3 i 3.6.7.1 je $\vdash \neg(p \wedge \neg p) \rightarrow A$. Kako je prema teoremi 3.6.7.7 $\vdash \neg(p \wedge \neg p)$, pravilom modus ponens dobijamo $\vdash A$. Dakle, ako nije $\vdash A$, onda je $\neg A$ konzistentno.

($\Rightarrow$) Pretpostavimo da je $\vdash A$. Tada bi bilo $\neg A \vdash A$ i, prema teoremi 3.6.7.2, $\neg A \vdash A \rightarrow B$ za bilo koju formulu B. Odatle je i $\neg A \vdash B$, što bi značilo da je $\neg A$ nekonzistentno. Prema tome nije $\vdash A$. ■

Kada kažemo da je iskazni račun kompletan u odnosu na iskaznu logiku, to znači da za svaku formulu A važi da

$$\text{ako je } \models A, \text{ onda je } \vdash A,$$

odnosno

$$\text{ako nije } \vdash A, \text{ onda nije ni } \models A.$$

Prema tome, kompletnost znači da

$$\text{ako nije } \vdash A, \text{ onda je } \neg A \text{ zadovoljivo.}$$

Međutim, u teoremi 3.6.12 smo pokazali da nije $\vdash A$ ako i samo ako je $\neg A$ konzistentno, tako da je iskazni račun kompletan ako za proizvoljnu formulu A važi

$$\text{ako je } A \text{ konzistentna, onda } A \text{ ima model.}$$

U nastavku ćemo dokazati zapravo nešto jače tvrđenje, takozvanu teoremu proširene kompletnosti, u kojoj se dokazuje da svaki konzistentan skup formula ima model.

Teorema 3.6.13 (Lindenbaumova lema) Svaki konzistentan skup formula T se može proširiti do maksimalno konzistentnog skupa formula.

Dokaz. Neka je $A_0, A_1, \ldots$ jedno nabrajanje svih formula i T konzistentan skup formula. Definisaćemo niz skupova na sledeći način:

$$T_0 = T,$$

$$\text{za svaki } i > 0,\ T_{i+1} = \begin{cases} T_i & \text{ako je } T_i \cup \{A_i\} \text{ nekonzistentno} \\ T_i \cup \{A_i\} & \text{ako je } T_i \cup \{A_i\} \text{ konzistentno} \end{cases}$$

Konačno, neka je $T^* = \cup_i T_i$. Očigledno je da su svi T_i konzistentni skupovi formula i da je $T \subset T^*$.

Ako bi T^* bio nekonzistentan postojao bi konačan dokaz iz T^* za formulu $p \wedge \neg p$ za neko iskazno slovo p. Pošto je dokaz konačan niz formula, postoji maksimalan indeks tih formula u razmatranom nizu formula A_0, A_1, ... Neka je taj indeks n. Tada je $T_{n+1} \vdash p \wedge \neg p$, pa bi i skup T_{n+1} bio nekonzistentan, što je kontradikcija. Prema tome, skup T^* je konzistentan.

Ako skup T^* nije maksimalno konzistentan, postoji konzistentan skup formula S takav da je $T^* \subset S$ i $T^* \neq S$. Tada bi postojala formula $A_n \in S \setminus T^*$. Prema konstrukciji niza skupova, pošto $A_n \notin T^*$, skup $T_n \cup \{A_n\}$ je nekonzistentan, pa bi i skup S koji sadrži $T_n \cup \{A_n\}$ bio nekonzistentan. Dakle, skup T^* je i maksimalno konzistentan. ■

Teorema 3.6.14 (Proširena kompletnost) Skup formula T je konzistentan ako i samo ako je zadovoljiv.

Dokaz. ($\Leftarrow$) Neka je interpretacija I takva da je T tačan pri I. Ako bi skup T bio nekonzistentan, prema teoremi 3.6.10 postojalo bi iskazno slovo p takvo da je $T \vdash p \wedge \neg p$. Pošto su prema teoremi 3.6.2 sve teoreme valjane formule, pa i tačne pri I, i važi da ako $I(A) = \top$ i $I(A \to B) = \top$, onda je i $I(B) = \top$, to su i sve posledice skupa T tačne pri interpretaciji I. Prema tome bilo bi $I(p \wedge \neg p) = \top$, što je kontradikcija, pa nije ispravna pretpostavka o nekonzistentnosti skupa T, tj. skup T je konzistentan.

($\Rightarrow$) Obrnuto, pretpostavimo da je skup T konzistentan. Prema teoremi 3.6.13, skup T se može proširiti do maksimalno konzistentnog skupa T^*. Definišimo interpretaciju I_{T^*} tako da za svako iskazno slovo p važi

$$I_{T^*}(p) = \begin{cases} \top & \text{ako } p \in T^* \\ \bot & \text{ako } p \notin T^* \end{cases}$$

Pošto je u teoremi 3.6.11 pokazano da za svako iskazno slovo p važi ili $p \in T^*$ ili $\neg p \in T^*$, interpretacija je korektno definisana.

Sada ćemo indukcijom po složenosti formule pokazati da za svaku formulu A važi $A \in T^*$ ako i samo ako $I_{T^*}(A) = \top$. Najpre, za iskazna slova to važi po definiciji interpretacije I_{T^*}. Neka je $A = \neg B$. Formula $\neg B \in T^*$ ako i samo ako $B \notin T^*$ ako i samo ako $I_{T^*}(B) = \bot$ ako i samo ako $I_{T^*}(A) = \top$. Neka je $A = B \to C$. Ako je formula A tačna pri interpretaciji I_{T^*}, mogući su sledeći slučajevi: ili je $I_{T^*}(B) = \bot$ ili $I_{T^*}(B) = I_{T^*}(C) = \top$. Ako je $I_{T^*}(B) = \bot$, prema indukcijskom koraku je $B \notin T^*$, odnosno $\neg B \in T^*$. Kako je prema teoremi 3.6.7.2, $\vdash \neg B \to (B \to C)$ pravilom modus ponens dobijamo $T^* \vdash B \to C$, odnosno $B \to C \in T^*$, jer je skup T^* deduktivno zatvoren. Ako je $I_{T^*}(B) = I_{T^*}(C) = \top$, po indukcijskoj pretpostavci sledi da je $B \in T^*$ i $C \in T^*$, pa prema teoremi 3.6.7.6 važi $T^* \vdash B \to C$, odnosno $B \to C \in T^*$. Obrnuto, neka je $B \to C \in T^*$. Pošto je ili $B \in T^*$ ili $\neg B \in T^*$, pretpostavimo najpre da je $B \in T^*$. Kako je

$T^* \vdash B \to C$ i $T^* \vdash B$, pravilom modus ponens se dobija $T^* \vdash C$, odnosno $C \in T^*$. Prema indukcijskoj hipotezi $I_{T^*}(B) = \top$ i $I_{T^*}(C) = \top$, pa je $I_{T^*}(B \to C) = \top$. Konačno, pretpostavimo da je $\neg B \in T^*$. Prema već dokazanom, u tom slučaju je $I_{T^*}(B) = \bot$, pa je $I_{T^*}(B \to C) = \top$.

Dakle, skup T^* je tačan pri interpretaciji I_{T^*}. Odatle sledi da je i skup formula T, kao podskup skupa T^*, tačan pri interpretaciji I_{T^*}, odnosno da je skup T zadovoljiv. ■

Teoremom kompletnosti smo povezali semantički pojam valjanosti i sintaksni pojam dokazivosti. To omogućava da se u raznim dokazima slobodno mešaju semantički i sintaksni postupak. Ne primer, obrazloženje nekog koraka dokaza oblika $\vdash A$ može biti 'A je valjana formula'. Navešćemo i dve posledica teoreme o jakoj kompletnosti. Prvo tvrđenje povezuje pojmove semantičke i sintaksne posledice skupa formula, a drugo će imati značajnu primenu u odeljku 5.10.

Teorema 3.6.15 Neka su T skup formula i A formula. Tada je

$$T \models A \text{ ako i samo ako } T \vdash A.$$

Dokaz. ($\Leftarrow$) Pretpostavimo da je $T \models A$. To znači da skup formula $T \cup \{\neg A\}$ nije zadovoljiv, pa po teoremi 3.6.14 nije ni konzistentan. Tada za neko iskazno slovo p prema teoremi 3.6.10 važi $T, \neg A \vdash p \wedge \neg p$ i po teoremi dedukcije $T \vdash \neg A \to (p \wedge \neg p)$. Prema teoremama 3.6.7.5, 3.6.7.3 i 3.6.7.1 važi $T \vdash \neg(p \wedge \neg p) \to A$ i kako je prema teoremi 3.6.7.7 $\vdash \neg(p \wedge \neg p)$ imamo da je $T \vdash A$.

($\Rightarrow$) Neka je $T \vdash A$. Neka je I proizvoljna interpretacija takva da je skup T tačan pri I. Pošto su prema teoremi 3.6.2 sve teoreme valjane formule, pa i tačne pri I, i važi da ako $I(A) = \top$ i $I(A \to B) = \top$, onda je i $I(B) = \top$, to su i sve posledice skupa T tačne pri interpretaciji I. Prema tome $T \models A$. ■

Uzimajući da je skup T prazan, iz teorema 3.6.15 direktno sledi teorema obične kompletnosti 3.6.5.

Teorema 3.6.16 (Kompaktnost) Skup formula T je zadovoljiv ako i samo ako je svaki njegov konačan podskup zadovoljiv.

Dokaz. ($\Rightarrow$) Ako je T zadovoljiv, postoji interpretacija I takva da je T tačan pri I. Tada je trivijalno i svaki podskup skupa T tačan pri I.

($\Leftarrow$) Obrnuto, pretpostavimo da je svaki konačan podskup od T zadovoljiv, dok sam skup T nije zadovoljiv. Na osnovu teoreme 3.6.14, skup T nije konzistentan, pa prema teoremi 3.6.10 za neko iskazno slovo p važi $T \vdash p \wedge \neg p$. Ovaj dokaz iz T za $p \wedge \neg p$ je konačan, pa postoji neki konačan skup $T_0 \subset T$ takav da je $T_0 \vdash p \wedge \neg p$. Međutim, prema teoremi 3.6.14, tada ni skup T_0 ne bi bio zadovoljiv, što je suprotno pretpostavci. ■

3.7 Rezolucija u iskaznoj logici

Formalni sistem zasnovan na rezoluciji se unekoliko razlikuje od iskaznog računa o kome je bilo reči u odeljku 3.6. Najpre, ovde se razmatraju samo formule specijalnog oblika, takozvane klauze. Takođe, u ovom sistemu nema aksioma, već se izvođenje svodi na primenu jednog jedinog pravila.

Definicija 3.7.1 Literali L_1 i L_2 su *komplementarni*, ako je jedan od njih atomska formula, a drugi njena negacija. *Klauza*[11] je disjunkcija literala. *Prazna klauza* (u oznaci $\emptyset$) je disjunkcija nula literala, tj. ne sadrži ni jedan literal.

Pošto je klauza disjunkcija literala, neka interpretacija I zadovoljava klauzu C ako zadovoljava bar jedan njen literal, pa prazna klauza nikada nije zadovoljena. Skup klauza je zadovoljen pri interpretaciji I ako su pri I sve klauze tačne. Prazan skup klauza je uvek tačan. Dva skupa klauza S_1 i S_2 su *ekvivalentna*, u oznaci $S_1 \simeq S_2$, ako ih zadovoljavaju iste interpretacije. Ponekad će biti pogodno klauzu shvatiti kao skup literala, tako da se umesto $p \vee \neg q$ piše samo $\{p, \neg q\}$. Pri tome se podrazumeva da to nije nikakav nov pojam, već samo zgodniji zapis, kao i da je skup literala koji čine klauzu zadovoljiv ako je zadovoljiv bar jedan njegov član. *Pravilo rezolucije* je dato sledećom definicijom.

Definicija 3.7.2 Neka su C_1 i C_2 klauze i neka su L_1 i L_2 komplementarni literali, takvi da se L_1 nalazi u C_1, a L_2 u C_2. *Rezolventa* klauza C_1 i C_2 po literalima L_1 i L_2 je klauza

$$Res(C_1, C_2, L_1, L_2) = (C_1 \setminus \{L_1\}) \cup (C_2 \setminus \{L_2\})$$

dobijena tako što je iz C_1 izbrisan L_1, a L_2 iz C_2, i preostali literali povezani disjunkcijom. Klauze C_1 i C_2 su *roditelji*, a rezolventa njihov *potomak*.

Primer 3.7.3 Rezolucijom klauza $(p \vee \neg q)$ i $(q \vee r)$ dobija se klauza $(p \vee r)$. Rezolucijom klauza $(\neg p \vee q \vee r)$ i $(\neg q \vee s)$ dobija se klauza $(\neg p \vee r \vee s)$. Rezolucija se ne može primeniti na klauze $(p \vee \neg q)$ i $(\neg q \vee r)$. ■

U sledećoj teoremi ćemo pokazati da, kada skup klauza proširimo nekom rezolventom klauza iz skupa, dobijeni skup ne sadrži nove informacije u odnosu na polazni skup, zapravo da su ta dva skupa međusobno ekvivalentna. Odatle je niz primena rezolucije logički ispravan postupak zaključivanja.

Teorema 3.7.4 Neka su C_1 i C_2 dve klauze i R njihova rezolventa. Tada je $\{C_1, C_2\} \simeq \{C_1, C_2, R\}$.

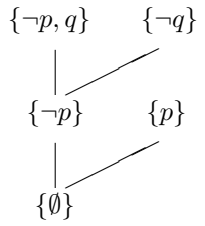
Dokaz. Neka je I interpretacija za koju je $I \models \{C_1, C_2, R\}$. Tada je trivijalno i $I \models \{C_1, C_2\}$. Obrnuto, neka je I interpretacija za koju je $I \models \{C_1, C_2\}$, tj. $I \models C_1$ i $I \models C_2$. Neka su L i $\neg L$ literali po kojima se vrši rezolucija klauza C_1 i C_2, takvi da je $L \in C_1$ i $\neg L \in C_2$. Ako $I \models L$, onda $I \models C_2 \setminus \{\neg L\}$, pa $I \models R$. Slično je i za $I \models \neg L$. Dakle, $I \models \{C_1, C_2, R\}$. ■

Definicija 3.7.5 *Dokaz* za klauzu C iz skupa klauza $\{A_1, A_2, \ldots\}$ je niz klauza $C_1, C_2, \ldots, C_m$, gde je svaka od klauza C_i bilo neka od A_i, bilo klauza dobijena primenom pravila rezolucije na neke dve prethodne klauze u dokazu i $C_m = C$.

Ako je E bilo kakav skup klauza, neka $R(E)$ označava skup svih klauza iz E i svih klauza dobijenih primenom pravila rezolucije na klauze iz skupa E. Neka je sada F skup klauza, $Res_0(F) = F$, i za $i \geq 0$, $Res_{i+1}(F) = R(Res_i(F))$. Neka $Res^*(F)$ označava uniju svih skupova $Res_i(F)$.

Formalni sistem zasnovan na pravilu rezolucije počiva na sledećem tvrđenju.

[11] Kod nas je u upotrebi i termin *sastavak*.



Slika 3.2. Rezolucijsko izvođenje iz primera 3.7.7.

Teorema 3.7.6 Skup klauza F je kontradiktoran ako i samo ako je prazna klauza u skupu $Res^*(F)$.

Dokaz. Prema teoremi 3.7.4 je $Res^*(F) \simeq F$.

($\Leftarrow$) Ako je $\emptyset \in Res^*(F)$, ovaj skup je kontradiktoran pošto ni jedna interpretacija ne zadovoljava praznu klauzu. Odatle je nezadovoljiv i skup F.

($\Rightarrow$) Ako je skup F nezadovoljiv, to je i skup $Res^*(F)$. Na osnovu teoreme 3.6.16 o kompaktnosti iz skupa $Res^*(F)$ se može izabrati konačan nezadovoljiv podskup $S = \{C_1, \ldots, C_m\}$. Označimo sa $|C|$ broj literala u klauzi C. Tvrđenje ćemo dokazati indukcijom po broju $n = |C_1| + \ldots + |C_m| - m$. Ako je $n = 0$, moguće je da su ili sve klauze jednočlane, ili da je neka klauza prazna. U drugom slučaju tvrđenje je dokazano, dok u prvom, pošto je S nezadovoljiv, moraju postojati dve klauze čiji su literali komplementarni. Primenom pravila rezolucije na ove dve klauze izvodi se prazna klauza, pa je $\emptyset \in Res^*(F)$. Neka indukcijska pretpostavka važi za svako $n < k$ i neka je $n = |C_1| + \ldots + |C_m| - m = k$. Sada bar jedna od klauza sadrži bar dva literala, neka je to C_i, ona je oblika $C_i' \cup C_i''$, gde su C_i' i C_i'' neprazni skupovi literala. Posmatrajmo skup klauza $\{C_1, \ldots, C_{i-1}, C_i', C_{i+1}, \ldots, C_m\}$. Kako je skup S nezadovoljiv, nezadovoljiv je i ovaj skup, ali je za njega $|C_1| + \ldots + |C_m| - m < k$, pa se iz ovog skupa klauza može izvesti prazna klauza. Ako u tom izvođenju nije učestvovala klauza C_i', isto važi i za polazni skup S. Ako klauza C_i' jeste učestvovala u izvođenju prazne klauze, razmotrimo izvođenje u kome se umesto C_i' koristi C_i. To bi bio dokaz za C_i''. Ponovnom primenom indukcijske pretpostavke na skup $\{C_1, \ldots, C_{i-1}, C_i'', C_{i+1}, \ldots, C_m\}$ zaključujemo da se i iz ovog skupa izvodi prazna klauza. Odatle se i iz polaznog skupa klauza S izvodi prazna klauza. ■

Primer 3.7.7 Neka je skup klauza $F = \{\{\neg p, q\}, \{\neg q\}, \{p\}\}$. Rezolventa prve dve klauze skupa je klauza $\{\neg p\}$. Rezolviranjem ove i poslednje klauze iz skupa F izvodi se prazna klauza. Prema teoremi 3.7.6 skup klauza F je kontradiktoran. ■

Rezolucijsko izvođenje se ponekad prikazuje u formi drveta. Izvođenje prazne klauze u primeru 3.7.7 je prikazano na slici 3.2.

Na prvi pogled rezolucija nije pogodna za rad sa iskaznim formulama proizvoljnog oblika. Međutim, kao što je u odeljku 3.3 rečeno, svaka iskazna formula ima ekvivalentnu formulu u konjunktivnoj normalnoj formi, odnosno definicionu formu koja je zadovoljiva ako i samo ako je zadovoljiva i sama formula. Na osnovu prethodnog tvrđenja i činjenice da je formula A zadovoljiva pri nekoj interpretaciji I ako i samo ako $\neg A$ nije zadovoljivo pri interpretaciji I, moguće je ispitivanje

valjanosti, odnosno zadovoljivosti svih formula iskazne logike. Ako se želi ispitati valjanost formule A, ona se najpre negira, potom se tako dobijena formula prevede u konjuktivnu normalnu formu ili definicionu formu i potom primenom rezolucije pokušava izvođenje prazne klauze. Ako se u tome uspe, skup klauza koji odgovara $KNF(\neg A)$, odnosno definicionoj formi formule $\neg A$, je kontradiktoran, pa je kontradiktorna i formula $\neg A$, odnosno A je valjana.

Da li je formula zadovoljiva ili kontradikcija se ispituje analognim postupkom u kome se polazi od $KNF(A)$ ili definicione forme formule A. Ako se rezolucijom dobije prazna klauza, formula A je kontradikcija, a u suprotnom je zadovoljiva.

Ako je skup klauza F konačan, nije teško proveriti da će i skup $Res^*(F)$ biti takođe konačan pa se može dobiti u konačno mnogo koraka izvođenja. Ovo je bitno, pošto garantuje da će se, ako računar ima dovoljno memorije, uvek dobiti odgovor na postavljeno pitanje (zadovoljivost, odnosno valjanost ulazne formule), što je još jedan dokaz odlučivosti tih problema za iskaznu logiku.

Primer 3.7.8 Negiranjem formule $p \rightarrow (q \rightarrow p)$ i njenim pevođenjem u konjunktivnu normalnu formu, dobija se skup klauza $F = \{p, q, \neg p\}$. Primenom pravila rezolucije na prvu i treću klauzu skupa izvodi se prazna klauza, pa je formula $p \rightarrow (q \rightarrow p)$ valjana. ■

Neka formula A je posledica skupa formula $\{B_1, \ldots, B_n\}$, ako i samo ako se iz skupa klauza koje odgovaraju formuli $(B_1 \wedge \ldots \wedge B_n \wedge \neg A)$ izvodi prazna klauza. Ovakav postupak se naziva *metoda opovrgavanja* jer se negira polazna formula $(B_1 \wedge \ldots \wedge B_n) \rightarrow A$ koja je, po teoremi 3.4.3, valjana ako i samo ako je $B_1, \ldots, B_n \models A$, pa ako je dobijena negacija kontradikcija, zaključuje se da A zaista jeste semantička posledica skupa formula $\{B_1, \ldots, B_n\}$.

Primer 3.7.9 Formula s_1 je posledica skupa formula $\{(p \wedge q) \rightarrow (r \wedge s), (p_1 \wedge q_1) \rightarrow r_1, (r_1 \wedge s) \rightarrow s_1, p, p_1, q, q_1\}$, pošto se iz skupa klauza $F = \{(\neg p \vee \neg q \vee r), (\neg p \vee \neg q \vee s), (\neg p_1 \vee \neg q_1 \vee r_1), (\neg r_1 \vee \neg s \vee s_1), p, p_1, q, q_1, \neg s_1\}$ izvodi prazna klauza. ■

3.7.1 Programska realizacija metode rezolucije

Jedna programska realizacija metode rezolucije za iskaznu logiku kojom se je ispituje valjanost iskaznih formula sprovodi se sledećom procedurom:

procedure IskaznaRezolucija
begin
1 Prevesti negaciju ulazne formule A u $KNF(\neg A)$ ili definicionu formu $D_1 \wedge D_2 \wedge \ldots \wedge D_n$.
$F := \{D_1, D_2, \ldots, D_n\}$
2 $Res_0(F) := F$
$Res_1(F) := R(Res_0(F))$
$i := 1$
3 **while** $((Res_i(F) \neq Res_{i-1}(F))$**and** $\emptyset \notin Res_i(F))$ **do**
begin
4 $i := i + 1$
$Res_i(F) := R(Res_{i-1}(F))$

```
        end
5       if (∅ ∈ Res_i(F))
        then Formula je valjana
        else Formula nije valjana
end
```

gde je R funkcija koja kao argument dobija skup klauza E, a kao rezultat vraća skup E proširen klauzama koje se dobijaju rezolucijom dve klauze iz skupa E. Svaku klauzu možemo predstaviti kao listu literala, uređenih po nekom fiksnom redolsedu, a skup klauza kao listu klauza. U jednom pozivu procedure R rezolviraju se klauze koje počinju komplementarnim literalima.

3.7.2 Komentar o metodi rezolucije

Primetimo da je pravilo rezolucije uopštenje pravila modus ponens. Naime, modus ponens se može predstaviti kao primena pravila rezolucije na klauze $\{\neg p \vee q\}$ i $\{p\}$, pri čemu se dobija klauza $\{q\}$.

Pravilo rezolucije se može objasniti i na sledeći način. Neka su klauze C_1 i C_2 redom oblika $C_1' \vee p$ i $\neg p \vee C_2'$, odnosno $\neg C_1' \rightarrow p$ i $p \rightarrow C_2'$. Na osnovu tranzitivnosti implikacije dobijamo $\neg C_1' \rightarrow C_2'$, odnosno $C_1' \vee C_2'$. Ovo je upravo klauza potomak klauza C_1 i C_2, pa je rezolucijsko pravilo primena tautologije o tranzitivnosti implikacije.

Metoda rezolucije je po svojoj prirodi mehanička, odnosno orijentisana ka računarskom izvršavanju, što se lako uočava kod primene na veće skupove formula kada se pojavljuje ogroman broj klauza. Čovek se tada teško snalazi pretražujući klauze koje treba rezolvirati. Naime, tokom rezolviranja od klauza roditelja se često dobijaju klauze koje nisu u polaznom skupu, niti su deo nekih klauza polaznog skupa, što ima za posledicu potencijalno eksplozivni rast broja klauza. Konačno, rezolucija se može elegantno primeniti samo ako je moguće proizvoljne formule prevesti u oblik konjunktivne normalne forme, što nije slučaj sa većinom logika. Pored relativno jednostavne implementacije, razlog za izučavanje metode rezolucije je pre svega u tome što predstavlja osnov mehanizma izvođenje na kome je baziran programski jezik Prolog.

3.8 Metoda analitičkih tabloa u iskaznoj logici

Za razliku od metode rezolucije, metoda analitičkih tabloa se primenjuje u većini logika koje se koriste u računarstvu. U postupku izvođenja se koriste samo podformule polaznih formula, odakle potiče pridev analitički u imenu metode. Osnovna ideja koja se koristi u metodi je sledeća. Pod pretpostavkom da imamo klasu modela kojom dajemo značenje (semantiku) formulama i neki formalni sistem koji karakteriše tu klasu modela, obično je pogodno koristiti aksiomatski sistem da bi se pokazalo da formula jeste teorema (i zbog korektnosti valjana formula), dok je konstrukcija kontramodela, koja je semantički postupak, pogodnija da se pokaže da nešto nije valjana formula, odnosno teorema. U metodi analitičkih tabloa pokušava se objedinjavanje ova dva postupka, tj. formalno i sistematski se nastoji napraviti

kontramodel za neku formulu, pa ako se u tome ne uspe, metoda garantuje da je formula valjana. Zvog ovog oslanjanja na semantiku, metoda se nekada naziva i *semantički tabloi*.

3.8.1 Ujednačavajuća notacija

Postoji više međusobno ekvivalentnih formulacija metode tabloa, a ovde ćemo prikazati opis koji koristi *ujednačavajuću notaciju*[12].

Definicija 3.8.1 Neka se iskazni jezik sastoji od prebrojivog skupa Φ iskaznih slova, logičkih veznika $\neg$, $\wedge$, $\vee$ i $\rightarrow$ i zagrada. Neka su T i F novi formalni simboli koji se ne nalaze u iskaznom jeziku. Ako je A iskazna formula, $T\ A$ i $F\ A$ su *označene formule*. *Konjugat* označene formule $T\ A$ (u oznaci $\overline{T\ A}$) je formula $F\ A$ i obrnuto $\overline{F\ A} = T\ A$.

Označene formule shvatamo na sledeći način. Za proizvoljnu interpretaciju I, $I(T\ A) = I(A)$ i $I(F\ A) = I(\neg A)$. Zapravo znak T ispred formule se može shvatiti kao 'tačno je', dok se znak F može shvatiti kao 'netačno je'.

Sve označene neatomske formule se dele u dve grupe, takozvane α i β formule. U tabeli 3.7 je prikazana podela na te dve grupe formula i njihove komponente α_1, α_2, β_1 i β_2 formule.

α	α_1	α_2
$T\ A \wedge B$	$T\ A$	$T\ B$
$F\ A \vee B$	$F\ A$	$F\ B$
$F\ A \rightarrow B$	$T\ A$	$F\ B$
$T\ \neg A$	$F\ A$	$F\ A$
$F\ \neg A$	$T\ A$	$T\ A$

β	β_1	β_2
$F\ A \wedge B$	$F\ A$	$F\ B$
$T\ A \vee B$	$T\ A$	$T\ B$
$T\ A \rightarrow B$	$F\ A$	$T\ B$

Tabela 3.7. α i β formule.

Očigledno je da se α formule ponašaju slično konjunkciji, dok se β formule ponašaju slično disjunkciji. Taj utisak je iskazan sledećim tvrđenjem.

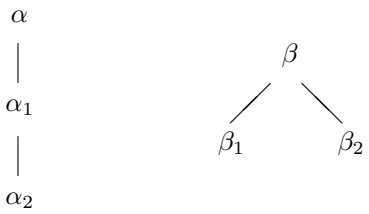
Teorema 3.8.2 Za proizvoljnu interpretaciju I važi:

- $I(\alpha) = \top$ ako i samo ako $I(\alpha_1) = I(\alpha_2) = \top$
- $I(\beta) = \top$ ako i samo ako $I(\beta_1) = \top$ ili $I(\beta_2) = \top$
- Označena formula je tačna pri interpretaciji I ako i samo ako je njen konjugat netačan.

Dokaz. Analizirajući tipove formula koje se klasifikuju kao α, odnosno β formule, lako se dokazuje traženo. Recimo, $I(T\ A \wedge B) = I(A \wedge B) = \top$ ako i samo ako $I(A) = I(B) = \top$ ako i samo ako $I(T\ A) = I(T\ B) = \top$. ■

Uloga ujednačavajuće notacije je da se mnogobrojni slučajevi složenih formula koje treba analizirati svode samo na dva, tj. na slučaj α i β formula.

[12] Unifying notation.



Slika 3.3. α i β pravilo.

3.8.2 Tablo

Analitički tablo za formulu A je uređeno binarno drvo[13] čiji čvorovi sadrže označene formule i koje se konstruiše prema sledećim pravilima:

- U korenu drveta je formula $F\ A$.
- Neka je čvor Y kraj neke grane do tog momenta konstruisanog drveta. Zavisno od formula na grani koja sadrži Y, nastavak konstrukcije je moguće izvesti na dva načina:

  α-pravilo: ako je neka α formula na grani, tada se grana produžava sa dva nova čvora od kojih jedan sadrži α_1 formulu, a drugi α_2 formulu,

  β-pravilo: ako je neka β formula na grani, tada se grana u čvoru Y grana, pri čemu levi naslednik sadrži β_1, a desni β_2 formulu.

Pravila za konstrukciju tabloa se često šematski prikazuju kao na slici 3.3. Ovde je važno obratiti pažnju na sledeće. Ako kroz jedan čvor prolazi više grana, tj. postoji grananje grane kojoj čvor pripada ispod čvora, primena pravila konstrukcije na taj čvor se odražava na sve grane kojima pripada, kao što je to slučaj sa čvorom (3) tabloa sa slike 3.5.

Za data dva uređena binarna drveta T_1 i T_2 čiji čvorovi sadrže označene formule kažemo da je T_2 *direktno proširenje* T_1 ako se T_2 dobija iz T_1 primenom jednog od α i β pravila.

Odatle je T tablo za formulu A ako postoji konačan niz drveta T_1, T_2, ..., $T_n = T$, i T_1 je drvo čiji je jedini čvor (koren) označen sa $F\ A$ i za svako $i = 1, 2, \ldots, n-1$, T_{i+1} je direktno proširenje od T_i.

Grana tabloa je *zatvorena* ako se na njoj nalazi neka formula i njen konjugat. Tablo je *zatvoren* ako mu je zatvorena svaka grana. *Dokaz* za (neoznačenu) formulu A je zatvoren tablo za A. Formula A je teorema (u oznaci $\vdash A$) ako postoji tablo koji je dokaz za nju. Grana je *završena* ako su odgovarajuća pravila konstrukcije primenjena na sve čvorove sa te grane. Tablo je *završen* ako mu je svaka grana završena ili zatvorena. Grana tabloa je *zadovoljiva* pri nekoj interpretaciji I ako

[13] *Drvo* je DAG (videti definiciju 4.3.1) u kome postoji samo jedan čvor, *koren*, sa stepenom ulaznog grananja nula, dok su svi ostali čvorovi njegovi potomci i imaju stepen ulaznog grananja jedan. Čvorovi koji imaju stepen izlaznog grananja nula se nazivaju *listovi*. *Grana* je put u drvetu koji počinje u korenu, a završava se u nekom listu. U *binarnom drvetu* čvorovi imaju najviše dve izlazne ivice, za koje se zna koja je leva, a koja desna.

(1) $F\ p \to (q \to p)$
(2) $T\ p$
(3) $F\ q \to p$
(4) $T\ q$
(5) $F\ p$
$\times$

Slika 3.4. Tablo dokaz za formulu $p \to (q \to p)$.

(1) $F\ (p \vee q) \to (p \wedge q)$
(2) $T\ p \vee q$
(3) $F\ p \wedge q$
(4) $T\ p$ (5) $T\ q$
(6) $F\ p$ (7) $F\ q$ (8) $F\ p$ (9) $F\ q$
$\times$ $\times$

Slika 3.5. Tablo za formulu $(p \vee q) \to (p \wedge q)$.

I zadovoljava sve označene formule koje se nalaze u čvorovima te grane. Tablo je *zadovoljiv* ako postoji interpertacija I takva da je bar jedna grana tabloa zadovoljiva pri I.

Primer 3.8.3 Slika 3.4 prikazuje tablo za formulu $p \to (q \to p)$. Da bi se konstrukcija učinila preglednijom, čvorovi su numerisani. Konstrukcija se obavlja na sledeći način. Najpre se u čvor (1) postavi formula sa prefiksom F. Primenom α pravila na čvor (1) dobijeni su čvorovi (2) i (3), a potom primenom istog pravila na čvor (3) dobijeni su (4) i (5). Kako (2) i (5) sadrže formule konjugate, jedina grana, pa i sam tablo su zatvoreni.

Na slici 3.5 prikazan je tablo za formulu $(p \vee q) \to (p \wedge q)$. Konstrukcija se obavlja na sledeći način. Najpre se u čvor (1) postavi formula sa prefiksom F. Primenom α pravila na čvor (1) dobijeni su čvorovi (2) i (3). Primenom pravila β na čvor (2) dolazi do grananja i dobijaju se čvorovi (4) i (5). Pravilo β se primenjuje i na čvor (3). Grananje se vrši na svim granama koje prolaze kroz ovaj čvor, tako da se u čvorovima (4) i (5) dodaju čvorovi (6), (7), (8) i (9). U ovom momentu tablo je završen, grane koje završavaju čvorovima (6) i (9) su zatvorene, ali preostale dve grane nisu, pa nije zatvoren ni celi tablo i on ne predstavlja dokaz za polaznu formulu. ■

Intuitivno, α i β pravila su konstruisana u skladu sa definicijom istinitosti formula, tako da, ako su formule koje sadrže čvorovi na grani zadovoljive, onda je to slučaj i sa granom dobijenom nakon primene α pravila, odnosno sa bar jednom granom nakon primene β pravila. U suštini, konstrukcija tabloa za neku formulu A, upravo kao što je u uvodu i rečeno, sistematski generiše sve kontramodele za A. Neuspeh u konstrukciji jednog kontramodela se ogleda zatvaranjem jedne grane, pošto skup formula sa zatvorene grane ne može biti zadovoljiv jer sadrži i neku označenu formulu i njen konjugat. Zatvaranje celog tabloa znači da A nema

kontramodel i predstavlja dokaz za A. U narednom odeljku formalizovaćemo ove konstatacije i pokazati da formula ima tablo dokaz ako i samo ako je valjana.

3.8.3 Korektnost i kompletnost metode tabloa

Teorema 3.8.4 Ako je tablo T_1 zadovoljiv i T_2 njegovo direktno proširenje, onda je i tablo T_2 zadovoljiv.

Dokaz. Neka je grana θ tabloa T_1 zadovoljiva pri intepretaciji I. Ako je T_2 dobijen pravilom koje ne menja granu θ, onda je trivijalno i T_2 zadovoljiv. Ako primena pravila menja granu θ, to može biti ili pravilo α ili β. Ako je θ proširena primenom pravila α na neku α-formulu, prema teoremi 3.8.2 je $I(\alpha) = I(\alpha_1) = I(\alpha_2) = \top$, pa su i novodobijena grana i tablo T_2 zadovoljivi. Slično je i sa pravilom β, kada su bar jedan nastavak grane θ, a time i tablo T_2, zadovoljivi. ■

Teorema 3.8.5 (Korektnost) Ako formula A ima tablo dokaz, ona je valjana.

Dokaz. Neka je tablo T dokaz za A. To znači da se u korenu tabloa nalazi formula $F\ A$ i da su sve grane tabloa T zatvorene. Očigledno je da ni jedna zatvorena grana ne može biti zadovoljiva jer sadrži formulu i njen konjugat. Kada bi formula A imala kontramodel, tj. kada bi bilo $I \models F\ A$ za neku interpretaciju I, morala bi, prema teoremi 3.8.4, biti zadovoljiva i bar jedna grana tabloa T. Kako to nije slučaj, znači da ni $F\ A$ nije zadovoljivo, tj. da je formula A valjana. ■

Teorema 3.8.6 Svaka završena grana tabloa koja nije zatvorena je zadovoljiva.

Dokaz. Neka je θ završena grana tabloa koja nije zatvorena i neka je S_θ skup formula sa te grane. On ima sledeće osobine:

1. ni za jedno iskazno slovo p nije $T\ p \in S_\theta$ i $F\ p \in S_\theta$,
2. ako je neka α-formula $\alpha \in S_\theta$, tada su i $\alpha_1, \alpha_2 \in S_\theta$ i
3. ako je neka β-formula $\beta \in S_\theta$, tada je $\beta_1 \in S_\theta$ ili $\beta_2 \in S_\theta$.

Definisaćemo interpretaciju I_θ tako da za svako iskazno slovo p bude $I_\theta(p) = \top$ ako i samo ako $T\ p \in S_\theta$. Zbog osobine 1 skupa S_θ ova definicija je korektna, pošto ni za koje iskazno slovo p ne važi $T\ p, F\ p \in S_\theta$. Indukcijom po složenosti formula se dokazuje da I_θ zadovoljava sve formule iz skupa S_θ. Za iskazna slova dokaz neposredno sledi po definiciji interpretacije I_θ. Ako se razmatra neka α-formula $\alpha \in S_\theta$, u skupu S_θ su i njene komponente α_1 i α_2 koje su manje složenosti, pa po indukcijskoj pretpostavci $I_\theta \models \alpha_1$ i $I_\theta \models \alpha_2$. Prema teoremi 3.8.2, tada je i $I_\theta \models \alpha$. Slično se dokazuje i slučaj za β-formule. ■

Teorema 3.8.7 (Kompletnost) Ako je formula A tautologija, svaki završeni tablo za A mora biti zatvoren.

Dokaz. Neka je T završen tablo za formulu A. Ako neka grana tabloa nije zatvorena, ona bi bila, prema teoremi 3.8.6, i zadovoljiva, pa bi postojala interpetacija I takva da je $I \models F\ A$, tj. $I \models \neg A$, što je kontradikcija sa pretpostavkom da je $\models A$. Dakle, svaki završeni tablo za A je dokaz za A. ■

Prema dokazanim teoremama, ako u završenom tablou za neku formulu A postoji grana θ koja nije zatvorena, formula A nije valjana, tj. ima kontramodel. Taj kontramodel se, prema dokazu teoreme 3.8.6 dobija definisanjem interpretacije I_θ za koju je $I_\theta(p) = \top$ ako i samo ako se formula $T\ p$ nalazi na grani. Dakle, pored toga što metoda tabloa za formulu A pokazuje da nije valjana, lako se konstruiše i bar neki njen kontramodel. Metoda tabloa se može koristiti i za ispitivanje da li je proizvoljna formula A zadovoljiva, tako što se u koren tabloa postavi formula $T\ A$ i dalje primenjuju ista pravila izvođenja. Ako se konstruiše završeni tablo koji nije zatvoren, kao i malope se sa grana koje nisu zatvorene nalaze modeli za A.

Primer 3.8.8 U primeru 3.8.3 je pokazano da je formula $p \to (q \to q)$ valjana, dok formula $(p \vee q) \to (p \wedge q)$ nije valjana. Jedan kontramodel za ovu formulu se konstruiše na osnovu grane koja završava čvorom (7): to je interpretacija za koju je $I(q) = \bot$ i $I(p) = \top$, pa je dalje $I(p \wedge q) = \bot$, $I(p \vee q) = \top$ i $I((p \vee q) \to (p \wedge q)) = \bot$. Drugi kontramodel se konstruiše na osnovu grane koja završava čvorom (8). To je interpretacija za koju je $I(p) = \bot$ i $I(q) = \top$. ■

Pošto svaka iskazna formula sadrži samo konačno mnogo veznika, a primenom bilo kog pravila konstrukcije dobijaju se jednostavnije formule, konstrukcija tabloa za bilo koju iskaznu formulu uvek završava u konačno mnogo koraka. Kako se u konačno mnogo koraka može proveriti i da li je tablo zatvoren, metoda tabloa predstavlja jedan postupak odlučivanja za probleme iskazne valjanosti i zadovoljivosti.

3.8.4 Programska realizacija metode tabloa

Metoda analitičkih tabloa se može isprogramirati, recimo, na sledeći način. Formula A se predstavlja u obliku binarnog drveta čiji čvorovi sadrže logičke veznike, a listovi iskazna slova formule. Tablo se konstruiše kao binarno drvo čiji čvorovi sadrže pokazivače na delove drveta formule i oznake T i F, zavisno od toga koji prefiks ima formula koja odgovara čvoru. Ako se radi sekvencijalno, konstrukcija tabloa se odvija grana po grana u skladu sa pravilima izvođenja. Posle primene svakog pravila izvođenja proverava se da li je grana zatvorena ili završena. Ako je grana zatvorena, prelazi se na sledeću granu ukoliko takva postoji. Ako je grana završena, a nije zatvorena, polazna formula nije valjana i eventualno se može prikazati njen kontramodel. Ako su sve grane zatvorene, formula je valjana. Memorijska reprezentacija svake grane podseća na stek na koji upravljački program dodaje nove čvorove, odnosno sa koga oslobađa nepotrebne čvorove prilikom prelaska na novu granu. Pogodnost ovakvog postupka je malo memorijsko zauzeće pošto nema kopiranja podformula formule A. U paralelnom izvođenju je pogodno na svaki raspoloživi procesor poslati poseban deo formule koji odgovara jednoj ili više grana, što se može uraditi prilikom primene β pravila kada se leva grana zadržava na aktuelnom procesoru, a desna šalje na neki slobodni procesor.

3.9 Genetski algoritmi za problem SAT

Kao što je u odeljku 3.2 rečeno, problem ispitivanja zadovoljivosti iskaznih formula, SAT, je prvi problem za koji je dokazana kompletnost u odnosu na NP, klasu

problema koji su rešivi nedeterminističkim mašinama u polinomijalnom vremenu. Kod SAT, kao i u slučaju drugih NP-kompletnih problema, složenost rešavanja u principu prevazilazi mogućnosti računara, tako da se pribegava heurističkim metodama koje najčeće nisu u stanju da negativno odgovore na postavljeno pitanje. To znači da, ako primerak problema koji dobijaju kao ulazni podatak nema rešenja (na primer: formula nije zadovoljiva), ove metode uglavnom ne daju odgovor. U tim situacijama se obično postavlja neki kriterijum za ograničenje dužine rada, pa ako je kriterijum ispunjen program završava rad uz poruku da nije uspeo da reši zadatak. Primetimo da to ne znači da dati problem nije rešiv, već da program nije u stanju da na to odgovori. Za metode ove vrste se zato često kaže da su polukompletne. U ovom odeljku prikazaćemo jedan noviji, heuristički, polukompletan, pristup rešavanju problema SAT koji je zasnovan na genetskim algoritmima. Veličina SAT-primeraka na kojima je testirana primena genetskih algoritama je u nekim situacijama takva da prevazilazi mogućnosti standardnih kompletnih postupaka (tablo, rezolicija), tako da je polukompletnost cena koja se plaća za efikasnost.

Pokazuje se da se bez gubitka opštosti u radu mogu koristiti samo formule date u obliku konjunktivne normalne forme, pri čemu su svi konjunkti - disjunkcije koje se sastoje od tačno tri literala. Tada se kaže da su formule u 3-cnf obliku. Eksperimentalni rezultati u dogotrajnim ispitivanjima su pokazali da najteži primeri problema SAT imaju osobinu da je broj klauza u formuli jednak $l = 4.3 * n$, gde je n broj različitih iskaznih slova u formuli, te da se testiranje kvaliteta rešenja problema SAT vrši upravo na formulama takvog oblika.

3.9.1 Genetski algoritmi

Genetski algoritmi su robusni i adaptivni metodi rešavanja problema bazirani na principima prirodne evolucije koji se primenjuju se kod široke klase optimizacionih problema. Genetski algoritmi obrađuju konačne skupove jedinki koji se nazivaju populacije. Svaka jedinka je zapis jednog potencijalnog rešenja datog problema u prostoru pretraživanja, predstavljena je genetskim kodom, a dodeljuje joj se funkcija prilagođenosti koja ocenjuje kvalitet date jedinke. Osnovni operatori genetskih algoritama su: selekcija, ukrštanje i mutacija.

Operator selekcije vrši izbor najboljih jedinki populacije, koje opstaju i prelaze u narednu generaciju. Loše jedinke, naprotiv, ne prelaze u naredne generacije već postepeno odumiru. Operator ukrštanja vrši rekombinaciju gena jedinki i time doprinosi raznovrnosti genetskog materijala. Na taj način i neki dobri geni koji pripadaju relativno lošijim jedinkama dobijaju svoje šanse za dalju reprodukciju. Posledica selekcije i ukrštanja je ukupno poboljšanje čitave populacije iz generacije u generaciju. Višestrukom primenom prethodnih operatora je moguće gubljenje dela genetskog materijala, čime neki regioni prostora pretraživanja postaju nedostupni. Primenom mutacije se vrši nasumična promena određenog gena i time se sprečava gubitak genetskog materijala. Početna populacija se generiše na slučajan način ili korištenjem neke druge heuristike.

3.9.2 Pristup problemu SAT preko genetskih algoritama

Za problem SAT je izvršena sledeća vrsta reprezentacije problema u obliku pogodnom za genetski algoritam. Jedinke u populaciji predstavljaju valuacije - potencijalne kandidate za zadovoljenje formule. Jedinke su date kao binarni nizovi dužine n, gde je n broj iskaznih slova u formuli. Binarna cifra na i-tom mestu označava vrednost i-tog iskaznog slova pri posmatranoj valuaciji. Funkcija prilagođenosti jedinke se definiše kao suma težina zadovoljenih klauza formule pri posmatranoj valuaciji. Na početku postupka sve klauze imaju jednaku jediničnu težinu, da bi tokom rada bile povećavane težine onih klauza koje najbolja jedinka ne zadovoljava. Promena težine klauza se vrši na svakih k ciklusa rada algoritma, gde je k parametar izvršavanja. Pored toga, u ispitivanju su varirani i sledeći parametri: veličina generacije, verovatnoća mutacije, tip operatora ukrštanja i tip operatora selekcije.

Kao proširenje metode genetskih algoritama korišćen je jedan oblik lokalnog pretraživanja. Na svakih m ciklusa rada osnovnog algoritma posmatraju se jedinke tekuće generacije, menjaju se redom vrednosti bitova jedinki i proverava da li su tako dobijene jedinke kvalitetnije, u kom slučaju se stare jedinke izbacuju, a nove uključuju u tekuću generaciju. Pri tome su testirane sledeće varijante lokalnog pretraživanja:

- promena se vrši na jednom slučajno izabranom bitu jedinke,
- promene se vrše redom određenim slučajnom permutacijom na svim bitovima dok god se ostvaruje poboljšanje, ili
- promene se vrše redom određenim slučajnom permutacijom na svim bitovima dok god se ostvaruje poboljšanje, s tim da se nakon jednog prolaza testiranje nastavlja dok god postoje poboljšanja.

Primena lokalnog pretraživanja je takođe testirana:

- na svim bitovima svake jedinke i
- samo na takozvanim zaleđenim bitovima, tj. bitovima koji imaju istu vrednost u svim jedinkama tekuće generacije.

3.9.3 Eksperimentalni rezultati

Opisani genetski algoritam je testiran na dve grupe problema koji sadrže zadovoljive formule. U prvoj grupi se nalazi 12 formula (po 3 formule sa redom 30, 40, 50 i 100 iskaznih slova) na kojima su sistematski podešavani parametri algoritma. U drugoj grupi formula nalazi se po 1000 formula sa primerima u kojima je broj iskaznih slova redom 20, 50 i 100 i po 100 formula sa primerima u kojima je broj iskaznih slova redom 75, 125, 150, 175, 200, 225 i 250, što sve skupa čini 4700 formula.

Formula iz prve grupe su testirane tako što se za fiksirane vrednosti spomenutih parametara i za svaku formulu algoritam izvršavao 5 puta, dok je u drugoj grupi problema većina parametara fiksirana na osnovu prethodnih rezultata, a svaka formula se izvršava dok se ne ustanovi njena zadovoljivost.

Testiranja na prvoj grupi problema su pokazala da izbor operatora ukrštanja i selekcije ne utiče značajno na rezultate. Pokazalo se da je pogodna verovatnoća mutacije bliska $1/n$, gde je n broj iskaznih slova u formuli. Smanjivanje ove verovatnoće dovodi do brzog ujednačavanja genetskog materijala zbog čega algoritam ne daje dobre rezultate. Sa druge strane, preveliko povećanje verovatnoće mutacije onemogućava konvergenciju ka rešenju. Ustanovljeno je da se pri veličini generacije od 10 jedinki postižu mnogo bolji rezultati nego ako je generacija znatno veća ili manja. Lokalno pretraživanje daje slabe rezultate ako se koristi u prvoj verziji, nešto bolje u drugoj verziji, a najbolje u trećoj verziji, dok je veliki dobitak vremena uz praktično iste rezultate ako se primenjuje samo na zaleđenim, a ne na svim bitovima. Ustanovljeno je da je 3 pogodna vrednost parametra koji se odnosi na broj koraka u kojima se primenjuje lokalno pretraživanje. Smanjivanjem ovog parametra ne postižu se značajno bolji rezultati, dok se vreme izvršavanja produžava. Samo lokalno pretraživanje bez korištenja genetskog algoritma, kao i genetski algoritam bez lokalnog pretraživanja, postizali su značajno slabije rezultate nego kada se kombinuju oba postupka. Najbolje varijante u kojima je genetski algoritam kombinovan sa lokalnim pretraživanjem u trećoj verziji nad zaleđenim bitovima uz ograničenje da je dopušteno najviše 100000 generacija u 5 pokušaja uvek su pronalazile rešenja, dok je procenat uspešnosti preko 80%, odnosno od 5 pokušaja bar 4 su dovodila do rešenja.

Testiranje u drugoj grupi problema sprovedeno je upotrebom parametara koji su u prethodnom testiranju doveli do najboljih rezultata i prihvatljivog vremena izvršavanja. Dakle, primenjuju se operatori jednopozicionog ukrštanja i turnirske selekcije, lokalno pretraživanje u trećoj verziji na svakih 3 koraka rada genetskog algoritma, dok generacije sadrže po 10 jedinki. Broj dozvoljenih generacija rada i verovatnoća mutacije su povremeno menjani tokom testiranja. Broj generacija je bio ograničen, pogotovo kod većih primera sa 100 i više iskaznih slova, na 20000, ali je u većini tih testova granica bila 2000 ili 3000. U testiranju su uspešno rešeni svi primeri. Rezultati rada pokazuju da je za rešavanje manjih primera (do 100 iskaznih slova) bilo dovoljno već dva ili tri nezavisna pokušaja. U 10 nezavisnih pokušaja rešeno je preko 80% svih primera. U slučaju formula sa 250 iskaznih slova, u 30 nezavisnih pokušaja rešeno je 88% primera, u 75 nezavisnih pokušaja rešeno je 95% primera, dok je najteži primerak rešen u 1154. pokušaju. Kao ilustraciju navedimo da je svih 1000 primeraka problema sa 20 iskaznih slova rešeno za otprilike 13 minuta na računaru baziranom na procesoru Celeron/468MHz, dok jedno izvršavanje genetskog algoritma koji kao ulaz ima formulu sa 250 iskaznih slova i radi 2000 generacija traje oko 1600 sekundi na računaru baziranom na procesoru Pentium III/800MHz.

3.10 Za dalje proučavanje

Literatura posvećena klasičnoj logici je brojna. Na primer, na našem i engleskom jeziku, mnogo širi prikazi mogu se naći u [9, 17, 71, 112, 116]. I na Internetu ima raspoloživih tekstova, recimo [76]. Ista problematika se u svojevrsnom algebarskom pristupu analizira u [127]. Metoda analitičkih tabloa opisana je u [119]. Konstrukcija definicione forme data je u [26]. Optimizacija logičkih izraza Karnuovom metodom opisana je u [38]. Kompletna analiza rezolucije sa opisima heuristika i

strategija i mnogobrojnim primerima data je u [18, 73], dok je na našem jeziku ta problematika, kao i primena raznih heurističkih postupaka u automatskom dokazivanju teorema opisana u [53]. Detaljan prikaz genetskih algoritama se može pronaći u [30, 68, 104], a primena na problem SAT je opisana u [97]. Primerci problema SAT su slobodno dostupni preko Interneta na adresama http://www.in.tu-clausthal.de/ gottlieb/benchmarks/3sat/ i http://aida.intellektik.informatik.tu-darmstadt.de/ hoos/SATLIB/benchm.html.

4

Bulove algebre

Bulove algebre (George Boole, 1815 – 1864) su često osnovno matematičko oruđe u sintezi i verifikaciji logičke strukture integrisanih elektronskih kola[1], pa ćemo u njihovom opisivanju u osnovi koristiti pristup koji imaju autori knjiga iz te oblasti. U drugom delu poglavlja razmotrićemo jednu noviju metodu za efikasno predstavljanje, minimizovanje i manipulisanje iskaznim formulama. Na samom kraju ćemo opisati jedan realan pristup sintezi i verifikaciji u kome se koriste neke od metoda koje smo prethodno opisali.

4.1 Definicija Bulovih algebri

Definicija 4.1.1 *Algebarski sistem* ili *algebra* je uređena $k+l+1$-torka $\langle A, R_1, \ldots, R_l, f_1, \ldots, f_k \rangle$, za neke l i k koji nisu oba jednaka nuli, koja se sastoji od:

- skupa nosača A,
- l relacija $R_i \subset A^{m_i}$, gde je m_i arnost relacije R_i i
- k funkcija $f_i : A^{n_i} \to A$, gde je n_i arnost funkcije f_i.

Binarna relacija $\leq$ definisana na nekom skupu A je *relacija parcijalnog poretka* ako je refleksivna ($x \leq x$), antisimetrična ($(x \leq y \wedge y \leq x) \to (x = y)$) i tranzitivna ($(x \leq y \wedge y \leq z) \to (x \leq z)$). Skup A na kome je definisana relacija parcijalnog poretka se naziva *parcijalno uređenje* ili *poset*. Primeri poseta su skup realnih brojeva sa operacijom manje ili jednako, partitivni skup nekog skupa i relacija biti podskup itd. Primetimo da ne moraju biti uporediva svaka dva elementa poseta.

Neka je $\langle A, \leq \rangle$ poset. Element $m \in A$ je *donja granica* za $a, b \in A$ ako je $m \leq a$, $m \leq b$. *Najveća donja granica* za $a, b \in A$ je $m \in A$ ako je m donja granica za a i b i za svaku donju granicu n elemenata a i b važi $n \leq m$. Najveća donja granica elemenata a i b se označava se $a \cdot b$.[2] *Gornja granica* i *najmanja gornja granica*[3] se definišu analogno za relaciju $\geq$ ($a \geq b$ ako i samo ako $b \leq a$). Oznaka za najmanju gornju granicu je $a + b$. Lako se vidi da su, ako postoje, $a \cdot b$ i $a + b$ jedinstveni.

[1] Čipova.

[2] Najveća donja granica se naziva i *meet* elemenata a i b.

[3] Najmanja gornja granica se naziva i *join* elemenata a i b.

Primer 4.1.2 Neka su skupovi $A = \{a, b, c, d\}$ i $I = \{(a, a), (b, b), (c, c), (d, d)\}$ i relacija parcijalnog poretka definisana sa $\leq = \{(a, b), (a, c), (a, d), (b, d)\} \cup I$. Tada je $a \leq b$ i $a \leq c$, pa je a donja granica za b i c. Kako je a jedini takav element, $a = b \cdot c$. Međutim, b i c nemaju ni jednu gornju granicu, pa $b + c$ nije definisan. Sa druge strane i b i d su gornje granice za a i b, ali je $b \leq d$, pa je $a + b = b$.

Neka su skup $A = \{a, b, c, d\}$ i relacija poretka definisana sa $\leq = \{(a, b), (a, d), (c, b), (c, d)\} \cup I$. Tada je $a \leq b$ i $a \leq d$, ali i $c \leq b$, $c \leq d$. I a i c su donje granice za b i d, ali kako nisu uporedivi, ne postoj $b \cdot d$. Međutim, $a + b = b$, $a + d = d$, $c + b = b$ i $c + d = d$. Sa druge strane, pošto b i d nisu uporedivi, nije definisano $a + c$. ■

Osnovna tvrđenja o $x \cdot y$ i $x + y$ su iskazana sledećom teoremom.

Teorema 4.1.3 Ako postoje $x \cdot y$ i $x + y$, važi $x \geq (x \cdot y)$ i $x \leq (x + y)$. U svakom posetu je $x \leq y$ ako i samo ako važi $x \cdot y = x$, odnosno $x + y = y$.

Dokaz. Prvo tvrđenje je direktan zapis definicije $x \cdot y$ i $x + y$ koji su donja, odnosno gornja granica za x i y. U drugom tvrđenju, za smer ($\Rightarrow$) pretprostavimo da je $x \leq y$, pa je x jedna donja granica za x i y. Ako je $m = x \cdot y$, onda je $m \leq x$ i $m \leq y$. Kako je m najveća donja granica, onda je i $x \leq m$, pa je $m = x$. Slično se dokazuje i $x + y = y$. U suprotnom smeru dokaz je trivijalan jer $x = x \cdot y$ upravo znači $x \leq y$. ■

Ako su uporediva svaka dva elementa skupa A, za neki poset $\langle A, \leq \rangle$, skup A je *totalno uređen*. Primer ovakvog skupa je skup realnih brojeva u odnosu na relaciju poretka. Ako svaki neprazan podskup totalno uređenog skupa ima najmanji element, skup je *dobro uređen*. Primer ovakvog skupa je skup prirodnih brojeva, dok skup realnih brojeva nije dobro uređen.

Mreža[4] je poset $\langle A, \leq \rangle$ u kome svaka dva elementa x i y imaju $x \cdot y$ i $x + y$. Sada se $x \cdot y$ i $x + y$ mogu shvatiti i kao binarne operacije. Lako se vidi da u svakoj mreži čiji je nosač konačan neprazan skup postoje najveći i najmanji element koji se označavaju redom sa 1 i 0. Ovi elementi postoje i u nekim mrežama koje nisu konačne. Ako za neki element x mreže postoji element y tako da je $x + y = 1$ i $x \cdot y = 0$, onda je element y *komplement* ili *inverz* elementa x. Mreža je *komplementirana* ako svaki element ima inverz. Mreža je *distributivna* ako za sve x, y i z važe jednakosti:

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z) \qquad \text{i} \qquad x + (y \cdot z) = (x + y) \cdot (x + z)$$

Definicija 4.1.4 *Bulova algebra* je komplementirana distributivna mreža.

U tabeli 4.1 su opisane osobine koje važe za sve Bulove algebre. Može se pokazati i da je Bulova algebra svaki algebarski sistem $\langle A, +, \cdot, ', 1, 0 \rangle$ koji zadovoljava osobine navedene u tabeli 4.1 u kom slučaju se relacija parcijalnog poretka $\leq$ definiše sa $x \leq y$ ako $x + y = y$, odnosno ako $x \cdot y = x$. Među najpoznatije Bulove algebre spadaju:

- *iskazna algebra* $BA_2 = \langle \{\top, \bot\}, \vee, \wedge, \neg, \top, \bot \rangle$[5],

[4] Lattice.

[5] Uočiti paralelu između osobina iskazne algebre datih u tabeli 4.1 i tautologija datih u tabeli 3.4. Uopšte, videti teoremu 4.1.7.

Idempotencija	$x \cdot x = x$	$x + x = x$
Komutativnost	$x \cdot y = y \cdot x$	$x + y = y + x$
Asocijativnost	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	$x + (y + z) = (x + y) + z$
Absorpcija	$x \cdot (x + y) = x$	$x + (x \cdot y) = x$
Distributivnost	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	$x + (y \cdot z) = (x + y) \cdot (x + z)$
Egzistencija inverza, tj. za svaki x postoji x' tako da je $x + x' = 1$ i $x \cdot x' = 0$.		

Tabela 4.1. Osobine Bulovih algebri.

- *algebra skupova* $\langle \mathbb{P}(X), \cup, \cap, \mathbb{C}_X, X, \emptyset \rangle$, gde su X neprazan skup, $\mathbb{P}(X)$ partitivni skup skupa X i $\mathbb{C}_X$ komplement u odnosu na skup X,
- *intervalna algebra* u kojoj su elementi skupa nosača konačne unije poluotvorenih intervala oblika $[a, b) \subset [0, +\infty)$, a operacije su unija, presek i komplement u odnosu na $[0, +\infty)$, $1 = [0, +\infty)$ i $0 = [0, 0)$ i
- Lindenbaum-Tarski algebra u kojoj je skup nosač skup svih klasa ekvivalencije skupa iskaznih formula u odnosu na relaciju $A \sim B$ definisanu sa $\vdash A \leftrightarrow B$, a operacije su $[A] \vee [B] = [A \vee B]$, $[A] \wedge [B] = [A \wedge B]$, $\neg[A] = [\neg A]$, $1 = [A \vee \neg A]$ i $0 = [A \wedge \neg A]$.

Poznati rezultat

Teorema 4.1.5 (Stonova teorema o reprezentaciji) Svaka Bulova algebra je izomorfna nekoj algebri skupova.

govori da je jedina suštinska razlika između Bulovih algebri kardinalnost skupa nosača koji, opet, kod konačnih algebri mora biti stepen broja 2 zbog kardinalnosti partitivnog skupa konačnog skupa. Teoreme 4.1.6 i 4.1.7 ističu poseban značaj Bulove algebre BA_2.

Teorema 4.1.6 Jednakost $t_1 = t_2$ važi u svim Bulovim algebrama ako i samo ako važi u Bulovoj algebri BA_2.

Teorema 4.1.7 Iskazna formula A na jeziku $\{\neg, \wedge, \vee\}$ je tautologija ako i samo ako u Bulovoj algebri BA_2 važi jednakost $A = 1$, pri čemu se iskazna slova shvataju kao promenljive.

Za svaki iskaz α na jeziku neke Bulove algebre definiše se *dualni iskaz* $d\alpha$ koji se od iskaza α dobija sistematskom zamenom u α simbola $+$ i $\cdot$, odnosno 1 i 0.

Teorema 4.1.8 (Princip dualnosti) Iskaz α važi u svim Bulovim algebrama ako i samo ako u svim Bulovim algebrama važi i njemu dualni iskaz $d\alpha$.

Primer 4.1.9 U svakoj Bulovoj algebri važi $x + (x' \cdot y) = x + y$ jer $x + y = x + (x + x') \cdot y = x + (x \cdot y + x' \cdot y) = (x + x \cdot y) + x' \cdot y = x + x' \cdot y$. Prema principu dualnosti, u svakoj Bulovoj algebri važi $x \cdot (x' + y) = xy$. ■

U sledećoj teoremi sumiraćemo više osobina Bulovih algebri koje se koriste u skraćivanju izraza, što je od velikog značaja u projektovanju čipova.

Teorema 4.1.10 U svim Bulovim algebrama važi sledeće:

1. $x \cdot 0 = 0$, $x \cdot 1 = x$,
2. ako je $x \cdot y = 0$ i $x + y = 1$, onda je $y = x'$,
3. involucija, $(x')' = x$,
4. $x + (x' \cdot y) = x + y$,
5. De Morganov zakon, $(x + y)' = x' \cdot y'$,
6. konsenzus, $x \cdot y + x' \cdot z + y \cdot z = x \cdot y + x' \cdot z$ i
7. svi izrazi dualni prethodnim.

4.2 Bulove funkcije

Definicija 4.2.1 (Bulove funkcije) Za Bulovu algebru $\langle B, +, \cdot, ', 1, 0\rangle$ *klasa Bulovih funkcija sa n promenljivih* sadrži osnovne funkcije:

1. konstantne funkcije $f(x_1, \ldots, x_n) = b$ za svaki $b \in B$ i
2. funkcije projekcije $f(x_1, \ldots, x_n) = x_i$

i sve funkcije koje se od njih dobijaju konačnom primenom pravila:

1. $(f + g)(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) + g(x_1, \ldots, x_n)$,
2. $(f \cdot g)(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) \cdot g(x_1, \ldots, x_n)$ i
3. $(f')(x_1, \ldots, x_n) = (f(x_1, \ldots, x_n))'$

na već definisane funkcije f i g.

Ako je $f(x_1, x_2, \ldots, x_n)$ Bulova funkcija, označićemo $f_{x_1'}(x_2, \ldots, x_n) = f(0, x_2, \ldots, x_n)$ i $f_{x_1}(x_2, \ldots, x_n) = f(1, x_2, \ldots, x_n)$. Funkcije $f_{x_1'}(x_2, \ldots, x_n)$ i $f_{x_1}(x_2, \ldots, x_n)$ se nazivaju *kofaktori* funkcije $f(x_1, x_2, \ldots, x_n)$ u odnosu na x_1.

Teorema 4.2.2 (Bul-Šenonova (Shannon) teorema ekspanzije) Za svaku Bulovu funkciju $f : B^n \to B$ i sve $(x_1, \ldots, x_n) \in B$ je

$$f(x_1, x_2, \ldots, x_n) = x_1' \cdot f(0, x_2, \ldots, x_n) + x_1 \cdot f(1, x_2, \ldots, x_n)$$

i dualno

$$f(x_1, x_2, \ldots, x_n) = (x_1' + f(1, x_2, \ldots, x_n)) \cdot (x_1 + f(0, x_2, \ldots, x_n)).$$

Dokaz. Dokaz jednakosti se sprovodi indukcijom po složenosti funkcija. Za konstantne funkcije i funkcije projekcije tvrđenje trivijalno važi. U indukcijskom koraku se pretpostavi da tvrđenje važi za funkcije f i g i lako proverava da važi za funkcije koje se od njih dobijaju. ■

Primer 4.2.3 Primenom teoreme ekspanzije 4.2.2 na funkciju $f = x_1 \cdot x_2 \cdot x_3' + x_2 \cdot (x_1' + x_3)$ dobija se $f = x_1' \cdot x_2 + x_1 \cdot (x_2 \cdot x_3' + x_2 \cdot x_3) = (x_1' + x_1) \cdot x_2 = x_2$, odnosno dualnom varijantom teoreme $f = (x_1' + (x_2 \cdot x_3' + x_2 \cdot x_3)) \cdot (x_1 + x_2) = (x_1' + x_2) \cdot (x_1 + x_2) = x_2$, pri čemu je $f(0, x_2, \ldots, x_n) = x_2 \cdot (1 + x_3) = x_2$ i $f(1, x_2, \ldots, x_n) = (x_2 \cdot x_3' + x_2 \cdot x_3) = x_2$. Slično, primenom teoreme ekspanzije se može pokazati da je $f(x + y) \cdot f(x' + y) = x' \cdot (f(y) \cdot f(1)) + x \cdot (f(1) \cdot f(y)) = (x' + x) \cdot (f(y) \cdot f(1)) = f(1) \cdot f(y)$. ■

Razvijanje izraza koje se koristi u teoremi ekspanzije 4.2.2 može se nastaviti tako da se dobijaju *kanonske forme*[6]

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) &= x_1' \cdot \ldots \cdot x_{n-1}' \cdot x_n' \cdot f(0, \ldots, 0, 0) + \\ &\quad x_1' \cdot \ldots \cdot x_{n-1}' \cdot x_n \cdot f(0, \ldots, 0, 1) + \ldots + \\ &\quad x_1 \cdot \ldots \cdot x_{n-1} \cdot x_n \cdot f(1, \ldots, 1, 1) \end{aligned}$$

odnosno

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) &= (x_1 + \ldots + x_{n-1} + x_n + f(0, \ldots, 0, 0)) \cdot \\ &\quad (x_1 + \ldots + x_{n-1} + x_n' + f(0, \ldots, 0, 1)) \cdot \ldots \cdot \\ &\quad (x_1' + \ldots + x_{n-1}' + x_n' + f(1, \ldots, 1, 1)) \end{aligned}$$

koje su jedinstvene za svaku funkciju. Prva forma se naziva *kanonska disjunktivna forma* ili *kanonska minterm forma*, a druga *kanonska konjunktivna forma* ili *kanonska maksterm forma*. Izrazi $f(0, \ldots, 0, 0)$, $f(0, \ldots, 0, 1)$, $\ldots$, $f(1, \ldots, 1, 1)$ se nazivaju *diskriminante*. Izrazi $x_1' \cdot \ldots \cdot x_{n-1}' \cdot x_n'$, $x_1' \cdot \ldots \cdot x_{n-1}' \cdot x_n$, $x_1 \cdot \ldots \cdot x_{n-1} \cdot x_n$ se nazivaju *mintermi*, a izrazi $x_1' + \ldots + x_{n-1}' + x_n'$, $x_1' + \ldots + x_{n-1}' + x_n$, $x_1 + \ldots + x_{n-1} + x_n$ se nazivaju *makstermi*.

Primer 4.2.4 Posmatrajmo funkciju $f = x_1 \cdot x_2 + a \cdot x_2'$ nad Bulovom algebrom sa skupom nosačem $B = \{0, a, b, 1\}$. Njena disjunktivna kanonska forma je $a \cdot x_1' \cdot x_2' + 0 \cdot x_1' \cdot x_2 + a \cdot x_1 \cdot x_2' + 1 \cdot x_1 \cdot x_2$, dok je konjunktivna kanonska forma $(a + x_1' + x_2') \cdot (0 + x_1' + x_2) \cdot (a + x_1 + x_2') \cdot (1 + x_1 + x_2)$. ■

Očigledno, svaka Bulova funkcija je potpuno okarakterisana diskriminantama, odnosno vrednostima u ovim specijalnim tačkama, bez obzira na skup nosač B koji ne mora biti $\{0, 1\}$.

Jedna od posledica ovog razvoja je da se može izračunati broj različitih n-arnih Bulovih funkcija za neku Bulovu algebru B. Naime, broj preslikavanja n-torki nula i jedinica (kojih ima 2^n) u skup kardinalnosti $|B|$ je $|B|^{2^n}$, pa je i broj Bulovih funkcija takođe $|B|^{2^n}$. Kako svih n-arnih funkcija za Bulovu algebru kardinalnosti $|B|$ ima $|B|^{|B|^n}$ vidi se da nisu sve funkcije $f : B^n \to B$ Bulove u smislu definicije 4.2.1. Te funkcije se nazivaju *pseudo-Bulove funkcije*. Iako sve funkcije $f : B^n \to B$ nisu Bulove, ovih ima veoma mnogo, čak i za male dimenzije skupa B. Na primer, razmotrimo Bulovu algebru sa 4 elementa i Bulove funkcije arnosti 4. Ukupan broj takvih Bulovih funkcija iznosi $|B|^{2^4} = 4^{2^4} = 2^{32}$.

[6] Primetimo potpunu paralelu sa tabličnom metodom formiranja potpunih formi opisanom u odeljku 3.3.2.

Zadatak dizajniranja čipova je često povezan sa nalaženjem najbolje funkcije iz nekog obimnog skupa funkcija za neku nekompletno datu specifikaciju u kojoj, iz nekih razloga, nisu definisane vrednosti funkcije za neke ulazne n-torke[7].

U deficiji 4.2.1 uvedene su tri operacije nad Bulovim funkcijama. Ako posmatramo skup svih n-arnih Bulovih funkcija nad nekom fiksiranom Bulovom algebrom zajedno sa tim operacijama, pri čemu označimo sa $1(x_1, \ldots, x_n) = 1$ i $0(x_1, \ldots, x_n) = 0$ i ova algebra funkcija je Bulova algebra. Relacija $\leq$ se definiše sa $f \leq g$ ako za sve n-torke elemenata $(a_1, \ldots, a_n) \in B$, $f(a_1, \ldots, a_n) \leq g(a_1, \ldots, a_n)$. Najčešće se nekompletno specifikovane funkcije zadaju pripadnošću nekom intervalu oblika $\{f : a \leq f \text{ i } f \leq b\}$ u Bulovoj algebri funkcija.

4.3 Minimizacija Bulovih funkcija i BDD

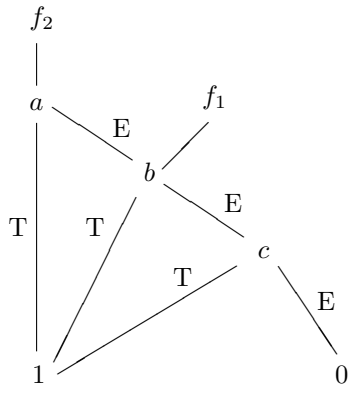
Mogućnost efikasnog predstavljanja Bulovih funkcija je značajna osobina koja se često koristi u realnim primenama Bulovih algebri jer donosi uštede u materijalu i/ili vremenu izračunavanja. U ovom odeljku ćemo opisati postupak baziran na korištenju *binarnih diagrama odlučivanja*[8] koji se skraćeno označavaju sa BDD. BDD poseduje osobinu *kanoničnosti* prema kojoj su Bulove funkcije jednake ako i samo ako su jednake njima odgovarajuće BDD-reprezentacije.

Definicija 4.3.1 *Direktan graf* je uređeni par $G = \langle V_G, E_G \rangle$, gde je V_G skup *čvorova*[9] i E_G skup *ivica*[10] za koji važi $E_G \subset V_G^2$, tj. svaka ivica $e \in E_G$ je uređeni par čvorova $u, v \in V_G$, tj. $e = (u, v)$, pri čemu je u *čvor repa*[11], a v *čvor glave*[12]. *Put* je niz ivica e_1, e_2, ..., takvih da je za svako i čvor glave ivice i čvor repa ivice $i+1$. *Ciklus* je zatvoreni put koji se završava istim čvorom kojim i počinje. DAG[13] je direktan graf u kojem nema ciklusa. *Stepen izlaznog grananja čvora* $v \in V_G$ je broj ivica $e \in E_G$ za koje je v čvor repa. *Stepen ulaznog grananja čvora* $v \in V_G$ je broj ivica $e \in E_G$ za koje je v čvor glave. *Polazni čvor* je čvor $v \in V_G$ za koji je stepen ulaznog grananja 0. *Završni čvor* je čvor $v \in V_G$ za koji je stepen izlaznog grananja 0. *Unutrašnji čvor* je čvor $v \in V_G$ za koji su i stepen ulaznog grananja i stepen izlaznog grananja veći od 0. Čvor v je *potomak*[14] čvora u ako postoji put čija početna ivica ima u za čvor repa, a završna ivica v za čvor glave.

Kod *nedirektnog grafa* ivice su oblika $e = \{u, v\}$, tj. skupovi, a ne uređeni parovi.

U ovom poglavlju će na većem broju slika biti predstavljeni direktni grafovi. Pri tome će biti pretpostavljeno da su ivice usmerene na dole, odnosno na desno ako su ivice horizontalne. Zbog toga na slikama neće biti ivica sa strelicama koje označavaju usmerenje, kako se ponegde radi.

[7] Recimo analizom okruženja se zaključuje da u fizičkoj realizaciji neke Bulove funkcije nije moguće da se na ulazu pojave neke vrednosti.
[8] Binary Decision Diagrams.
[9] Vertices, nodes.
[10] Edges, arcs.
[11] Tail vertex.
[12] Head vertex.
[13] Direktan acikličan graf. Direct acyclic graph.
[14] Naslednik.



Slika 4.1. BDD za funkciju $F(a, b, c) = (b + c, a + b + c)$.

Definicija 4.3.2 BDD za funkciju F sa više izlaza je DAG $\langle V \cup \Phi \cup \{0, 1\}, E\rangle$. Skup čvorova je podeljen u tri podskupa:

- V je skup unutrašnjih čvorova takav da svaki $v \in V$ ima stepen izlaznog grananja 2 i oznaku $l(v) \in S_F$, gde je $S_F = \{x_1, \ldots, x_n\}$ skup promenljivih od kojih zavisi funkcija F,
- $\{0, 1\}$ je skup završnih čvorova i
- Φ, skup funkcijskih čvorova, je skup polaznih čvorova.

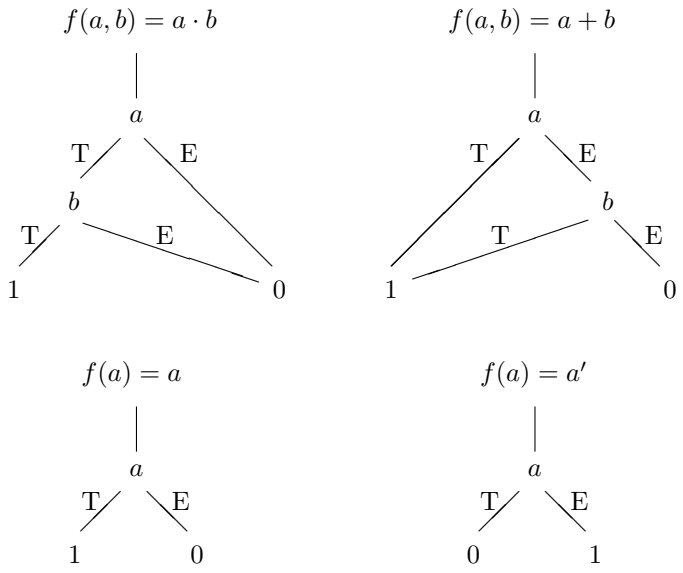
Funkcijski čvorovi predstavljaju komponente funkcije F. Za svaki čvor $v \in V$ izlazne ivice su označene sa T i E[15]. Unutrašnji čvor v se opisuje sa $(l(v), T, E)$. Promenljive iz S_F su uređene tako da, ako je v_j naslednik od v_i, onda kažemo da je $l(v_i) < l(v_j)$.

BDD reprezentuje Bulovu funkciju F sa jednim ili više izlaza. Bulova funkcija sa više izlaza u stvari predstavlja složeno logičko kolo u kome se nad istim skupom promenljivih istovremeno računa više funkcija sa po jednim izlazom. Da bi se efikasnije realizovali, zajednički podizrazi tih funkcija fizički se postavljaju na isto mesto na čipu. Na slici 4.1 je prikazan jedan BDD koji odgovara funkciji $F(a, b, c) = (f_1, f_2) = (b + c, a + b + c)$ sa dva izlaza.

Definicija 4.3.3 Funkcija F koju reprezentuje neki BDD definiše se induktivno:

1. Funkcija završnog čvora označenog sa 1 je konstantna funkcija 1.
2. Funkcija završnog čvora označenog sa 0 je konstantna funkcija 0.
3. Funkcija ivice je funkcija čvora glave.

[15]Oznake su skraćenice od *then* i *else* i odgovaraju redom vrednostima 1 i 0 promenljive iz čvora repa.



Slika 4.2. BDD reprezentacije za neke tipične Bulove funkcije.

4. Funkcija čvora $v \in V$ je data sa $l(v)f_T + l(v)'f_E$ gde su f_T i f_E redom funkcije izlaznih ivica T i E čvora v.

5. Funkcija funkcijskog čvora $\phi \in \Phi$ je funkcija njegove izlazne ivice.

Na slici 4.2 su prikazane BDD-reprezentacije nekih tipičnih funkcija.

Primer 4.3.4 Na primer, izračunajmo funkciju čiji je BDD označen sa $a + b$ na slici 4.2. Najpre je $f(1) = 1$ i $f(0) = 0$. Zatim je za čvor označen sa b, $f_T = 1$ i $f_E = 0$, pa je $f(b) = b \cdot 1 + b' \cdot 0 = b$. Za čvor a je $f_T = 1$ i $f_E = a' \cdot b$, pa je $f(a) = a \cdot 1 + a' \cdot b = a + b$. Konačno, za čvor $f(a,b)$ funkcija izlazne ivice, a time i samog funkcijskog čvora, jednaka je $a + b$. ■

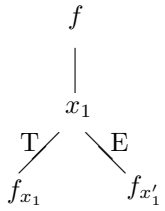
4.3.1 Konstrukcija BDD-reprezentacije Bulovih funkcija

Sledećim postupkom u kome se koristi teorema ekspanzije 4.2.2 polazeći od analitičkog opisa funkcije dolazi se do njene BDD-reprezentacije:

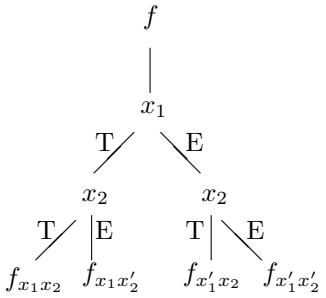
1. Uoči se jedan redosled promenljivih koje se javljaju u funkciji f: $x_1 < x_2 < \ldots < x_n$.

2. Za prvu promenljiva u nizu se izračunaju njeni kofaktori f_{x_1} i $f_{x_1'}$ i konstruiše parcijalni BDD kao na slici 4.3.

3. Računaju se kofaktori $(f_{x_1})_{x_2}$, $(f_{x_1})_{x_2'}$ i $(f_{x_1'})_{x_2}$, $(f_{x_1'})_{x_2'}$ u odnosu na sledeću promenljivu x_2 iz niza i konstruiše se parcijalni BDD kao na slici 4.4.



Slika 4.3. Parcijalni BDD nakon primene ekspanzije u odnosu na x_1.



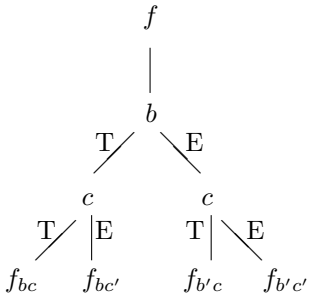
Slika 4.4. Parcijalni BDD nakon primene ekspanzije u odnosu na x_2.

4. Prethodni korak se ponavlja dok god se ne dođe do konstantnih kofaktora 1 i 0 ili se ne iscrpe sve promenljive, pri čemu se koristi da je $(x_i)_{x_i} = 1$, a $(x_i)_{x_i'} = 0$ i $(x_i)_{x_j} = (x_i)_{x_j'} = x_i$, za $i \neq j$.

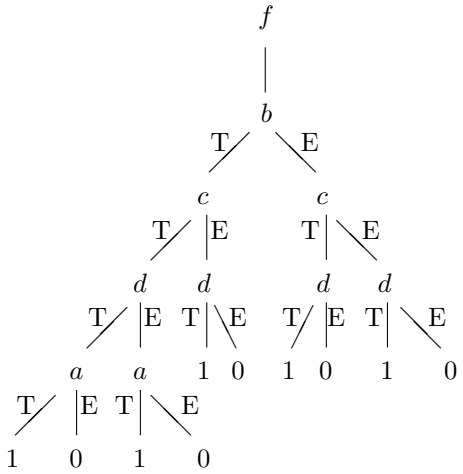
Primer 4.3.5 Neka je $f(a,b,c,d) = a \cdot b \cdot c + b' \cdot d + c' \cdot d$ funkcija za koju treba konstruisati BDD i neka je izabrani redosled promenljivih $b < c < d < a$. U prvom koraku izračunavaju se kofaktori od f u odnosu na b, $f_b = f(a,1,c,d)_b = a \cdot c + c' \cdot d$ i $f(a,0,c,d) = f_{b'} = d + c' \cdot d$. Zatim se izračunavaju $f_{b,c} = a$ i $f_{b,c'} = f_{b',c} = f_{b',c'} = d$. Sada parcijalni BDD izgleda kao na slici 4.5. Kofaktori u odnosu na d su $f_{b,c,d} = f_{b,c,d'} = a$, $f_{b,c',d} = f_{b',c,d} = f_{b',c',d} = 1$ i $f_{b,c',d'} = f_{b',c,d'} = f_{b',c',d'} = 0$. Poslednjih šest kofaktora su konstantne funkcije i na njima sa postupak prekida. Na prva dva kofaktora se primenjuje jedina preostala promenljiva i dobija $f_{b,c,d,a} = f_{b,c,d',a} = 1$, odnosno $f_{b,c,d,a'} = f_{b,c,d',a'} = 0$. Završni BDD za funkciju f izgleda kao na slici 4.6. ■

4.3.2 Redukovani BDD

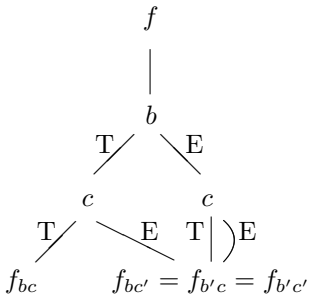
U primeru 4.3.5 se lako uočava da su neki kofaktori koji se izračunavaju tokom konstrukcije BDD-reprezentacije funkcije međusobno jednaki. Na primer, $f_{b,c'} = f_{b',c} = f_{b',c'} = d$. U takvim slučajevima se može kreirati samo jedan čvor što će optimizovati veličinu BDD-reprezentacije. Ova situacija je prikazana na slici



Slika 4.5. Parcijalni BDD za funkciju $a \cdot b \cdot c + b' \cdot d + c' \cdot d$.



Slika 4.6. Završeni BDD za funkciju $a \cdot b \cdot c + b' \cdot d + c' \cdot d$.



Slika 4.7. Optimizovani BDD.

4.7. Sledećom definicijom se formalizuje šta se podrazumeva pod minimalnom reprezentacijom Bulove funkcije.

Definicija 4.3.6 BDD je *redukovan* ako je ispunjeno:

1. Svaki unutrašnji čvor $v \in V$ je potomak nekog funkcijskog čvora $f \in \Phi$.
2. U grafu ne postoje izomorfni podgrafovi.
3. Za svaki čvor v i njegove izlazne ivice T i E je $f_T \neq f_E$.

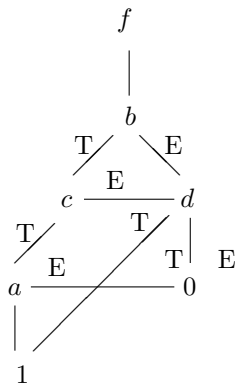
Uslovi 2 i 3 garantuju da je redukovani BDD jedna vrsta minimizovane reprezentacije odgovarajuće Bulove funkcije. Kanoničnost je još jedno važno svojstvo koje, poput potpunih formi, ima redukovani BDD.

Definicija 4.3.7 Neki sistem za reprezentovanje Bulovih funkcija je *kanonski* ako za dve funkcije F i G važi da su jednake ako i samo ako im je reprezentacija jedinstvena.

Teorema 4.3.8 Za fiksirano uređenje promenljivih redukovani BDD je kanonski sistem za reprezentovanje Bulovih funkcija.

Sledećim sistematskim postupkom se polazeći od običnog dobija redukovani BDD u odnosu na jedinstveno uređenje promenljivih primenjeno prilikom konstrukcije:

1. Polazeći od završnih čvorova uočavaju se izomorfni podgrafovi. Za svaku klasu uočenih međusobno izomorfnih podgrafova, odbacuju se svi, sem jednog, na koga se usmeravaju sve ivice koje su pokazivale na preostale.
2. Svaki čvor čije obe izlazne ivice nakon prošlog koraka pokazuju na isti podgraf G se briše, dok se podgraf G direktno spaja sa čvorovima prethodnicima obrisanog čvora.
3. Pethodna dva koraka se ponavljaju dok god ima izmena na grafu.



Slika 4.8. Redukovani BDD.

Primetimo da korak 2 postupka predstavlja sledeću situaciju. Funkcija čvora $v \in V$ čije obe izlazne ivice pokazuju na isti podgraf G je oblika

$$f(v) = v \cdot f(G) + v' \cdot f(G) = f(G)$$

što je istovremeno i funkcija ivice čija je čvor v glava, pa je očigledno da je ovaj čvor suvišan i da njegovo uklanjanje ne utiče na vrednost reprezentovane funkcije.

Primer 4.3.9 Primenimo opisani postupak na završni BDD koji odgovara funkciji iz primera 4.3.5. Najpre se po četiri čvora 1, odnosno 0, svedu na po jedan čvor te vrste, a odgovarajuće ivice preusmere. Slično, dva podgrafa sa korenom u čvorovima označenim sa a se svode na jedan na koji se usmeravaju obe ivice najlevljeg čvora označenog sa d. Prema koraku 2, taj čvor se eliminiše, a T ivica levog čvora označenog sa c sada pokazuje na podgraf čiji je koren označen sa a. Dalje, tri podgrafa sa korenom označenim sa d se svode na jedan, na koji pokazuju E ivica levog čvora označenog sa c i obe ivice desnog čvora označenog sa c. Ponovo, prema koraku 2, taj čvor se briše i E ivica čvora označenog sa b pokazuje na podgraf čiji koren je označen sa d. Tako se dobija redukovani BDD prikazan na slici 4.8. ■

4.3.3 ITE-algoritam za direktnu konstrukciju redukovanog BDD-a

Umesto da se prvo konstruiše neoptimizovani BDD, pa da se on potom redukuje, i vremenski i prostorno je pogodnije da se konstrukcija redukovanog BDD-a izvede direktno. To se ostvaruje tako što se pre dodavanja čvora u graf proveri da li u grafu već postoji izomorfan podgraf. Jedna mogućnost je pretraživanje do tada konstruisanog grafa, ali efikasnija provera se postiže korištenjem *tabele jedinstvenosti*[16], re-

[16] Unique table.

alizovane kao heš tabela[17], u kojoj se čuvaju funkcije reprezentovane do tog trenutka konstrukcije. Upotreba tabele jedinstvenosti garantuje da će konstrukcija BDD-a u kome se kao komponente javljaju jednake funkcije završiti tako što funkcije imaju jedinstveni podgraf koji ih predstavlja. Posledica toga je da se i provera jednakosti funkcija vrši u konstantnom vremenu upoređivanjem na šta pokazuju izlazne ivice iz funkcijskih čvorova. Ključ koji se koristi za heš tabelu je oblika $(v, f_x, f_{x'})$, gde je v celi broj - redni broj promenljive po kojoj se izračunavaju kofaktori, a f_x i $f_{x'}$ pokazivači na podgrafove koji reprezentuju odgovarajuće kofaktore. Ako takav čvor već postoji u tabeli, prilikom konstrukcije BDD-a se samo pokazuje na njega, inače se on dodaje i u do tada konstruisani podgraf i u heš tabelu.

U nastavku teksta ćemo opisati osnovne ideje ITE-algoritma za direktnu konstrukciju redukovane BDD reprezentacije funkcije. Funkcija koja se zadaje kao ulaz sadrži pored standardnih operacija NOT, AND i OR[18] i operacije XOR, $NAND$, NOR, $XNOR$, $\leq$[19] i druge kojima se direktno, zbog efikasnosti, realizuju razni iskazni logički veznici. Zanimljivo je da se sve unarne i binarne operacije izražavaju pomoću jedne ternarne operacije

$$ITE(f, g, h) = f \cdot g + f' \cdot h$$

od koje i dolazi naziv celog algoritma[20]. ITE-algoritam je rekurzivan i zasniva se na sledećoj transformaciji u kojoj se koristi teorema ekspanzije 4.2.2:

$$\begin{aligned} ITE(f, g, h) &= f \cdot g + f' \cdot h \\ &= x_1 \cdot (f_{x_1} \cdot g_{x_1} + f'_{x_1} \cdot h_{x_1}) + x'_1 \cdot (f_{x'_1} \cdot g_{x'_1} + f'_{x'_1} \cdot h_{x'_1}) \\ &= ITE(x_1, ITE(f_{x_1}, g_{x_1}, h_{x_1}), ITE(f_{x'_1}, g_{x'_1}, h'_{x_1})), \end{aligned}$$

pri čemu su završni koraci:

$$ITE(1, f, g) = ITE(0, g, f) = ITE(f, 1, 0) = ITE(g, f, f) = f.$$

ITE-algoritam se može opisati sledećom procedurom:

procedure ITE(f, g, h)
begin
    (rezultat, zavrsni_korak) := Zavrsni_Korak(f, g, h)
    if zavrsni_korak **then** return(rezultat)
    x := Prva_Promenljiva(f, g, h)
    T := ITE(f_x, g_x, h_x)
    E := ITE($f_{x'}$, $g_{x'}$, $h_{x'}$)

[17] Hash table. To je struktura podataka u kojoj se podatak locira prema nekom ključu. Funkcija heširanja preslikava ključ podatka u adresu. Ako više ključeva daju istu adresu, preklapanje podataka se izbegava upotrebom povezane liste. Brzina pristupa je povezana sa kvalitetom funkcije heširanja koja treba da obezbedi da se malo ključeva preslikava u istu adresu, a time i da povezane liste budu kratke. Ako je ovo postignuto, brizina pronalaženja podatka je (skoro) konstantna.

[18] Operacije $'$, $\cdot$ i $+$.

[19] $xXORy = (x \cdot y') + (x' \cdot y)$, $xNANDy = (x \cdot y)'$, $xNORy = (x + y)'$, $xXNORy = (xXORy)'$, $x \leq y = x' + y$.

[20] Ime operacije ITE je skraćenica od *if-then-else* konstrukcije koju operacija u stvari simulira. $NOTx = ITE(x, 0, 1)$, $xANDy = ITE(x, y, 0) = x \cdot y + x' \cdot 0$, $xORy = ITE(x, 1, y)$, $xXORy = ITE(x, y', y)$, ...

if $T = E$ **then** return(T)
$R :=$ Naci_ili_Dodati_u_Hes_Tabelu(x, T, E)
return(R)
end

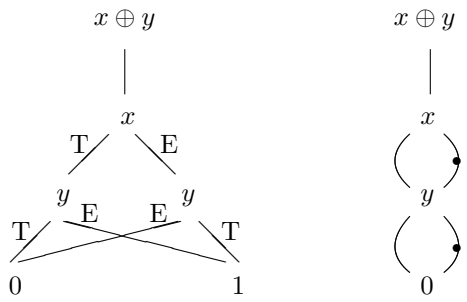
Dakle, funkcija se najpre zapiše u ITE-formi, nakon čega se poziva procedura. Ukoliko je reč o nekom od završnih slučajeva, postupak se odmah prekida. U suprotnom se nalazi prva promenljiva x i rekurzivno poziva ITE-procedura kojom se pronalaze T i E izlazne ivice čvora označenog sa x. Ako te ivice pokazuju na isti podgraf, čvor koji treba označiti sa x se odbacuje i pokazivač na podgraf za T se vraća kao rezultat. U suprotnom, ispituje se da li se podatak za ključ (x, T, E) nalazi u tabeli jedinstvenosti. Ako je podatak pronađen vraća se pokazivač na njega, a u suprotnom se novi podatak smešta u tabelu i vraća pokazivač na njega.

Ovako prezentovan algoritam ima eksponencijalan broj rekurzivnih poziva, a time i vreme izvršavanja, pošto, sem u završnom slučaju, svaki poziv generiše dva nova poziva. Ovaj problem se donekle rešava uvođenjem još jedne tabele u koju se smeštaju već izračunate funkcije. Ta tabela se naziva *tabela izračunatih funkcija*. Uočimo razliku između ove i tabele jedinstvenosti. Kod tabele jedinstvenosti se utvrđuje da li je već konstruisan podgraf određenog oblika, dok se kod tabele izračunatih funkcija utvrđuje da li je već obrađena neka funkcija, dakle provera se vrši pre nego se podgrafovi i generišu. Ako je tabela izračunatih funkcija dovoljno velika, ITE-procedura će biti pozvana po jednom za svaku različitu kombinaciju čvorova f, g i h, pa bismo dobili polinomijalnu složenost izračunavanja. Ovde je nerealna pretpostavka o veličini tabele izračunatih funkcija koja mora biti ograničena, pa se efikasnost postiže njenom pažljivom realizacijom.

Pored ove, postoje i druge metode za podizanje efikasnosti algoritma, ali njihovo izlaganje zbog velikog obima izostavljamo.

4.3.4 BDD sa komplementiranim ivicama

U do sada datim definicijama spominjane ivice u BDD-reprezentacijama funkcija su *regularne*. U cilju efikasnijeg reprezentovanja ponekada se koriste *komplementirane ivice*. Atribut komplementiranja mogu imati izlazne E-ivice čvorova $v \in V$ i izlazne ivice funkcijskih čvorova $f \in \Phi$. BDD sa komplementiranim ivicama je takođe kanonski sistem. Ako ivica ima atribut komplementiranja, njena funkcija je komplement funkcije čvora glave. Na slici 4.2 se to vidi za funkcije $f(a) = a$ i $f(a)' = a'$, a u opštem slučaju nije teško proveriti, da su BDD-reprezentacije za funkciju f i njen komplement f' veoma slične, zapravo da se jedna dobija od druge zamenom vrednosti završnih čvorova. Prema tome, BDD reprezentacija funkcija f i f' može biti data kao jedan BDD za funkciju $F = (f, f')$ sa dve izlazne vrednosti tako da BDD reprezentuje, recimo, funkciju f, a pored funkcijskog čvora za f postoji i funkcijski čvor za f' čija je izlazna ivica komplementirana i ulazi u isti čvor kao i izlazna ivica iz čvora za f. Odavde se konstrukcija komplementa funkcije i provera da li je neka funkcija komplement neke druge funkcije izvode u konstantnom vremenu. Recimo, funkcija g je komplement funkcije f ako im izlazne ivice pokazuju na isti podgraf i jedna je komplementirana, a druga nije.



Slika 4.9. BDD sa običnim i BDD sa komplementiranim granama za $x \oplus y$.

Primer 4.3.10 Na slici 4.9 date su dve BDD-reprezentacije funkcije ekskluzivne disjunkcije. Prvi BDD ima regularne, a drugi i komplementirane grane označene tačkama. Očigledna je ušteda koju postiže drugi BDD. ■

4.4 Komentar o metodi BDD

Veličina BDD-reprezentacija je eksponencijalna u odnosu na broj promenljivih u najgorem slučaju, ali se dobro ponaša za mnoge bitne funkcije, recimo BDD sa komplementiranim ivicama za AND, OR ili XOR je polinomijalno veliki u odnosu na veličinu zapisa funkcija. Ovo je značajno za XOR jer se ostvaruje znatna ušteda u odnosu na predstavljanje pomoću NOT, AND i/ili OR. Slično se i komplementiranje efikasno predstavlja upotrebom komplementiranih ivica. Koristeći BDD problem valjanosti se rešava u konstantnom vremenu - proverom da li se redukovani BDD za formulu sastoji samo od završnog čvora 1. Sa druge strane, problem čini to što efikasnost predstavljanja metodom BDD zavisi od izabranog redosleda promenljivih, a nalaženje dobrog redosleda nije uvek lako. Takođe, postoje funkcije koje se kompaktnije prikazuju nekim drugim postupcima i situacije u kojima su neke druge metode reprezentacije bliže konačnoj fizičkoj realizaciji.

4.5 Logička sinteza i verifikacija

Postupak *logičke sinteze* čipova u osnovi obuhvata:

1. Specifikaciju, tj. opis željenog funkcionisanja čipa na nekom programskom jeziku visokog nivoa u kome se specificira izlaz za svaki pojedinačni ulaz.

2. Prevođenje tog opisa u *mrežne liste* koje sadrže opise ćelija i njihove povezanosti.

3. Realizaciju mrežne liste na fizičkom medijumu.

Mrežne liste sadrže logičke elemente[21] i sekvencijalne elemente u koje spadaju memorijski registri i flip-flopovi[22]. Razlika između logičkih i sekvencijalnih elemenata je u tome što izlaz logičkog elementa zavisi samo od trenutnih ulaza, dok izlaz sekvencijalnog elementa zavisi i od prethodnog stanja samog elementa. Ćelija u mrežnoj listi je celina koja sadrži više pojedinačnih elemenata. Recimo jedna ćelija je puni sabirač. Kako se na različitim nivoima sinteze razlikuje detaljnost opisa, to se pojam ćelije postepeno menja i čini sve detaljnijim kako sinteza odmiče.

Poslednja dva koraka sinteze u sebi sadrže međukorake u kojima se vrše razne transformacije u cilju optimizacije finalnog proizvoda. Na nivou manipulisanja mrežnim listama optimizuju se broj ćelija u mreži i/ili dužina najdužeg puta u mreži shvaćenoj kao graf[23]. Ovi zahtevi su ponekad međusobno suprotstavljeni i njihovo usklađivanje prevazilazi okvire ovog teksta.

Postupak *verifikacije* se odnosi na dokazivanje da razne transformacije mrežnih lista očuvavaju njihovu ekvivalentnost i, na kraju, da dobijena mreža odgovara funkcionalnom opisu čipa koji je zadat na početku sinteze. U jednom stvarnom projektu verifikacija je urađena na sledeći način. Pre svega, postupak se odnosio samo na verifikaciju logičkog dela mrežne liste koji je preveliki, reda nekoliko stotina hiljada, pa čak i miliona ćelija sa hiljadama ulaza, da bi se direktno, bilo proverom svih ulaznih kombinacija, bilo na neki drugi način ispitala (ne)ekvivalentnost u realnom vremenu. Na osnovu iskustva formira se skup T ulaznih vektora za testiranje koji sadrži par hiljada elemenata. Dužina svakog vektora jednaka je broju ulaza mreže. Ako se dve mrežne liste M_1 i M_2, odnosno formule koje ove mreže predstavljaju, razlikuju na skupu T, pokazano je da mreže nisu ekvivalentne. Ako se mrežne liste ne razlikuju na skupu T, mreže se dele na nivoe, tako da svaki nivo ima istu dužinu puta počev od ulaza mreže. Zatim se uočavaju tačke prvog nivoa mreže, tj. izlazi iz nekih ćelija sa prvog nivoa, koje imaju uvek istu vrednost prilikom testiranja i pokušava se dokazati da su odgovarajuće podmreže koje dovode do tih izlaza ekvivalentne. Dokazivanje ekvivalentnosti se vrši na jedan od tri načina:

- Ako je broj ulaza, tj. promenljivih, relativno mali (do desetak) testiranje se vrši upoređivanjem svih mogućih ulaza i dobijenih rezultata.
- Ako je broj ulaza veći od desetak, a u formulama nema mnogo veznika ekskluzivne disjunkcije, koristi se metoda analitičkih tabloa.
- Ako je broj ulaza veći od desetak, a u formulama ima mnogo veznika ekskluzivne disjunkcije, koristi se metoda BDD.

U opštem slučaju odgovor koji se dobija pri obradi nekih tačaka je jedan od: ekvivalentne su, nisu ekvivalentne, ne zna se da li su mreže ekvivalentne. Poslednji odgovor se, recimo, javlja u situaciji kada su formule prevelike i računar na kome se formule ispituju nema dovoljno resursa da razreši problem. Pri radu se koriste već dokazane ekvivalentnosti nekih podmreža. To znači da se te podmreže predstavljaju samo sa po jednim bitom u ulaznom vektoru mreža M_1' i M_2' koje dovode do

[21] *Gates*. To su pored standardnih NOT, AND i OR i XOR, $NAND$ i drugi elementi koji se zbog efikasnosti direktno fizički realizuju.

[22] Flip-flop je bistabilni elektronski element sposoban da bude u jednom od dva stanja i da vrši promenu stanja u zavisnosti od prethodnog stanja i ulaza.

[23] Duži putevi u grafu dovode do većeg zagrevanja čipa što otežava povećanje frekvencije rada.

razmatranih tačaka. Ako se za neke tačke ustanovi neekvivalentnost, a u postupku se pozivalo na ekvivalentnost nekih podmreža M_1'' i M_2'' mreža M_1' i M_2', postupak se ponavlja pri čemu se ekvivalentnost podmreža ne koristi. Smisao ovog koraka je u tome što se pojedini ulazni vektori na kojima se mreže M_1' i M_2' razlikuju ne mogu pojaviti jer ih mreže M_1'' i M_2'' ne dopuštaju, već daju samo vrednosti izlaza na kojima se mreže M_1' i M_2', možda, ne razlikuju. Kada se, ipak, definitivno utvrdi neekvivalentnost nekih tačaka traže se ulazni vektori na kojima se tačke razlikuju. Ti vektori se ne nalaze u polaznom skupu T ulaznih vektora za testiranje, jer bi se na početku pokazalo da se analizirane tačke razlikuju. Nakon toga se vrši provera celih mreža na proširenom skupu T i ponavlja isti postupak. Na početku verifikacije uočene tačke su blizu ulaza, tj. predstavljaju relativno kratke formule opisane manjim brojem, do desetak, ćelija, a kako postupak odmiče tačke su sve dublje. Postupak se nastavlja dok god se ne dokaže ekvivalentnost ili neekvivalentnost celih mreža ili se ustanovi da se za razmatrane mreže ne može pokazati ni jedno ni drugo.

Logički aparat predstavlja osnovno sredstvo u sintezi i verifikaciji, ali, kako je na nekoliko mesta i naglašeno, u pojedinim koracima je još uvek potrebno preduzimati heurističke korake[24] jer nije pronađen optimalan i dovoljno opšti postupak, tako da je polje istraživanja još uvek široko.

4.6 Za dalje proučavanje

I Bulove algebre su veoma dobro analizirane u literaturi. Jedna od najpoznatijih referenci je [44]. BDD-metoda reprezentovanja Bulovih funkcija, kao i primene u verifikaciji logičkih kola detaljno su prikazane u [41]. Ispitivanje da li dobijena mreža odgovara funkcionalnom opisu datom na početku sinteze se, tipično, izvodi upotrebom konačnih automata nad beskonačnim rečima [125].

[24] Obratite pažnju da se u opisu postupka koriste konstrukcije: na osnovu iskustva, do desetak, ima mnogo, nema mnogo itd.

5

Predikatska logika prvog reda

Formalni jezik korišten u radu sa iskaznom logikom omogućavao je ispitivanje osnovnih logičkih operacija nad iskazima. Međutim, lako je formulisati iskaz koji očigledno sledi iz nekih drugih iskaza, a da se takvo izvođenje ne može opravdati sredstvima koja su do sada bila na raspolaganju. Zbog toga dolazi do uvođenja novih elemenata u logički jezik. U ovom poglavlju jezik klasične iskazne logike se širi kvantifikatorima i relacijskim i funkcijskim simbolima čime se dobija predikatska logika. Kako je dozvoljeno kvantifikatore primenjivati samo na promenljive, reč je o logici prvog reda. Ovo povećanje izražajnosti se, međutim, plaća gubitkom odlučivosti. Nakon definisanja jezika, osnovnih semantičkih i sintaksnih pojmova, u ovom poglavlju ćemo opisati i uopštenja dve već spomenute metode automatskog dokazivanja: rezolucije i tabloa.

5.1 Uvođenje kvantifikatora

Razmotrimo sledeći primer.

Primer 5.1.1 Neka su dati iskazi: "Svaki čovek je smrtan", "Sokrat je čovek" i "Sokrat je smrtan". Iako je intuitivno jasno da je poslednji iskaz posledica prva dva, nikakva semantička, ili sintaksna metoda koje su do sada korištene ne bi to potvrdila. Slično je i sa iskazima: "Nije tačno da neki čovek koji živi ne koristi kiseonik" i "Svaki živi čovek koristi kiseonik". ■

Ovakva situacija je posledica činjenice da istinitost nekih iskaza ne zavisi samo od forme njihove povezanosti, već i od unutrašnje strukture iskaza i značenja reči svaki i neki. U formulama u predikatskoj logici govoriće se o predikatima (relacijama) kao o svojstvima pojedinačnih objekata (elemenata nekog skupa), kao i o vezama između objekata. U daljem tekstu podrazumevaće se da se reči svaki i neki odnose samo na objekte, a ne i na njihova svojstva. Takva predikatska logika se naziva *predikatska logika prvog reda*. Zbog naznačene pretpostavke odrednica prvog reda će u buduće biti izostavljena.

Jezik predikatske logike nastaje izmenama i proširivanjem jezika iskazne logike. Time se dobija jezik koji je dovoljno bogat da se njime mogu zapisati gotovo svi iskazi koji su u svakodnevnoj upotrebi u matematici, a samim tim i brojni problemi koji se rešavaju pomoću računara. Dobitak na izražajnosti plaćen je, kao što će se videti u narednim odeljcima, povećanom složenošću postupka izvođenja i što je još gore sa stanovišta programiranja, gubitkom odlučivosti, jer ne postoji postupak koji će za proizvoljnu predikatsku formulu utvrditi da li jeste ili nije valjana. Iako se ne oseća uvek, ovaj efekat bitno ograničava snagu bilo kog programa koji, na možda i sasvim neprepoznatljiv način, operiše rečenicama predikatske logike.

Mnogi pojmovi (semantička i sintaksna posledica, dokaz, teorema, korektnost, kompletnost, karakterizacija, odlučivost itd.) koji su uvedeni pri razmatranju iskazne logike se u predikatskom slučaju analogno definišu, pa u takvim slučajevima nećemo ponavljati definicije.

5.2 Jezik i formule predikatske logike

Jezik predikatske logike je prebrojiv skup koji se sastoji iz skupa funkcijskih simbola $\{f, g, h, f_1, f_2, g_1, \ldots\}$, skup relacijskih simbola $\{P, Q, R, P_1, P_2, Q_1, \ldots\}$, logičkih veznika $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\exists$, $\forall$ i pomoćnih simbola, tj. leve i desne zagrade i zareza. Pored ovih simbola u predikatskim formulama će se pojavljivati i simboli promenljivih x, y, z, u, v, w, x_1, x_2, y_1, $\ldots$ Logički veznici i pomoćni znaci se nazivaju *logički simboli*.

Promenljive u jeziku predikatske logike odgovaraju, zavisno od situacije u kojoj se pojavljuju, nekom određenom objektu, ili objektima iz nekog skupa objekata koji se razmatra. Funkcijski simboli odgovaraju funkcijama nad tim objektima, a relacijski simboli svojstvima objekata ili odnosima između objekata. Često se, zbog svog posebnog značaja, relacija jednakosti označava simbolom $=$, a ne spominjanim simbolima. Međutim, u odeljku koji uvodi pojam interpretacije, u primeru 5.3.4, razmotriće se i slučaj kada simbolu $=$ ne mora odgovarati relacija jednakosti. Među logičkim veznicima novi su jedino *kvantifikatori* $\exists$ i $\forall$ kojima odgovaraju reči postoji (tj. neki) i svaki.

Funkcijskim i relacijskim simbolima jezika pridružena je *arnost*, tj. broj argumenata na koje se primenjuju. Ako je arnost funkcijskog simbola f jednaka 2, nije korektno pisati $f(x, y, z)$, ili $f(z)$, a jeste $f(u, w)$.

Definicija 5.2.1 *Konstante* su funkcijski simboli čija je arnost 0. Označavaju se simbolima a, b, c, a_1, a_2, b_1, $\ldots$

Složenije konstrukcije u predikatskoj logici su termi i predikatske formule (ili samo formule).

Definicija 5.2.2 *Termi* se definišu na sledeći način:

- Promenljive i konstante su termi.
- Ako su $t_1, t_2, \ldots, t_n$ termi i f funkcijski simbol arnosti n, onda je $f(t_1, t_2, \ldots, t_n)$ term.
- Termi se dobijaju samo konačnom primenom prethodnih pravila.

Definicija 5.2.3 *Predikatske formule* se definišu na sledeći način:

- Ako su $t_1, t_2, \ldots, t_n$ termi i P relacijski simbol arnosti n, onda je $P(t_1, t_2, \ldots, t_n)$ *atomska formula*.
- Atomske formule su predikatske formule.
- Ako su A i B predikatske formule i x promenljiva, onda su $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \to B)$, $(A \leftrightarrow B)$, $(\exists x)(A)$ i $(\forall x)(A)$ predikatske formule.
- Predikatske formule se dobijaju samo konačnom primenom prethodnih pravila.

Kao i kod iskaznih formula, brisaće se zagrade kada to ne izaziva zabunu. Pri tome nikada neće biti brisane zagrade koje obuhvataju argumente termova i atomskih formula.

Za formule oblika $(\forall x)A$ kaže se da su *univerzalno kvantifikovane*, ili univerzalne, dok su formule oblika $(\exists x)A$ *egzistencijalno kvantifikovane*, ili egzistencijalne.

Primer 5.2.4 Termi su: x, a, $f(x)$, $g(u, w, w)$, $h(a, z)$, $f_1(h(x, x))$, $f_1(a, z, w))$ itd. Atomske formule su: $P()$, $Q(x)$, $R(a, b, x, c)$, $P_1(f_1(h(x, x), z, h1(a, z, w)), x)$ itd. Predikatske formule su: $P(x)$, $(\exists x)(P(x))$, $(\forall x)(\exists y)(R(a, b, x, c) \to Q(z))$ i $(\forall x)((\exists y)(R(a, b, x, c)) \to (\forall x)Q(x))$. Ako P označava pojam "biti čovek", Q "biti smrtan", a c "Sokrat", onda formule $(\forall x)(P(x) \to Q(x))$, $P(c)$ i $Q(c)$ označavaju iskaze "Svaki čovek je smrtan", "Sokrat je čovek" i "Sokrat je smrtan", dok formula $((\forall x)(P(x) \to Q(x)) \wedge P(c)) \to Q(c)$ označava rečenicu "Ako je svaki čovek smrtan i Sokrat je čovek, onda je Sokrat smrtan". Rečenica "Za svako ϵ veće od 0 postoji δ veće od 0 tako da kad je razlika x i x_0 po apsolutnoj vrednosti manja ili jednaka od δ, tada je razlika $f(x)$ i $f(x_0)$ po apsolutnoj vrednosti manja ili jednaka od ϵ" kojom se u matematičkoj analizi uvodi pojam neprekidnosti se zapisuje u obliku $(\forall\epsilon)((\epsilon > 0) \to (\exists\delta)((\delta > 0) \wedge (|x - x_0| \leq \delta \to |f(x) - f(x_0)| \leq \epsilon)))$. ■

U predikatskim formulama razlikuju se promenljive na koje deluje neki kvantifikator, od promenljivih na koje ne deluje ni jedan kvantifikator.

Definicija 5.2.5 Pojava promenljive x u predikatskoj formuli A je *vezana*:

1. ako je to pojava oblika $(\exists x)$ ili $(\forall x)$, ili
2. ako je to pojava u podformulama formule A koje su oblika $(\exists x)(B)$, odnosno $(\forall x)(B)$.

Ako pojava promenljive x nije vezana, onda je ta pojava *slobodna*. Formula je *zatvorena* ako su u njoj sve pojave promenljivih vezane. Zatvorena formula se naziva i *rečenica*. Ako su $x_1, \ldots, x_n$ sve slobodne promenljive u formuli A, formula $(\forall x_1)\ldots(\forall x_n)A$ se naziva *univerzalno zatvorenje*, a formula $(\exists x_1)\ldots(\exists x_n)A$ *egzistencijalno zatvorenje* formule A.

Primer 5.2.6 U formuli $(\forall x)P(x, f(y))$ obe pojave promenljive x su vezane, dok je pojava promenljive y slobodna. U formuli $(\forall x)P(x, f(y)) \to (\exists z)Q(x, z)$, prve dve pojave promenljive x su vezane, a treća pojava je slobodna. Pojava promenljive y je slobodna, a pojava promenljive z je vezana. ■

Definicija 5.2.7 Ako je A formula i t term, kažemo da je t *slobodan za promenljivu* x *u* A ako ni za koju promenljivu y iz terma t promenljiva x nema slobodno pojavljivanje u nekoj podformuli formule A koja je oblika $(\forall y)B$ ili $(\exists y)B$.

Primer 5.2.8 U formuli $(\forall y)P(x, y)$ sve promenljive sem y su slobodne za x. Term $f(x, y)$ nije slobodan za promenljivu x u formuli $(\forall x)(\forall y)P(x, y)$, jer se x javlja kao slobodna promenljiva u formuli $(\forall y)P(x, y)$, a jeste slobodan za x u formuli $(\forall z)P(x, z)$. Svaki term bez promenljivih, kao što je na primer konstanta, je slobodan za svaku promenljivu u svakoj formuli. ■

Sa $A(x_1, \ldots, x_m)$ označavaćemo da je skup slobodnih promenljivih formule A podskup skupa $\{x_1, \ldots, x_m\}$. Ako su $A(x)$ formula i t term, sa $A(t/x)$ ćemo označiti rezultat zamene u A svih slobodnih pojava promenljive x termom t.

Na kraju ovog odeljka treba još skrenuti pažnju da se obično skup promenljivih smatra fiksiranim, kakvim je prethodno dat. Međutim, često se variraju skupovi funkcijskih i relacijskih simbola od kojih u tom slučaju zavisi i skup formula predikatske logike. Nekada se funkcijskim i relacijskim simbolima daju "svakodnevni" nazivi, poput naziva za rođačke veze ili Stariji i sl., a koriste se i matematički simboli poput znaka $=$, $<$, $>$, ...

Primer 5.2.9 Formula iz primera 5.1.1 se može zapisati kao
$((\forall x)(\text{Čovek}(x) \rightarrow \text{Smrtan}(x)) \wedge \text{Čovek(Sokrat)}) \rightarrow \text{Smrtan(Sokrat)}$. ■

5.3 Interpretacija i tačnost predikatskih formula

Iskaznim slovima je pri interpretaciji značenje bilo ili tačno ili netačno, a istinitosnim tablicama se određivala zadovoljivost formula. U predikatskoj logici tumačenje simbola je komplikovanije, što dovodi do toga da metodom istinitosnih tablica nije moguće odrediti istinitosni status nekih formula. Recimo, u odeljku 5.10 je objašnjeno da za formule oblika $(\forall x)A$ ili $(\exists x)A$ tablična metoda u opštem slučaju nije primenljiva. U ovom odeljku će biti razmotren pojam istinitosne vrednosti predikatskih formula.

Interpretacijom u predikatskoj logici simboli konstanti, funkcija i relacija, dobijaju značenje elemenata nekog skupa, odnosno funkcija i relacija nad njima. Taj skup se naziva *domen interpretacije*.

Definicija 5.3.1 Neka su Dom skup i Rel i Fun skupovi relacija, odnosno funkcija, definisanih nad skupom Dom. *Relacijsko-funkcijska struktura* je algebarski sistem $\langle Dom, Rel, Fun\rangle$. Skup Dom je *domen* te strukture.

Primer 5.3.2 Neka skup $Dom = \{\text{Pera}, \text{Mika}, \text{Žika}, \text{Veca}\}$ bude domen. Primer funkcije bi bio Otac, a primer relacije Rođak, koji su definisani sa: Otac(Mika) = Pera, Otac(Žika) = Mika, Otac(Veca) = Pera, odnosno Rodjak(Mika, Pera), Rodjak(Žika, Mika), Rodjak(Veca, Pera) itd. Dalje, može se pretpostaviti da je relacija Rođak relacija ekvivalencije, dakle refleksivna, simetrična i tranzitivna. Slično, mogu se razmatrati i druge funkcije (Majka, Brat, ...) i relacije (Stariji, Udata, ...). Neka je skup relacija nad skupom Dom označen sa Rel, a skup fukcija sa Fun. Tada je $\langle Dom, Rel, Fun\rangle$ jedna relacijsko-funkcijska struktura. ■

Definicija 5.3.3 *Interpretacija* je par $\langle D, \psi \rangle$, gde je D relacijsko-funkcijska struktura, a ψ funkcija koja relacijskim i funkcijskim simbolima jezika predikatske logike pridružuje relacije i funkcije odgovarajuće arnosti definisane na domenu strukture D. Interpretacije u kojima se znak $=$ interpretira kao jednakost nazivaju se *normalni modeli* .

Primer 5.3.4 Neka je $\langle Dom, Rel, Fun \rangle$ relacijsko-funkcijska struktura iz primera 5.3.2. Jedna interpretacija formule $(((f(x) = y) \wedge (f(y) = z)) \rightarrow P(x, a))$ bi bio iskaz "Ako je y otac od x i z otac od y, onda su x i Pera rođaci". Funkcijski simbol f je interpretiran kao funkcija Otac, konstanta a kao Pera, a relacijski simboli P i $=$, kao relacije Rođak i jednakost. Pri drugoj interpretaciji, formuli bi odgovarao iskaz "Ako brat od x voli y i brat od y voli z, onda je x stariji od Vece". Ovde je funkcijski simbol protumačen kao funkcija Brat, konstanta a kao Veca, relacijski simbol P kao relacija Stariji, a relacijski simbol $=$ kao relacija Voleti. ■

Kao i u iskaznom slučaju, o istinitosnoj vrednosti predikatske formule ima smisla govoriti tek pošto se interpretira, tj. pošto se odredi značenje simbola iz jezika predikatskog računa. Tek tada formule govore o stvarnim objektima i njihovim osobinama, pa se može diskutovati njihova istinitost. Sledeće definicije daju pravila za izračunavanje vrednosti terama i formula pri nekoj interpretaciji. Najpre je potrebno uvesti preslikavanja koje promenljivima iz formula pridružuje vrednosti iz domena. Pri tome se pretpostavlja da su sve promenljive, bez obzira kako se zvale, poređane u jedan niz i da su označene redom sa x_1, x_2, ...

Definicija 5.3.5 *Valuacija* v u odnosu na domen Dom je preslikavanje skupa promenljivih u elemente domena. *Valuacija za interpretaciju* $I = \langle \langle Dom, Rel, Fun \rangle, \psi \rangle$ je valuacija u odnosu na domen te interpretacije. Kaže se da je $v(x_i)$ *vrednost promenljive* x_i pri valuaciji v. Ako su v i w dve valuacije u odnosu na isti domen D, $v \equiv_{x_k} w$ označava da je $v(x_i) = w(x_i)$, za svako i, sem eventualno za $i = k$.

Na dalje ćemo, kada se govori o valuaciji i interpretaciji, uvek smatrati da je reč o valuaciji za tu interpretaciju.

Definicija 5.3.6 *Vrednost terma* t pri interpretaciji $I = \langle \langle Dom, Rel, Fun \rangle, \psi \rangle$ i valuaciji v (u oznaci $I(t)_v$) je:

- ako je t simbol konstante a, $I(t)_v = \psi(a)$,
- ako je t promenljiva x_i, $I(t)_v = v(x_i)$ i
- ako je t oblika $f(t_1, t_2, \ldots, t_n)$, i ako su vrednosti terama $t_1, t_2, \ldots, t_n$ pri interpretaciji I i valuaciji v, redom $I(t_1)_v$, $I(t_2)_v$, ..., $I(t_n)_v$, onda je $I(t)_v = \psi(f)(I(t_1)_v, I(t_2)_v, \ldots, I(t_n)_v)$.

Definicija 5.3.7 *Istinitosna vrednost predikatske formule* A pri interpretaciji $I = \langle \langle Dom, Rel, Fun \rangle, \psi \rangle$ i valuaciji v (u oznaci $I(A)_v$) je:

- ako je A atomska formula $P(t_1, t_2, \ldots, t_n)$, i ako su vrednosti terama t_1, t_2, ..., t_n pri interpretaciji I i valuaciji v redom $I(t_1)_v$, $I(t_2)_v$, ..., $I(t_n)_v$, onda je $I(A)_v = \psi(A)(I(t_1)_v, I(t_2)_v, \ldots, I(t_n)_v)$, tj. $I(A)_v$ je tačno ako i samo ako je n-torka $(I(t_1)_v, I(t_2)_v, \ldots, I(t_n)_v)$ u relaciji $\psi(A)$,

- ako je A oblika $\neg B$, $I(A)_v$ je tačno ako i samo ako je $I(B)_v$ netačno,
- ako je A oblika $B \wedge C$, $I(A)_v$ je tačno ako i samo ako su tačni i $I(B)_v$ i $I(C)_v$,
- ako je A oblika $B \vee C$, $I(A)_v$ je tačno ako i samo ako je tačno $I(B)_v$ ili $I(C)_v$,
- ako je A oblika $B \rightarrow C$, $I(A)_v$ je tačno uvek, sem ako je $I(B)_v$ tačno, a $I(C)_v$ netačno,
- ako je A oblika $B \leftrightarrow C$, $I(A)_v$ je tačno ako i samo ako su vrednosti $I(B)_v$ i $I(C)_v$ jednake,
- ako je A oblika $(\forall x_i)B$, $I(A)_v$ je tačno ako i samo ako je $I(B)_w$ tačno za svaku valuaciju w za koju je $v \equiv_{x_i} w$ i
- ako je A oblika $(\exists x_i)B$, $I(A)_v$ je tačno ako i samo ako postoji valuacija w za koju je $v \equiv_{x_i} w$ takva da je $I(B)_w$ tačno.

$I(A)_v$ je netačno ako i samo ako nije tačno.

U delu definicije 5.3.7 koji se odnosi na formule čije su podformule povezane iskaznim veznicima, izračunavanje istinitosne vrednosti se svodi na tablično izračunavanje. U delu definicije koji se odnosi na kvantifikatore precizirano je intuitivno shvatanje reči svaki i neki (tj. postoji). Kaže se da je formula $(\forall x_i)B$ tačna pri interpretaciji I ako je pri istoj interpretaciji formula B tačna bez obzira koju vrednost iz domena imala promenljiva x_i. Slično, formula $(\exists x_i)B$ je tačna pri interpretaciji I ako postoji element iz domena kojim se zameni promenljiva x_i u B tako da dobijena formula bude tačna pri istoj interpretaciji.

Definicija 5.3.8 Formula A je *tačna* pri interpretaciji I (u oznaci $I \models A$) ako je za svaku valuaciju v, $I(A)_v = \top$. Tada se kaže i da je I *model* za A. Formula A je *netačna* pri interpretaciji I ako je za svaku valuaciju v, $I(A)_v = \bot$. Ako je A netačna pri interpretaciji I, onda je I *kontramodel* za A.

U tvrđenju 5.3.10 ćemo pokazati da su zatvorene formule u svakoj interpretaciji bilo tačne, bilo netačne, pa se ponašaju slično iskaznim slovima. Sa druge strane, kao što se vidi u primeru 5.3.9 vrednost pri nekoj interpretaciji formula koje nisu zatvorene često nije ni tačna, ni netačna. Prema tome, istinitosna vrednost formule pri nekoj interpretaciji i valuaciji ne zavisi od vezanih promenljivih, već eventualno od slobodnih.

Primer 5.3.9 Posmatrajmo formulu $(\forall x)(y \leq x)$ i interpretaciju $I = \langle\langle \mathbb{N}, \{\leq\}, Fun\rangle, \psi\rangle$, gde su $\mathbb{N}$ skup prirodnih brojeva, skup Fun proizvoljan, skup relacija sadrži samo relaciju manje ili jednako definisanu na skupu prirodnih brojeva i ψ preslikavanje koje binarnom relacijskom simbolu $\leq$ pridružuje upravo tu relaciju. Neka su v valuacija za koju je $v(y) = 0$ i w valuacija za koju važi $v \equiv_y w$ i $w(y) = 100$. Očigledno je da je $I((\forall x)(y \leq x))_v = \top$ i $I((\forall x)(y \leq x))_w = \bot$. ■

Teorema 5.3.10 Neka je $I = \langle\langle Dom, Rel, Fun\rangle, \psi\rangle$ interpretacija.

Neka su t term i $x_1, \ldots, x_n$ sve promenljive koje se javljaju u t. Za bilo koje dve valuacije v i w, ako je $v(x_i) = w(x_i)$ za $0 \leq i \leq n$, onda je $I(t)_v = I(t)_w$.

Neka je A formula čije su sve promenljive (i slobodne i vezane) u skupu $\{x_1, \ldots, x_n\}$. Za bilo koje dve valuacije v i w, ako je $v(x_i) = w(x_i)$ za sve *slobodne promenljive* x_i, $0 \leq i \leq n$, u formuli A, onda je $I(A)_v = I(A)_w$.

Dokaz. Dokaz se izvodi indukcijom po složenost terma, odnosno formule.

Ako je $t = x_i$ za neko $i \leq n$, onda je $I(t)_v = v(x_i) = w(x_i) = I(t)_w$. Ako je $t = c$ za neki simbol konstante c, onda je $I(t)_v = \psi(c) = I(t)_w$. Ako je $t = f(t_1, \ldots, t_m)$ i za svaki term t_i $(0 \leq i \leq m)$ tvrđenje važi onda je $I(t)_v = \psi(f)(I(t_1)_v, \ldots, I(t_m)_v) = \psi(f)(I(t_1)_w, \ldots, I(t_m)_w) = I(t)_w$, čime je prvi deo tvrđenja dokazan.

Ako je $A = P(t_1, \ldots, t_m)$ atomska formula, na osnovu prethodno dokazanog za terme, $I(A)_v = \psi(P)(I(t_1)_v, \ldots, I(t_m)_v) = \psi(P)(I(t_1)_w, \ldots, I(t_m)_w) = I(A)_w$. Tvrđenje za formulu A jednog od oblika $\neg B$, $B \wedge C$, $B \vee C$, $B \to C$ i $B \leftrightarrow C$ se sprovodi na osnovu indukcijske pretpostavke. Recimo, $I(A)_v = I(\neg B)_v = \neg I(B)_v = \neg I(B)_w = I(\neg B)_w = I(A)_w$. Neka je $A = (\forall y)B$, $y \in \{x_1, \ldots, x_n\}$. $I((\forall y)B)_v = \top$ ako i samo ako za svaku valuaciju v' takvu da je $v \equiv_y v'$ važi $I(B)_{v'} = \top$. Prema indukcijskoj pretpostavci važi $I(B)_{v'} = I(B)_{w'}$ za svaku interpretaciju w' koja se sa v' poklapa na promenljivim iz skupa $\{x_1, \ldots, x_n\}$ koje su slobodne u B, dok je za ostale promenljive x, $w(x) = w'(x)$. Zapravo je, zbog pretpostavke o poklapanju v i w, za takve w' istovremeno i $w \equiv_y w'$. Odatle je za svaku valuaciju v' takvu da je $v \equiv_y v'$, $I(B)_{v'} = \top$ ako i samo ako je za svaku valuaciju w' takvu da je $w \equiv_y w'$, $I(B)_{w'} = \top$, što je ekvivalentno sa $I((\forall y)B)_w = \top$. Prema tome je $I(A)_v = I(A)_w$. Slično se pokazuje i za formule oblika $(\exists y)B$, pa je tvrđenje dokazano. ■

Definicija 5.3.11 Formula A je *zadovoljiva* ako postoje interpretacija I i valuacija v takve da je $I(A)_v = \top$, a *kontradikcija* ili *nezadovoljiva* ako je za svaku interpretaciju i svaku valuaciju $I(A)_v = \bot$.

Prema teoremi 5.3.10, istinitosna vrednost rečenice ne zavisi od valuacije, pa je *rečenica zadovoljiva* ako je tačna pri nekoj nekoj interpretaciji, tj. ima model. U opštem slučaju, formula je zadovoljiva ako je pri nekoj interpretaciji tačno njeno egzistencijalno zatvorenje, dok je formula tačna pri nekoj interpretaciji ako je tačno njeno univerzalno zatvorenje.

Primer 5.3.12 Ponovo razmotrimo relacijsko-funkcijsku strukturu datu u primeru 5.3.2. Formula $(\forall x)P(a, x)$ je tačna ako se P interpretira kao relacija Rođak, i konstanta a kao Pera, pošto su sve osobe u srodstvu sa Perom. Takođe je tačna i formula $(\exists x)(\exists y)P(y, x)$ jer postoje osobe koje su rođaci.

Ako je $Dom = \{1, 2\}$ i ako se pri interpretaciji I relacijski simbol P tumači kao unarna relacija "biti jednak jedinici", a simbol Q kao binarna relacija "biti veći ili jednak", formula $(\forall x)(\exists y)P(x)$ je netačna pri interpretaciji I, dok su formule $(\exists x)P(x)$ i $(\forall x)(\exists y)Q(x, y)$ tačne pri interpretaciji I.

Ako je Dom skup celih brojeva i pri nekoj interpretaciji I relacijski simbol P se tumači kao jednakost, konstanta a kao nula, a simbol f kao sabiranje, formula $(\forall x)(\exists y)P(f(x, y), a)$ se interpretira sa "Za svaki ceo broj postoji ceo broj tako da je njihov zbir nula", što je očigledno tačno. Međutim, dovoljno je samo promeniti skup Dom da bude skup pozitivnih celih brojeva, pa da pri analognoj interpretaciji ova formula bude netačna. ■

Primer 5.3.13 Stek[1] je struktura podataka koja se često koristi u računarstvu za privremeno čuvanje podataka, recimo pri prenosu stvarnih argumenata u podprograme ili prilikom prevođenja izvornih u izvršne verzije programa. Jedini način kojim se manipuliše stekom zasniva se na principu LIFO[2], tj. 'poslednji stavljen, prvi uzet'. Osnovne operacije koje se pri tome koriste omogućavaju stavljanje na vrh steka, očitavanje vrha steka i uklanjanje podatka sa vrha steka[3].

Sredstvima predikatske logike moguće je opisati ovakvu strukturu podataka. Jezik koji razmatramo se sastoji od simbola konstante c_{novi}, dva unarna funkcijska simbola f_{ostatak} i f_{vrh} i relacijskih simbola R_{stavi} i $=$. Neka je A neprazan skup, A^* skup svih reči alfabeta A i ε prazna reč. Funkcije Vrh i Ostatak preslikavaju skup A^* u skup A^* tako da važi:

- $\text{Vrh}(\varepsilon) = \varepsilon$,
- $\text{Vrh}(wa) = a$, za sve $a \in A$, $w \in A^*$,
- $\text{Ostatak}(\varepsilon) = \varepsilon$ i
- $\text{Ostatak}(wa) = w$, za sve $a \in A$, $w \in A^*$.

Relacija $\text{Stavi} \subset A^* \times A \times A^*$ je definisana sa:

- $(w, a, u) \in \text{Stavi}$ ako i samo ako $u = wa$, za sve $a \in A$, $w, u \in A^*$,

dok je $=_{A^*}$ relacija jednakosti među rečima. Posmatrajmo interpretaciju $I = \langle\langle A^*, \{\text{Stavi}, =_{A^*}\}, \{\text{Vrh}, \text{Ostatak}\}\rangle, \psi\rangle$ koja daje sledeće značenje simbolima jezika:

- $\psi(R_{\text{stavi}}) = \text{Stavi}$,
- $\psi(=)$ je relacija $=_{A^*}$,
- $\psi(c_{\text{novi}}) = \varepsilon$,
- $\psi(f_{\text{ostatak}}) = \text{Ostatak}$ i
- $\psi(f_{\text{vrh}}) = \text{Vrh}$.

Formula

$$A_{\text{atom}}(x) = (\exists y)(\neg(y = c_{\text{novi}}) \wedge (f_{\text{ostatak}}(y) = c_{\text{novi}}) \wedge (f_{\text{vrh}}(y) = x))$$

ima jednu slobodnu promenljivu x. Pri nekoj valuaciji v je $I(A_{\text{atom}}(x))_v$ tačno ako i samo ako je $v(x) \in A$. Formula

$$\begin{aligned} A_{\text{LIFO}} = (\forall x_1)(\forall x_2)(\forall x_3)((A_{\text{atom}}(x_2) \quad &\wedge \quad R_{\text{stavi}}(x_1, x_2, x_3)) \rightarrow \\ &(\quad (x_1 = f_{\text{ostatak}}(x_3)) \wedge (x_2 = f_{\text{vrh}}(x_3)))) \end{aligned}$$

modelira način rada sa stekom pošto $I \models A_{\text{LIFO}}$ ako i samo ako x_2 predstavlja element skupa A, dok su x_1 i x_3 reči iz A^* takve da se x_3 dobija stavljanjem x_2 na stek x_1, a x_1 uklanjanjem elementa sa vrha steka x_3. ■

[1] Engleska reč *stack* se prevodi kao gomila ili stog, ali ćemo zbog raširene upotrebe koristiti njenu domaću verziju *stek*.

[2] Skraćenicu čine prva slova fraze *Last in first out*.

[3] Engleski nazivi ovih operacija su *push*, *top* i *pop*.

5.4 Valjane formule predikatskog računa

Definicija 5.4.1 Formula A je *valjana* (u oznaci $\models A$) ako je tačna pri svakoj interpertaciji.

Primetimo da je formula A valjana ako i samo ako formula $\neg A$ nije zadovoljiva. Pošto se svaka formula pri ispitivanju tačnosti ponaša kao sopstveno univerzalno zatvorenje, isto važi i za ispitivanje valjanosti. Rečenica A je valjana ako i samo ako je tačna u svakom modelu, tj. ako i samo ako $\neg A$ nema model.

Semantički postupak ispitivanja istinitosti formula je komplikovaniji nego u iskaznoj logici, pa ga ovde nećemo razmatrati, već ćemo odgovarajuće probleme razmotriti u okviru nekih formalnih sistema.

Zanimljivu klasu valjanih formula predstavljaju *izvodi iz tautologija*, tj. formule nastale iz tautologija sistematskom zamenom shema slova predikatskim formulama.

Primer 5.4.2 Formula $(\exists x)P(x) \to (\exists x)P(x)$ je izvod tautologije $A \to A$. ■

Neke od valjanih formula su date u tabeli 5.1. Pri tome se za distributivne zakone $\exists$ prema $\wedge$ i $\forall$ prema $\vee$ zahteva se da se promenljiva x ne pojavljuje slobodno u formuli B, dok u zakonima o preimenovanju promenljiva y ne sme imati slobodnu pojavu u formuli A.

Izvodi iz tautologija	
$\neg(\forall x)A \leftrightarrow (\exists x)\neg B$	(zakon De Morgana)
$\neg(\exists x)A \leftrightarrow (\forall x)\neg B$	(zakon De Morgana)
$(\exists x)(A \vee B) \leftrightarrow ((\exists x)A \vee (\exists x)B)$	(zakon distribucije $\exists$ prema $\vee$)
$(\forall x)(A \wedge B) \leftrightarrow ((\forall x)A \wedge (\forall x)B)$	(zakon distribucije $\forall$ prema $\wedge$)
$(\forall x)(\forall y)A \leftrightarrow (\forall y)(\forall x)A$	(zakon promene redosleda kvantifikacije)
$(\exists x)(\exists y)A \leftrightarrow (\exists y)(\exists x)A$	(zakon promene redosleda kvantifikacije)
$(\exists x)(A \wedge B) \leftrightarrow ((\exists x)A \wedge B)$	(zakon distribucije $\exists$ prema $\wedge$)
$(\forall x)(A \vee B) \leftrightarrow ((\forall x)A \vee B)$	(zakon distribucije $\forall$ prema $\vee$)
$(\exists x)A(x) \leftrightarrow (\exists y)A(y)$	(zakon o preimenovanju vezane promenljive)
$(\forall x)A(x) \leftrightarrow (\forall y)A(y)$	(zakon o preimenovanju vezane promenljive)

Tabela 5.1. Spisak nekih valjanih formula.

Slede dva primera u kojima su date formule koje nisu valjane, iako to na prvi pogled možda ne izgleda tako. Primer 5.4.3 pokazuje da se redosled različitih kvantifikatora u formuli ne sme proizvoljno menjati, ako se želi da se sačuva istinitost formule. Formula u primeru 5.4.4 opominje da treba biti obazriv u pogledu unutrašnje strukture formula.

Primer 5.4.3 Formula $(\exists y)(\forall x)P(x,y) \to (\forall x)(\exists y)P(x,y)$ je valjana. Intuitivno shvaćeno, njeno značenje je "Ako postoji objekat c takav da za svako x važi $P(x,c)$, tada za svaki x postoji objekat, recimo baš taj c, tako da je $P(x,c)$." Međutim, formula $(\forall x)(\exists y)P(x,y) \to (\exists y)(\forall x)P(x,y)$ nije valjana. Jedna interpretacija pri kojoj formula ne važi ima za domen skup realnih brojeva, a simbol P se tumači kao biti veće. I dok leva strana formule znači da "Za svaki realni broj postoji realni broj koji je veći od njega", što je svakako tačno, desna strana formule znači da "Postoji realan broj veći od svakog drugog realnog broja", što očigledno ne važi. ■

Primer 5.4.4 Neka je t oznaka za term. Formula $(\forall x)A(x) \to A(t/x)$ nije valjana. Neka je $A(x)$ formula oblika $(\exists y)P(x,y)$, i neka je term t jednak y. Tada je cela formula $(\forall x)(\exists y)P(x,y) \to (\exists y)P(y,y)$ i nije tačna, recimo, pri interpretaciji u kojoj je domen skup realnih brojeva, a simbol P se tumači kao biti veće. ■

Primer 5.4.4 je donekle suprotan intuiciji. Iako je promenljiva x univerzalno kvantifikovana, nije korektno zameniti je bilo čime. Primetimo da term y nije slobodan za promenljivu x u formuli $(\forall x)(\exists y)P(x,y)$. U sledećoj teoremi ćemo pokazati da je uslov "term je slobodan za promenljivu" dovoljan da formula razmatrana u primeru 5.4.4 bude valjana.

Teorema 5.4.5 Ako je term t slobodan za promenljivu x u formuli $A(x)$, formula $(\forall x)A(x) \to A(t/x)$ je valjana.

Dokaz. Neka za neku interpretaciju I važi $I \not\models (\forall x)A(x) \to A(t/x)$. To znači da je za neku valuaciju v, $I((\forall x)A(x) \to A(t/x))_v = \bot$, odnosno da za neku valuaciju v važi $I((\forall x)A(x))_v = \top$ i $I(A(t/x))_v = \bot$. Posmatrajmo valuaciju $v' \equiv_x v$ koja se jednaka sa v, sem što je $v'(x) = I(t)_v$. Tada bi bilo $I(A(x))_{v'} = I(A(t/x))_v = \bot$, odakle bi suprotno pretpostavci bilo $I((\forall x)A(x))_v = \bot$. ■

5.5 Preneks forme i Skolemova forma

Zbog postojanja kvantifikatora u predikatskim formulama, normalne forme ovde imaju nešto drugačiji oblik nego u iskaznoj logici. Pokazalo se najpogodnijim da se formula transformiše u takozvanu preneks normalnu formu.

Definicija 5.5.1 Formula oblika $(Q_1x_1)(Q_2x_2)\dots(Q_nx_n)A$, gde je svaki od Q_i bilo znak $\forall$, bilo $\exists$, a u formuli A nema kvantifikatora, je u *preneks normalnoj formi*. $(Q_1x_1)(Q_2x_2)\dots(Q_nx_n)$ se naziva *prefiks*, dok je A *matrica* formule.

Primer 5.5.2 Formula $(\exists x)(\forall y)(\forall z)(P(x,z) \to \neg Q(z,z,a,y))$ je u preneks normalnoj formi. Prefiks formule je $(\exists x)(\forall y)(\forall z)$, a matrica je formula $(P(x,z) \to \neg Q(z,z,a,y))$. ■

Svaka predikatska formula se može transformisati u njoj ekvivalentnu preneks formu. Sledeća procedura opisuje taj postupak:

procedure PreneksForma
begin
1 Primenom zakona o uklanjanju ekvivalencije i implikacije
dobiti formule u kojima nema simbola $\leftrightarrow$ i $\to$.
2 Ponavljati primenu
2a iskaznih De Morganovih zakona,
2b predikatskih De Morganovih zakona i
2c zakona uklanjanja dvojne negacije
da bi se simboli $\neg$ uklonili, ili spustili neposredno do atomskih formula.
3 Ako je potrebno primeniti zakone o preimenovanju vezanih promenljivih.
4 Ponavljati primenu zakona distibutivnosti kvantifikatora prema $\wedge$ i $\vee$

kako bi se kvantifikatori pomerali u levo.
end

Primer 5.5.3 Neka je procedura PreneksForma primenjena za dobijanje preneks forme formule $(\forall x)P(x) \to (\exists x)Q(x)$. Najpre se, primenom pravila (1), dobija formula $(\neg(\forall x)P(x)) \vee (\exists x)Q(x)$. Primenom pravila (2b) dobija se formula $(\exists x)\neg P(x) \vee (\exists x)Q(x)$. Primenom pravila (4) kvantifikator $\exists$ se izvlači u levo i dobija se formula $(\exists x)(\neg P(x) \vee Q(x))$ koja jeste u preneks formi. ■

Preimenovanje promenljivih je potrebno, recimo u formuli kao što je $(\exists x)A \wedge (\exists x)B$ u kojoj se najpre vrši preimenovanje $(\exists x)A \wedge (\exists y)B$, pod pretpostavkom da se promenljiva y ne javlja u A, a zatim se formula transformiše u $(\exists x)(\exists y)(A \wedge B)$.

Kako u matrici neke formule nema kvantifikatora, moguće je direktno primeniti iskazne De Morganove i distributivne zakone, pri čemu se dobija formula ekvivalentna polaznoj. Odnosno, primenom procedure NormalnaForma matrica formule se prevodi u neku od normalnih formi. Zbog rezolucije, posebno interesantne će biti matrice koje su u konjunktivnoj normalnoj formi.

Takođe zbog rezolucije, biće razmotrena još jedna transformacija formule u preneks formi, takozvana *skolemizacija*[4], kojom se iz formule eliminišu egzistencijalni kvantifikatori. Neka je data predikatska formula A u preneks formi (Q_1x_1) (Q_2x_2) $\ldots$ $(Q_nx_n)B$ i neka je Q_r (za $1 \leq r \leq n$) prvi po redu egzistencijalni kvantifikator. Ako su $Q_{s_1}, Q_{s_2}, \ldots, Q_{s_m}$ svi univerzalni kvantifikatori levo od Q_r u prefiksu, i $s_1 < s_2 < \ldots < s_m$, tada se promenljiva x_r u formuli A zamenjuje termom $f(x_{s_1}, x_{s_2}, \ldots, x_{s_m})$, pri čemu se simbol f ne pojavljuje u A. Ovakav funkcijski simbol se naziva *Skolemova funkcija*. Potom se iz prefiksa briše Q_rx_r, a postupak se ponavlja dok god se ne eliminišu svi egzistencijalni kvantifikatori iz formule. Skolemizacijom se od formule A u preneks formi dobija formula A_s koja je takođe u preneks formi, ali u čijem prefiksu nema egzistencijalnih kvantifikatora. Ta forma se naziva *Skolemova standardna forma*. Pri tome važi sledeće tvrđenje.

Teorema 5.5.4 Formula A zadovoljiva ako i samo ako je zadovoljiva i njena Skolemova standardna forma A_s.

Dokaz. Tvrđenje ćemo dokazati pokazujući da nakon proizvoljnog koraka zamene egzistencijalnog kvantifikatora, za polaznu formulu B i dobijenu formulu B' važi da je prva zadovoljiva ako i samo ako je to slučaj i sa drugom.
($\Leftarrow$) Neka je $B = (\forall x_1)\ldots(\forall x_n)(\exists y)C(y)$ i $B' = (\forall x_1)\ldots(\forall x_n)C(f(x_1, \ldots, x_n)/y)$, gde je C formula u preneks formi. Ako je B' zadovoljivo, to znači da za neku interpretaciju I i valuaciju v važi $I(B')_v = \top$. Neka je valuacija v' takva da su vrednosti $v'(x_1)$, $\ldots$, $v'(x_n)$ proizvoljne, dok je $v'(y) = I(f(x_1, \ldots, x_n))_v$. Sada je trivijalno $I(C(y))_{v'} = \top$. Pošto su vrednosti $v'(x_1)$, $\ldots$, $v'(x_n)$ proizvoljne, to je i $I(B)_v = \top$.
($\Rightarrow$) Neka je formula B zadovoljiva, tj. za neku interpretaciju I i valuaciju v je $I(B)_v = \top$. Tada, prema definiciji istinitosne vrednosti formula postoji valuacija v' takva da su vrednosti $v'(x_1)$, $\ldots$, $v'(x_n)$ proizvoljne, dok je $v'(y)$ takvo da je $I(C(y))_{v'} = \top$. Kako je f novi funkcijski simbol, interpretacija I nije definisana za

[4]Albert Skolem (1887 – 1963).

njega. Dodefinišimo I do interpretacija I' za koju je $I'(f(x_1,\ldots,x_n))_{v'} = v'(y)$. Opet, trivijalno sledi da je $I'(C(f(x_1,\ldots,x_n)/y))_{v'} = \top$, pa je $I'(B)_v = \top$. ■

Primer 5.5.5 Od preneks forme $(\exists x)(\exists x_1)(\forall y)(\exists z)(P(x,y)\vee Q(y,z)\vee R(x_1))$ skolemizacijom se najpre dobija formula $(\exists x_1)(\forall y)(\exists z)(P(a,y)\vee Q(y,z)\vee R(x_1))$, zatim $(\forall y)(\exists z)(P(a,y)\vee Q(y,z)\vee R(b))$, i konačno $(\forall y)(P(a,y)\vee Q(y,f(y))\vee R(b))$ koja je u Skolemovoj standardnoj formi. ■

Napomenimo da se postupak dobijanja definicione forme iz odeljka 3.3.1 može uopštiti za konstrukciju Skolemove standardne forme u kojoj je matrica u konjunktivnoj normalnoj formi.

5.6 Definiciona forma

Kao i u odeljku 3.3.1, i za predikatske logike se definiše definiciona forma za koju važi analogan teoreme 3.3.9. Zbog bogatijeg jezika sam postupak transformacije je, mada u osnovi isti, ipak nešto složeniji. U nastavku se pretpostavlja da su promenljive $x_1, x_2, \ldots, x_k$ slobodne u G, f_G simbol Skolemove funkcije i L atomska formula dobijena iz atomske formule L_H zamenom svih pojava promenljive x sa $f_G(x_1, x_2, \ldots, x_k)$:

- ako je G atomska formula, onda je
  $C_G = (\forall x_1)\ldots(\forall x_k)(L_G \vee \neg G) \wedge (\forall x_1)\ldots(\forall x_k)(\neg L_G \vee G)$,
- ako je $G = \neg H$, onda je
  $C_G = (\forall x_1)\ldots(\forall x_k)(L_G \vee L_H) \wedge (\forall x_1)\ldots(\forall x_k)(\neg L_G \vee \neg L_H)$,
- ako je $G = H \wedge J$, onda je
  $C_G = (\forall x_1)\ldots(\forall x_k)(L_G \vee \neg L_H \vee \neg L_J) \wedge (\forall x_1)\ldots(\forall x_k)(\neg L_G \vee L_H) \wedge (\neg L_G \vee L_J)$,
- ako je $G = H \vee J$, onda je
  $C_G = (\forall x_1)\ldots(\forall x_k)(L_G \vee \neg L_H) \wedge (\forall x_1)\ldots(\forall x_k)(L_G \vee \neg L_J) \wedge (\neg L_G \vee L_H \vee L_J)$.
- ako je $G = H \rightarrow J$, onda je
  $C_G = (\forall x_1)\ldots(\forall x_k)(L_G \vee L_H) \wedge (\forall x_1)\ldots(\forall x_k)(L_G \vee \neg L_J) \wedge$
  $(\forall x_1)\ldots(\forall x_k)(\neg L_G \vee \neg L_H \vee L_J)$
- ako je $G = (\exists x)H$, onda je
  $C_G = (\forall x_1)\ldots(\forall x_k)(L_G \vee \neg L_H) \wedge (\forall x_1)\ldots(\forall x_k)(\neg L_G \vee L)$ i
- ako je $G = (\forall x)H$, onda je
  $C_G = (\forall x_1)\ldots(\forall x_k)(L_G \vee \neg L) \wedge (\forall x_1)\ldots(\forall x_k)(\neg L_G \vee L_H)$.

5.7 Predikatski račun prvog reda

Formalni sistem nazvan *predikatski račun* prvog reda formalizuje predikatsku logiku prvog reda slično kao što je to bilo i u slučaju iskaznog računa L i iskazne logike. Jezik predikatskog računa se sastoji od fiksiranog dela koji predstavljaju logički

veznici $\neg$ i $\wedge$, kvantifikator $\forall$ i pomoćni simboli (leva i desna mala zagrada i zarez) i promenljivog dela koji sačinjavaju najviše prebrojivi i međusobno disjunktni skupovi relacijskih simbola, funkcijskih simbola i simbola konstanti (koje se mogu shvatiti i kao unarne funkcije). Svakom relacijskom i funkcijskom simbolu se pridružuje arnost. Preostali logički veznici i kvantifikator $\exists$ se definišu standardno, recimo $(\exists x)A$ je skraćenica za $\neg(\forall x)\neg A$. Pravila za obrazovanje terama i formula su u skladu sa definicijama 5.2.2 i 5.2.3.

Jedna aksiomatizacija predikatskog računa se sastoji od shema aksioma:

1. Svaka formula koja je izvedena iz tautologija je aksioma.

2. Ako promenljiva x nije slobodna u formuli A, aksioma je

$$(\forall x)(A \to B) \to (A \to (\forall x)B).$$

3. Ako je term t slobodan za promenljivu x u formuli A, aksioma je

$$(\forall x)A \to A(t/x).$$

i pravila izvođenja:

1. Modus ponens, tj. iz formula A i $A \to B$ izvesti B.

2. Generalizacija, tj. iz A izvesti $(\forall x)A$.

Pojmovi dokaza, teoreme, sintaksne posledice skupa formula itd. se definišu kao i u iskaznom slučaju, u odeljku 3.5. Dalje, za predikatski račun se mogu dokazati teoreme analogne teoremama o karakterizaciji za iskazni račun. U nastavku su date njihove formulacije i istaknute novine u dokazima.

Teorema 5.7.1 (Teorema dedukcije) Za rečenicu B, formulu A i skup formula T važi[5]

$$T, B \vdash A \text{ ako i samo ako } T \vdash B \to A.$$

Dokaz. Razlika u odnosu na dokaz teoreme 3.6.3 je samo u indukcijskom koraku u kome se formula A dobija iz $T \cup \{B\}$ primenom pravila generalizacije na A'. Preciznije, neka je $A = (\forall x)A'$ i neka se iz $T, B \vdash A'$ dobija $T, B \vdash A$. Prema indukcijskoj hipotezi je $T \vdash B \to A'$. Generalizacijom se dobija $T \vdash (\forall x)(B \to A')$, a kako je B zatvorena formula, primenom modus ponensa na ovu formulu i aksiomu $(\forall x)(A \to B) \to (A \to (\forall x)B)$, sledi $T \vdash B \to (\forall x)A'$, tj. $T \vdash B \to A$. ■

Teorema 5.7.2 (Korektnost) Predikatski račun je korektan u odnosu na predikatsku logiku.

[5] Pošto se u dokazu zatvorenost formule B koristi samo prilikom analize primene pravila generalizacije, na osnovu direktnog uvida u dokaz teoreme, može se zahtev za zatvorenost formule B oslabiti tako da se u izvođenju formule A iz $T \cup \{B\}$ ne sme pojaviti primenena pravila generalizacije u kojima je kvantifikovana promenljiva slobodna u formuli B.

Dokaz. Slično iskaznoj logici, dokazuje se da su sve aksiome valjane formule. Pravilo izvođenja modus pones očuvava valjanost prema teoremi 3.6.2. Konačno, ako je $\models A(x)$, prema definicijama tačnosti i valjanosti predikatskih formula, važi i $\models (\forall x)A(x)$, pa je korektno i pravilo izvođenje generalizacija. ■

Teorema 5.7.3 (Proširena kompletnost) Skup T rečenica predikatskog računa je konzistentan ako i samo ako je zadovoljiv.

Dokaz. Najpre uvedimo jedan pomoćni pojam. Neka je T skup rečenica razmatranog predikatskog jezika J. Skup T je *skup sa svedocima u* J[6] ako za svaku formulu $A(x)$ sa najviše jednom slobodnom promenljivom x važi

$$\text{ako je } \neg(\forall x)A(x) \in T, \text{ onda je } \neg A(t/x) \in T$$

za neki term t jezika J.

Dalje se dokazuje uopštenje leme Lindenbauma 3.6.13 koje glasi: ako je C' prebrojiv beskonačan skup konstanti koje ne pripadaju uočenom predikatskom jeziku J, svaki konzistentan skup rečenica T na jeziku J se može proširiti do maksimalno konzistentnog skupa rečenica T^* koji je skup sa svedocima u jeziku $J \cup C'$. Jedini novi korak u konstrukciji skupa T^* se odnosi na dodavanje skupu T_i rečenice oblika $\neg(\forall x)B(x)$. Tada se uočava konstanta $c \in C'$ koja do tog momenta nije korištena u konstrukciji. Ako skup $T_i \cup \{\neg(\forall x)B(x), \neg B(c/x)\}$ nije konzistentan, onda je $T_i, \neg(\forall x)B(x) \vdash B(c/x)$. Pošto se c ne javlja u $T_i \cup \{\neg(\forall x)B(x)\}$, sledi da je $T_i, \neg(\forall x)B(x) \vdash B(x)$ i $T_i, \neg(\forall x)B(x) \vdash (\forall x)B(x)$. Odatle je $T_i \vdash (\forall x)B(x)$, što je kontradikcija jer je T_i konzistentan, a $T_i \cup \{(\forall x)B(x)\}$ nije.

Interpretacija I_{T^*} će biti definisana tako da važi:

- domen se sastoji od svih terama jezika u kojima nema promenljivih,
- za svaki term $f(t_1, \ldots, t_n)$ u kome nema promenljivih $I_{T^*}(f(t_1, \ldots, t_n)) = f(t_1, \ldots, t_n)$ i
- za svaku atomsku formulu $P(t_1, \ldots, t_n)$ bez promenljivih $I_{T^*} \models P(t_1, \ldots, t_n)$ ako i samo ako $P(t_1, \ldots, t_n) \in T^*$.

Konačno, pokazuje se da za svaku rečenicu A važi $I_{T^*} \models A$ ako i samo ako $A \in T^*$. Recimo, ako je $A = (\forall x)B(x) \in T^*$, onda, zbog aksiome $(\forall x)B(x) \to B(t/x)$, formula $B(t/x) \in T^*$ za svaki t iz domena interpretacije, pa prema indukcijskoj hipotezi važi $I_{T^*} \models B(t/x)$ za sve elemente domena. Odatle, $I_{T^*} \models (\forall x)B(x)$. Ako $A \notin T^*$, onda, pošto je T^* skup sa svedocima, postoji neki term t iz domena takav da je $I_{T^*} \models \neg B(t/x)$. Sada direktno sledi da $I_{T^*} \not\models (\forall x)B(x)$. ■

Sledeća tvrđenja su posledice teoreme 5.7.3.

Teorema 5.7.4 (Kompaktnost) Skup rečenica T je zadovoljiv ako i samo ako je svaki njegov konačan podskup zadovoljiv.

[6] Naziv skup sa svedocima dolazi odatle što se za svaku formulu $\neg(\forall x)A(x)$ koja pripada skupu, postoji i svedok, term t, takav da i $\neg A(t/x)$ pripada skupu.

Naredne teoreme se nazivaju Skolem-Levenhaimovim[7] i odnose se na veličinu domena modela predikatskih rečenica.

Teorema 5.7.5 Ako prebrojiv skup rečenica T ima model, on ima i prebrojiv model.

Dokaz. Dovoljno je uočiti da je domen interpretacije I_{T^*} uvedene u teoremi 5.7.3 prebrojiv model za T. ■

Teorema 5.7.6 Ako skup rečenica T ima proizvoljno veliki konačan model, on ima i beskonačan model. Konzistentan skup rečenica ima model proizvoljno velike kardinalnosti veće do jednake od kardinalnosti prebrojivih skupova. Ako skup rečenica ima model kardinalnosti α i važi $\alpha \leq \beta$ za neki kardinalni broj β, tada T ima model i kardinalnosti β.

Ova tvrđenja ukazuju i na ograničenost izražajnosti predikatske logike prvog reda. Naime, u toj logici nije moguće do na izomorfizam aksiomatski opisati bilo koje beskonačne matematičke strukture, recimo aritmetiku ili realne brojeve, jer ako je njima odgovaraća teorija zadovoljiva, ona ima modele različitih kardinalnosti.

5.8 Teorije i uopštenja predikatske logike prvog reda

Preciziranjem skupa funkcijskih i relacijskih simbola i uključivanjem nekih specijalnih aksioma iz predikatskog računa nastaju *specijalni predikatski računi*, aksiomatizacije matematičkih teorija, poput teorije grupa, polja, aritmetike itd.

Teorija prvog reda na odgovarajućem jeziku je skup rečenica prvog reda tog jezika. Ponekada se umesto teorija prvog reda kaže i *elementarna teorija*. Teorija prvog reda T je *kompletna* ako za svaku rečenicu A na jeziku teorije važi ili $T \vdash A$ ili $T \vdash \neg A$. Teorija prvog reda T je *kategorična* ako su joj svi modeli (za jezik te teorije) izomorfni.

Primer 5.8.1 *Teorija brojeva* ili elementarna Peanova[8] aritmetika PA je jednakosna teorija čiji jezik sadrži binarni relacijski znak $=$, binarne funkcijske simbole $+$ i $\cdot$, unarni funkcijski simbol $'$ i konstantu 0, a pored jednakosnih aksioma $((\forall x)x = x$, $(\forall x)(\forall y)(x = y \to (A(x) \to A(y/x))))$, specijalne aksiome su:

1. $(\forall x)\neg(0 = x')$
2. $(\forall x)(\forall y)(x' = y' \to x = y)$
3. $(\forall x)x + 0 = x$
4. $(\forall x)(\forall y)(x + (y') = (x + y)')$
5. $(\forall x)x \cdot 0 = 0$

[7] Leopold Löwenheim (1878 – 1957).
[8] Giuseppe Peano(1858 – 1932).

6. $(\forall x)(\forall y)(x \cdot (y') = (x \cdot y) + x)$

7. za svaku formulu $A(x_0, \ldots, x_n)$ u kojoj se x_0 ne javlja vezano u A, aksioma indukcije je

$$\begin{aligned}(A(0/x_0, x_1, \ldots, x_n) \quad &\wedge \quad (\forall x_0)(A(x_0, \ldots, x_n) \rightarrow A(x_0'/x_0, \ldots, x_n))) \\ &\rightarrow \quad (\forall x_0)A(x_0, \ldots, x_n)\end{aligned}$$

Standardni model teorije brojeva je struktura $\langle \mathbb{N}, +, \cdot, ', 0\rangle$, gde je $\mathbb{N}$ skup prirodnih brojeva i u kojoj 0, + i $\cdot$ imaju uobičajeno značenje, a $'$ je operacija naslednika. Pretpostavka da je standardni model zaista normalni model za teoriju brojeva dovodi do neobičnih posledica. Na osnovu teoreme 5.7.6 tada proizilazi da postoje normalni modeli teorije veće kardinalnosti od standardnog modela koji je prebrojiv. Takođe, postoje i prebrojivi modeli koji nisu izomorfni standardnom modelu. Svi ti modeli koji nisu izomorfni standardnom modelu nazivaju se *nestandardni*.

Pokazuje se da se rekurzivne funkcije mogu predstaviti[9] u formalnom sistemu Q, takozvanoj Robinsonovoj aritmetici. Q nastaje kada se iz PA odstrani aksioma indukcije. Pod pretpostavkom da su Q i PA konzistentni, zbog neodlučivosti halting problema ni ovi sistemi nisu odlučivi. Kako je svaka rečenica na razmatranom jeziku bilo tačna, bilo netačna, kada se razmatra u strukturi prirodnih brojeva, zaključujemo da skup istinitih rečenice prvog reda koje govore o prirodnim brojevima nije rekurzivno aksiomatizabilan ni sa Q, ni sa PA. Ovo je posledica činjenice da bismo u slučaju postojanja rekurzivne aksiomatizacije mogli nabrojati sve dokaze, tako da bismo u tom nabrajanju stigli ili do A ili do $\neg A$ za svaku rečenicu A, te bi Q i PA bile odlučive. U vezi sa ovim razmatranjima su čuvene Gedelove teoreme o nepotpunosti. Prva teorema upravo kaže da se istinite rečenice prvog reda o prirodnim brojevima ne mogu rekurzivno aksiomatizovati, dok se u drugoj teoremi pokazuje da se konzistentnost sistema PA ne može dokazati u okviru samog tog sistema. Zanimljivo je da postoje i kompletni fragmenti teorije brojeva, recimo teorija čiji jezik je skup $\{=, 0, S\}$ i takozvana Presburgerova aritmetika, ili aditivna aritmetika, čiji jezik je skup $\{=, +, 0, S\}$. ■

Kao što smo videli, logika prvog reda ima određena ograničenja koja su posledica nedovoljne izražajnosti. Zato se razvijaju njena proširenja u kojima se na pogodan način mogu izraziti izvesni pojmovi. Recimo, *logika drugog reda* dozvoljava kvantifikovanje ne samo promenljivih koje odgovaraju pojedinačnim objektima, već i promenljivih koje se odnose na relacije. Tako se formula $(\forall P)P(a, b)$ shvata kao: 'za svaku binarnu relaciju važi da su a i b u toj relaciji'. *Beskonačna logika* $L_{\omega_1\omega}$ dozvoljava formule koje sadrže ne samo konačne, već i prebrojive konjunkcije i disjunkcije. Na primer, ako je T prebrojiv skup formula, formula je i $\wedge_{A_i \in T} A_i$. Kod *logika sa uopštenim kvantifikatorima* pored univerzalnog i egzistencijalnog postoje i novi kvantifikatori oblika (Qx) koji se mogu tumačiti na razne načine, recimo kao beskonačno mnogo, neprebrojivo mnogo itd.

[9] Za svaku parcijalno rekurzivnu funkciju $f(x_1, \ldots, x_n)$ postoji formula $A(x_1, \ldots, x_n, y)$ na jeziku aritmetike sa $n+1$ slobodnom promenljivom takva da za proizvoljne prirodne brojeve a_1, $\ldots$, a_n i b, $f(a_1, \ldots, a_n) \downarrow b$ ako i samo ako $\vdash A(0^{a_1}, \ldots, 0^{a_n}, y) \leftrightarrow y = 0^b$, gde je 0^k skraćeni zapis terma u kome je k puta funkcijski simbol $'$ primenjen na 0.

Ova proširenja su često veoma izražajna. Na primer, postoji rečenica drugog reda čije su posledice tačno one rečenice prvog i drugog reda koje važe u $\mathbb{N}$. Ili, postoji rečenica drugog reda čiji su svi modeli izomorfni sa $\mathbb{N}$ itd. Međutim, te logike imaju i neke osobine (nemaju rekurzivnu aksiomatizaciju, ne važi teorema kompaktnosti itd.) zbog kojih nisu pogodne za primene.

5.9 (Ne)odlučivost u predikatskoj logici prvog reda

U odeljku 3.5 je rečeno da je problem da li je formula teorema rekurzivno aksiomatizabilnog formalnog sistema barem parcijalno odlučiv. Zbog teoreme kompletnosti 5.7.3 isto važi i za problem da li je neka predikatska formula valjana. Ali, dok je za iskaznu logiku taj problem i odlučiv, ovde to nije slučaj, čime se daje negativan odgovor na Hilbertov Entscheidungsproblem. U narednim teoremama prodiskutovaćemo odlučivost problema valjanosti i njemu srodnog problema zadovoljivosti i u nekim fragmentima predikatske logike.

Teorema 5.9.1 Problemi valjanosti i zadovoljivosti rečenica u predikatskoj logici nisu odlučivi.

Dokaz. U dokazu se koristi tvrđenje iz primera 2.8.2 u kome je pokazano da problem da li je funkcija definisana za neki argument nije odlučiv. Zbog jednakosti klasa parcijalno izračunljivih funkcija i Tjuring-izračunljivih funkcija koja je pokazana u odeljku 2.5 ni problem zaustavljanja Tjuringovih mašina (odnosno halting problem) nije odlučiv.

Za proizvoljnu determinističku Tjuringovu mašinu M i njen ulazni podatak a posmatraćemo rečenicu $A_{M(a)}$ koja će biti valjana ako i samo ako je $M(a)\downarrow$, zbog čega problem valjanosti rečenica predikatske logike neće biti odlučiv.

Formula $A_{M(a)}$ će biti oblika $B \to C$. Takođe ćemo fiksirati i interpretaciju I pri kojoj se formule B i C shvataju kao opis Tjuringove mašine $M(a)$, odnosno opis situacije u kojoj $M(a)$ dolazi u završno stanje.

Pretpostavimo da traka mašine nije ograničena ni sa jedne strane i da se u početnom stanju q_0 glava nalazi nad najlevljim znakom ulaznog podatka koji je upisan u ćeliji koja je označena brojem 0. Neka su $S = \{q_0, \ldots, q_n, q_z\}$ skup stanja i $A = \{1\}$ alfabet mašine M, dok 0 predstavlja blanko znak. Znaci koji se javljaju u formuli $A_{M(a)}$ su:

- binarni relacijski simboli Q_i, $i \in \{0, \ldots, n, z\}$, A_0, A_1, $=$, $\neq$ i $<$,
- simbol konstante 0 i
- unarni funkcijski simbol $'$.

Oznaku 0^k ćemo koristiti kao skraćenicu za term u kome je funkcijski simbol $'$ primenjen k puta na simbol konstante 0. U jednom delu dokaza se koristi interpretacija $I = \langle \mathbb{Z}, \psi \rangle$ koja na svojevrstan način opisuje rad mašine $M(a)$. Domen interpretacije je skup celih brojeva, dok se simboli formule interpretiraju tako da:

- $\psi(Q_i)$ je skup svih parova (t, x) takvih da je mašina M u koraku t rada u stanju q_i nad ćelijom trake označenom sa x,
- $\psi(A_j)$ je skup svih parova (t, x) takvih da je u koraku t rada mašine M u ćeliji x trake znak $a_j \in \{0, 1\}$,
- $=, \neq$ i $<$ su redom relacije jednakosti, nejednakosti i biti manje,
- $\psi(0) = 0$ i
- unarni funkcijski simbol $'$ predstavlja funkciju naslednika.

Najpre ćemo opisati konstrukciju formule $A_{M(a)} = B \to C$.

Za svaku naredbu mašine M oblika $q_i\ a_j\ a_k\ q_m$, posmatra se formula

$$\begin{aligned}(\forall t)(\forall x)(\forall y)((Q_i(t,x) \wedge A_j(t,x)) \quad &\to \\ (Q_m(t',x) \wedge A_k(t',x) \quad &\wedge \quad (y \neq x \to (\wedge_{l=0}^{1}(A_l(t,y) \to A_l(t',y))))))\end{aligned}$$

koja se pri interpretaciji I shvata kao: "ako se u koraku t rada glava mašine u stanju q_i nalazi nad ćelijom x u kojoj je upisan znak a_j, onda je u koraku $t+1$ mašina u stanju q_m, glava joj se nalazi iznad iste ćelije u koju je sada upisan znak a_k, dok se sadržaj svih ostalih ćelija ne menja".

Za svaku naredbu mašine M oblika $q_i\ a_j\ R\ q_m$ posmatra se formula

$$(\forall t)(\forall x)(\forall y)((Q_i(t,x) \wedge A_j(t,x)) \to (Q_m(t',x') \wedge (\wedge_{l=0}^{1}(A_l(t,y) \to A_l(t',y)))))$$

koja se pri interpretaciji I shvata kao: "ako se u koraku t rada glava mašine u stanju q_i nalazi nad ćelijom x u kojoj je upisan znak a_j, onda je u koraku $t+1$ mašina u stanju q_m, glava joj se nalazi iznad ćelije $x+1$, a sadržaj ni jedne ćelije se ne menja".

Za svaku naredbu mašine M oblika $q_i\ a_j\ L\ q_m$ posmatra se formula

$$(\forall t)(\forall x)(\forall y)((Q_i(t,x') \wedge A_j(t,x')) \to (Q_m(t',x) \wedge (\wedge_{l=0}^{1}(A_l(t,y) \to A_l(t',y)))))$$

koja se interpretira slično prethodnoj vrsti formula.

Sledeća formula opisuje konfiguraciju mašine u početnom koraku rada:

$$\begin{aligned}Q_0(0,0) \quad &\wedge \quad A_{i_0}(0,0) \wedge \ldots \wedge A_{i_{n-1}}(0, 0^{(n-1)}) \\ &\wedge \quad (\forall y)((y \neq 0 \wedge \ldots \wedge y \neq 0^{(n-1)}) \to A_0(0,y)).\end{aligned}$$

dok se narednom formulom kaže da je svaki celi broj naslednik tačno jednog celog broja:

$$(\forall z)(\exists x)(z = x') \wedge (\forall z)(\forall x)(\forall y)((z = x' \wedge z = y') \to (x = y)).$$

Konačno, formula

$$\begin{aligned}(\forall x)(\forall y)(\forall z)((x < y \wedge y < z) \quad &\to \quad (x < z)) \wedge \\ (\forall x)(\forall y)(x' = y \to x < y) \quad &\wedge \quad (\forall x)(\forall y)(x < y \to x \neq y).\end{aligned}$$

garantuje da je za $p \neq q$ ispunjeno $(\forall x)x^{(p)} \neq x^{(q)}$.

Formula B je konjunkcija svih prethodnih formula, dok se formulom C:

$$(\exists t)(\exists x)Q_z(t,x)$$

kaže da postoji korak rada u kome se mašina zaustavlja.

Sada pokažimo da je formula $A_{M(a)} = B \to C$ valjana ako i samo ako $M(a)\downarrow$. Pretpostavimo najpre da je $\models B \to C$. Očigledno je da su sve formule koje formiraju B tačne pri interpretaciji I, pa kako je $I \models B \to C$, mora biti i $I \models C$, što znači da, ako je formula $A_{M(a)}$ zadovoljiva pri interpretaciji I, mašina M se zaustavlja. Obrnuto, pretpostavimo da se mašina zaustavlja, što znači da postoji konačni niz konfiguracija koje predstavljaju izračunavanje mašine M za ulazni podatak x. Za svaki korak rada p, odgovarajuća konfiguracija se opisuje formulom δ_p oblika:

$$\begin{aligned} Q_i(0^p,0^q) \wedge A_{j_1}(0^p,0^{q_1}) \wedge \ldots \wedge A_{j_k}(0^p,0^{q_k}) \quad &\wedge \\ (\forall y)((y \neq q_1 \wedge \ldots \wedge y \neq q_k) \quad &\to \quad A_0(0^p,y)) \end{aligned}$$

kojom se precizira aktuelno stanje, položaj glave i sadržaj trake. Indukcijom po broju koraka rada se pokazuje da, ako mašina $M(a)$ ne staje pre koraka s u radu, onda je $B \models \delta_s$, odnosno $\models B \to \delta_s$. Za korak $s = 0$, ovo tvrđenje važi trivijalno jer je formula δ_0 deo formule B. Iz indukcijske pretpostavke tvrđenje se izvodi analizom delova formule B i odgovarajućih naredbi koje menjaju konfiguraciju. Recimo, formula

$$(\forall t)(\forall x)(\forall y)((Q_i(t,x) \wedge A_j(t,x)) \to (Q_m(t',x') \wedge (\wedge_{l=0}^{1}(A_l(t,y) \to A_l(t',y))))$$

koja se odnosi na pomeranje glave u desno, zajedno sa opisom tekuće konfiguracije kao posledicu ima opis naredne konfiguracije, a slično je i za ostale slučajeve. Pretpostavimo da se mašina $M(a)$ zaustavlja nakon s koraka rada. To znači da je $\models B \to \delta_s$. Jedan konjunkt u formuli δ_s je oblika $Q_z(0^s,0^p)$, pa direktno sledi da je $\models B \to (\exists t)(\exists x)Q_z(t,x)$, odnosno $\models A_{M(a)}$.

Dakle, problem valjanosti rečenica predikatskog računa prvog reda nije odlučiv. Kako je za rečenicu A, $\models A$ ekvivalentno sa $\neg A$ nije zadovoljivo, isto se odnosi i na problem zadovoljivosti rečenica. ■

Teorema 5.9.2 Problem valjanosti rečenica u predikatskoj logici je parcijalno odlučiv, dok je problem zadovoljivosti rečenica koparcijalno odlučiv.

Dokaz. U teoremi 5.9.1 se pokazuje da problemi valjanosti i zadovoljivosti formula predikatskog računa nisu odlučivi. Kako je predikatski račun rekurzivno aksiomatizabilan, prema rečenom u odeljku 3.5, problem valjanosti je parcijalno odlučiv. Pošto je zatvorena formula valjana ako i samo ako je njena negacija kontradikcija, to je problem zadovoljivosti komplement problema valjanosti, tj. on je koparcijalno odlučiv. ■

Teorema 5.9.3 Ako jezik predikatske logike, pored logičkih simbola, sadrži samo jedan binarni relacijski simbol, problemi valjanosti i zadovoljivosti nisu odlučivi.

Dokaz. Primetimo da se u formuli $A_{M(a)}$ iz dokaza teoreme 5.9.1 pojavljuju samo neki binarni relacijski simboli, znak $=$, simbol konstante 0 i jedan unarni funkcijski simbol koji se interpretira kao naslednik. Pokazuje se da postoji rečenica u kojoj se od nelogičkih simbola javljaju samo binarni relacijski simboli i koja je ekvivalentna sa $A_{M(a)}$, a zatim i da se ta rečenica može ekvivalentno transformisati u rečenicu koja sadrži samo jedan binarni relacijski simbol. ■

Međutim, ako je jedini binarni relacijski simbol u formuli $=$, i razmatraju se samo normalni modeli u kojima se on interpretira kao jednakost, situacija se menja.

Teorema 5.9.4 Ako jezik predikatske logike, pored logičkih simbola, sadrži samo binarni relacijski simbol $=$ i unarne relacijske simbole, problemi valjanosti i zadovoljivosti rečenica u normalnim modelima su odlučivi.

Dokaz. Pokazuje se da je rečenica razmatrane logike koja ima k unarnih predikatskih simbola i r promenljivih zadovoljiva ako i samo ako je zadovoljiva pri nekoj interpretaciji čiji domen ima najviše $2^k \cdot r$ elemenata. Broj interpretacija sa takvim domenom koje se razlikuju u značenju koje daju simbolima koji se javljaju u nekoj konkretnoj formuli je konačan, pa je problem zadovoljivosti odlučiv, a time i njegov komplement, problem valjanosti. ■

Posledice teoreme 5.9.4 su odlučivost čiste jednakosne logike i unarne logike, jer:

- kada rečenica od nelogičkih simbola sadrži samo $=$, tj. $k = 0$ u formulaciji teoreme 5.9.4, rečenica je zadovoljiva ako i samo ako je zadovoljiva u modelu čiji domen ima r elemenata, gde je r broj promenljivih u rečenici,
- rečenica koja od nelogičkih simbola sadrži samo k unarnih relacijskih simbola je zadovoljiva ako i samo ako je zadovoljiva u modelu čiji domen ima 2^k elemenata, jer se pokazuje da je takva rečenica ekvivalentna rečenici na istom jeziku sa samo jednom promenljivom, pa je $r = 1$ u formulaciji teoreme 5.9.4.

U teoremi 5.10.9 je dat drugačiji dokaz potpunosti za unarnu predikatsku logiku. Pored upravo spomenutih problema, pokazano je da za predikatsku logiku važi i:

- problem valjanosti rečenica oblika $(\forall x_1) \ldots (\forall x_m) A(x_1, \ldots, x_m)$ u kojima nema kvantifikatora u A je odlučiv jer je ovakva formula valjana ako i samo ako važi u svim modelima čiji domeni sadrže m elemenata,
- problem valjanosti rečenica oblika $(\exists x_1) \ldots (\exists x_m) A(x_1, \ldots, x_m)$ u kojima nema kvantifikatora u A je odlučiv jer je ovakva formula valjana ako i samo ako važi u svim modelima čiji domeni sadrže 1 element,
- problem valjanosti rečenica oblika $(\forall x_1) \ldots (\forall x_m)(\exists y_1) \ldots (\exists y_n) A(x_1, \ldots, x_m, y1, \ldots, y_n)$ u kojima nema kvantifikatora u A je odlučiv jer je ovakva formula valjana ako i samo ako važi u svim modelima čiji domeni sadrže najviše m elemenata,
- problem zadovoljivosti rečenica koje u preneks formi imaju prefiks oblika $(\exists y_1) \ldots (\exists y_m)(\forall x_1)(\forall x_2)(\exists z_1) \ldots (\exists z_k)$ i problem valjanosti rečenica koje u preneks formi imaju prefiks oblika $(\forall y_1) \ldots (\forall y_m)(\exists x_1)(\exists x_2)(\forall z_1) \ldots (\forall z_k)$ su odlučivi itd.

5.10 Erbranova teorema

Ako je za interpretaciju $I = \langle\langle Dom, Rel, Fun\rangle, \psi\rangle$, $Dom = \{c_0, c_1, \ldots\}$, mogla bi se tačnost formule oblika $(\forall x)B(x)$ pri interpretaciji I svesti na tačnost u Dom formula oblika $\psi(B)(c_i)$. Preciznije, formula $(\forall x)B(x)$ bi bila tačna ako i samo ako je tačna jedna konjunkcija oblika $\psi(B)(c_0) \wedge \psi(B)(c_1) \wedge \ldots \wedge \psi(B)(c_n) \wedge \ldots$ Slično, formula $(\exists x)B(x)$ bi bila tačna ako i samo ako je tačna jedna disjunkcija oblika $\psi(B)(c_0) \vee \psi(B)(c_1) \vee \ldots \vee \psi(B)(c_n) \vee \ldots$ Zato se kaže da je univerzalno kvantifikovana formula uopštenje konjunkcije, dok je egzistencijalno kvantifikovana formula uopštenje disjunkcije. Istinitosna tablica za formulu $(\forall x)P(x)$, shvaćenu kao beskonačna konjunkcija, bi u opštem slučaju za interpretaciju sa beskonačnim domenom sadržala red beskonačne dužine. Kako ni broj različitih interpretacija neke predikatske formule nije konačan, za većinu predikatskih formula istinitosne tablice bi morale sadržati beskonačno mnogo redova. Zbog toga tablična metoda nije u opštem slučaju primenljiva za predikatske formule, a situacija je još gora kada se uzme u obzir da domeni mogu biti i neprebrojivi skupovi. Na prvi pogled može izgledati da je ovo razmatranje nebitno u okviru računarske realizacije, gde su svi domeni sigurno konačni. Međutim, ne treba zaboraviti da su skupovi koji se implementiraju na računarima (kao skup realnih brojeva, recimo) tek aproksimacije često beskonačnih skupova sa kojima se barata u matematici i da se istinitosni pojmovi definišu koristeći upravo takve idealizovane skupove. Recimo, postoje formule koje nikada nisu tačne na konačnim domenima, ali jesu tačne na nekim beskonačnim domenima. Zato u razmatranjima moramo koristiti i skupove koji nisu konačni.

Primer 5.10.1 Posmatrajmo formulu $(\forall x)(P(x, f(x)) \wedge \neg P(x, x)) \wedge (\forall x)(\forall y)(\forall z)$ $((P(x, y) \wedge P(y, z)) \rightarrow P(x, z))$. Ona je zadovoljena pri interpretaciji čiji je domen skup pozitivnih celih brojeva, gde se funkcijski simbol f tumači kao naslednik, a relacijski simbol P kao manje. U proizvoljnoj interpretaciji I koja je model za formulu, interpretacija relacijskog simbola P mora biti tranzitivna irefleksivna relacija. Zato interpretacija funkcijskog simbola f mora biti unarna funkcija koja uvek daje nove elemente domene, jer bi se u suprotnom napravio ciklus i za neki element domena dobilo da je u relaciji sa samim sobom. Dakle, ova formula nije zadovoljiva ni pri jednoj interpretaciji koja ima konačan domen. ■

Jasno je da je predikatska logika izražajnija od iskazne i da se kvantifikatori ne mogu izraziti konačnim iskaznim sredstvima. Interpretacija za neku formulu ima beskonačno mnogo sa međusobno različitim domenima. Postavlja se pitanje kako utvrditi da li je neka formula zadovoljiva. U odeljku 5.9 je pokazano da taj problem nije odlučiv. Međutim, u nastavku teksta ćemo pokazati da se zadovoljivost svake predikatske formule ipak može svesti na zadovoljivost iskaznih formula kojih, doduše, može biti i beskonačno puno. Pri tome biće opisan sistematski postupak za nalaženje eventualnih modela za predikatsku formulu.

Definicija 5.10.2 Neka su A rečenica u Skolemovoj standardnoj formi i C_A skup konstanti koje se javljaju u A. Ako je $C_A = \emptyset$, skupu C_A dodajemo konstantu λ. *Erbranov univerzum* $D(A)$ sadrži

1. sve konstante iz C_A i

2. ako je f funkcijski simbol koji se pojavljuje u formuli A čija je arnost n i $t_1, \ldots, t_n \in D(A)$, tada je i $f(t_1, \ldots, t_n) \in D(A)$.

Za rečenicu $A = (\forall x_1) \ldots (\forall x_n) A^*(x_1, \ldots, x_n)$ u Skolemovoj standardnoj formi *Erbranova ekspanzija* je $E(A) = \{A^*(t_1/x_1, \ldots, t_n/x_n) : t_1, \ldots, t_n \in D(A)\}$.

Primer 5.10.3 Ako je formula $A = (\forall x)(P(x, c) \to Q(x))$, onda je $D(A) = \{c\}$ konačan. Ako je formula $B = (\forall x)(P(x, f(x), c) \to Q(g(x, x)))$, onda je $D(B) = \{c, f(c), g(c, c), f(f(c)), f(g(c, c)), g(c, f(c)), g(f(c), f(c)), \ldots\}$ beskonačan. Ako je formula $B = (\forall x)(P(x, f(x), x) \to Q(g(x, x)))$, u kojoj nema konstanti onda je $D(B) = \{\lambda, f(\lambda), g(\lambda, \lambda), f(f(\lambda)), f(g(\lambda, \lambda)), g(\lambda, f(\lambda)), g(f(\lambda), f(\lambda)), \ldots\}$. ■

Primer 5.10.4 Za formulu $A = (\forall x)(P(x, c) \to Q(x))$ Erbranova ekspanzija je $E(A) = \{P(c, c) \to Q(c)\}$. Za formulu $B = (\forall x)(P(x, f(x), c) \to Q(g(x, x)))$ Erbranova ekspanzija je $E(B) = \{(P(c, f(c), c) \to Q(g(c, c))), (P(f(c), f(f(c)), c) \to Q(g(f(c), f(c)))), (P(g(c, c), f(g(c, c)), c) \to Q(g(g(c, c), g(c, c)))), \ldots\}$. ■

Definicija 5.10.5 Za rečenicu A u Skolemovoj standardnoj formi *Erbranova interpretacija* je bilo koja interpretacija $I = \langle \langle Dom, \ldots \rangle, \psi \rangle$ koja zadovoljava:

- $Dom = D(A)$ i
- za svaki funkcijski simbol f arnosti n koji se pojavljuje u formuli A i terme $t_1, \ldots t_n \in D(A)$, za proizvoljnu valuaciju v je $I(f(t_1, \ldots t_n))_v = f(t_1, \ldots t_n)$.

Dakle, za Erbranove interpretacije jedino ne preciziramo značenje relacijskih simbola, dok se značenje funkcijskih simbola opisuje sintaksno tako da je vrednost terma bez promenljivih on sam, slično kao što se radi u dokazu teoreme 5.7.3 o potpunosti predikatskog računa prvog reda. Sledeće teoreme dovode u vezu zadovoljivost formule, Erbranov univerzum, Erbranovu ekspanziju i Ebranovu interpretaciju.

Teorema 5.10.6 Rečenica A u Skolemovoj standardnoj formi je zadovoljiva ako i samo ako je zadovoljiva pri nekoj Erbranovoj interpetaciji I.

Dokaz. ($\Rightarrow$) Ako je za neku Erbranovu interpetaciju I, $I \models A$, formula A je zadovoljiva.
($\Leftarrow$) Neka je $I = \langle Dom_I, \Psi_I \rangle$ proizvolja interpretacija takva da je $I \models A$. Definisaćemo Erbranovu interpretaciju $I_A = \langle D(A), \psi_A \rangle$ koja će zadovoljavati formulu A. Najpre, ako formula A ne sadrži simbole konstanti, za specijalnu konstantu ćemo proizvoljno definisati $\psi_A(\lambda) \in Dom_I$. Zatim ćemo definisati interpretacije relacijskih simbola tako da je

$$(t_1, \ldots t_n) \in \psi_A(P) \text{ ako i samo ako } (I(t_1), \ldots I(t_n)) \in \psi_I(P)$$

za sve terme bez promenljivih $t_1, \ldots, t_n \in D(A)$ i relacijske simbole P koji se javljaju u formuli A. Dalje ćemo indukcijom po broju univerzalnih kvantifikatora pokazati da za svaku zatvorenu formulu B u preneks formi koja je sastavljena od istih funkcijskih i relacijskih simbola kao i formula A važi ako $I \models B$, onda $I_A \models B$. Ako je $n = 0$, formula B nema univerzalnih kvantifikatora, pa ni promenljivih,

odakle po definiciji interpretacije I_A sledi tvrđenje. Neka tvrđenje važi za svaki $n < k$. Neka je $B = (\forall x)C$ i C ima $k-1$ univerzalni kvantifikator, ali se x javlja slobodno u C. Kako $I \models (\forall x)C$, to znači da za svaku valuaciju v, $I(C)_v = \top$, pa i za svaku valuaciju v' takvu da je $v'(x) = I(t)$ za svaki term $t \in D(A)$. Odatle, $I_A(C)_w = \top$ za svaku valuaciju w sa kodomenom $D(A)$, pa je $I_A \models (\forall x)C$, odnosno $I_A \models B$.

Konačno, pošto je i A formula opisnog oblika, važi $I_A \models A$. ■

Teorema 5.10.7 Rečenica A u Skolemovoj standardnoj formi je zadovoljiva ako i samo ako je zadovoljiva Erbranova ekspanzija $E(A)$.

Dokaz. Neka je $A = (\forall x_1) \dots (\forall x_n)B$, gde je B formula bez kvantifikatora. Dokaz ćemo sprovesti tako što ćemo pokazati da je A zadovoljivo pri nekoj Erbranovoj interpretaciji I_A ako i samo ako je $E(A)$ zadovoljivo. Očigledno je $I_A \models A$ ako i samo ako za bilo koju valuaciju v sa domenom $D(A)$ važi $I_A(B)_v = \top$ ako i samo ako $I_A(B(t_1/x_1, \dots, t_n/x_n)) = \top$ za sve $t_1, \dots, t_n \in D(A)$ ako i samo $I_A \models E(A)$. Na osnovu teoreme 5.10.6 to znači da je A zadovoljiva ako i samo ako je zadovoljiva Erbranova ekspanzija $E(A)$. ■

Primetimo da formule iz $E(A)$ možemo shvatiti kao iskazne formule pošto su sastavljene samo od atomskih formula bez promenljivih. Odatle je moguće izvesti najavljenu aproksimaciju rečenica skupovima iskaznih formula. Naime, rečenicu prevodimo u Skolemovu standardnu formu koja je zadovoljiva ako i samo ako je polazna rečenica zadovoljiva, a zadovoljivost Skolemove standardne forme se na osnovu teoreme 5.10.7 svodi na zadovoljivost skupa iskaznih formula. Konačno, primenom teoreme kompaktnosti za iskaznu logiku dobijamo

Teorema 5.10.8 (Erbranova teorema) Rečenica A koja je u Skolemovoj standardnoj formi nije zadovoljiva ako i samo ako postoji konačan podskup Erbranove ekspanzije $E(A)$ koji nije zadovoljiv.

Jedna procedura ispitivanja (ne)zadovoljivosti formule izvodi se na sledeći način. Najpre se formula transformiše u Skolemovu standardnu formu, a zatim se deo po deo konstruiše odgovarajuća Erbranova ekspanzija. U svakom koraku se ispituje zadovoljivost dobijene delimične ekspanzije nekom metodom ispitivanja zadovoljivosti skupova iskaznih formula. Ako se u bilo kom koraku dobije da delimična ekspanzija nije zadovoljiva, onda ni polazna formula nije zadovoljiva. Ako je Erbranova ekspanzija formule konačna ili će se pokazati da je zadovoljiva ili da nije zadovoljiva i na osnovu toga ćemo zaključiti o (ne)zadovoljivosti polazne formule. Ako polazna formula nije zadovoljiva, a Erbranova ekspanzija nije konačna, pre ili posle će to biti slučaj i sa nekom delimičnom ekspanzijom, pa ako računar na kome se radi ima dovoljno memorije, to će se i efektivno pokazati. Međutim, ako je formula zadovoljiva, a Erbranova ekspanzija beskonačna, odgovor nikada nećemo dobiti.

Kako je proizvoljna predikatska formula valjana ako i samo ako je njeno univerzalno zatvorenje valjano, prethodni postupak se može uopštiti i na proizvoljne formule.

Teorema 5.10.9 Odlučivi su problemi valjanosti i zadovoljivosti formula predikatske logike u kojoj su jedini nelogički simboli unarni relacijski simboli i simboli konstanti.

Dokaz. Odlučivost proizilazi iz činjenice da je Erbranov univerzum formule takve logike konačan, jer se u matrici Skolemove standarne forme ne javljaju funkcijski simboli. Kako je tada i Erbranova ekspanzija formule konačna, kao i u iskaznoj logici, problem njene (ne)zadovoljivosti je odlučiv. ■

5.11 Rezolucija u predikatskoj logici

U Erbranovoj teoremi 5.10.8 je pokazano da se za neku formulu valjanost, odnosno nezadovljivost, može utvrditi ispitivanjem nezadovoljivosti skupova iskaznih formula. Primetimo da se pri tom, zbog svođenja na definicionu ili konjunktivnu normalnu formu, možemo ograničiti na skupove iskaznih klauza. Prema teoremi o rezoluciji u iskaznoj logici 3.7.6, skup iskaznih klauza je nezadovoljiv ako i samo ako se iz njega metodom rezolucije može izvesti prazna klauza. U takvom jednom postupku će se najpre generisati mnoge slične klauze, a zatim bi se dobili i mnogi slični njihovi potomci, kao u primeru 5.11.1.

Primer 5.11.1 Neka je formula $A = (\forall x)(\exists z)(\forall y)((P(x,y) \vee Q(x)) \wedge \neg P(x,z))$. Tada je matrica formule u konjunktivnoj normalnoj formi $A^* = (P(x,y) \vee Q(x)) \wedge \neg P(x, f(x))$, Erbranov univerzum je $D(A) = \{\lambda, f(\lambda), f(f(\lambda)), \ldots,\}$, a Erbranova ekspanzija u obliku skupa klauza $E(A) = \{\{P(\lambda,\lambda), Q(\lambda)\}, \{\neg P(\lambda, f(\lambda))\}, \{P(\lambda, f(\lambda)), Q(\lambda)\}, \{\neg P(f(\lambda), f(f(\lambda)))\}, \{P(f(\lambda), f(f(\lambda))), Q(f(\lambda))\}, \ldots\}$. Iskazna rezolucija primenjena na klauze $\{P(f^n(\lambda), f^{n+1}(\lambda)), Q(f^n(\lambda))\}$ i $\{\neg P(f^n(\lambda), f^{n+1}(\lambda))\}$ daje klauzu $\{Q(f^n(\lambda))\}$, za svako n. ■

U ovom odeljku ćemo opisati pravilo rezolucije za predikatske formule kojim se pokušava da se mnoge primene pravila iskazne rezolucije izvedu odjednom. Koristićemo definicije literala i klauze iz odeljka 3.7 imajući u vidu da atomske formule više nisu iskazna slova.

Umesto da posmatramo Erbranovu ekspanziju iz primera 5.11.1, koristimo samo matricu shvaćenu kao skup klauza $A^* = \{\{P(x,y), Q(x)\}, \{\neg P(x, f(x))\}\}$. Literal $P(x,y)$ nije negacija literala $\neg P(x, f(x))$. Pošto su promenljive x i y univerzalno kvantifikovane, podrazumevamo da umesto njih u prvom i drugom literalu mogu stajati λ, $f(\lambda)$ itd. Nakon toga neki primerci literala jesu međusobne negacije. Dakle, potrebno je nekako dovesti u vezu, tj. ujednačiti, formule $P(x,y)$ i $\neg P(x, f(x))$. Takav postupak se naziva unifikacija. Neformalno rečeno, unifikujući neke formule podrazumeva se da su pojedini primerci formula isti, kao što se zamenom promenljivih x i y konstantama λ i $f(\lambda)$, od literala $P(x,y)$ i $\neg P(x, f(x))$ dobijaju primerci $P(\lambda, f(\lambda))$ i $\neg P(\lambda, f(\lambda))$ na koje se može primeniti iskazna rezolucija.

5.11.1 Unifikacija

Unifikacija je jedan vid zamene koja je do sada često spominjana. U zamenama, a time i unifikacijama, se uvek zamenjuje promenljiva nekim termom, što se može

shvatiti i kao dodeljivanje vrednosti promenljivoj ili, u programerskoj terminologiji, kao prenos argumenata. Zamena terma koji nije promenljiva nekim termom nije dozvoljena. Sledeća definicija precizira pojam zamene.

Definicija 5.11.2 *Zamena* je svaki konačan skup oblika $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$, gde su za svaki i, x_i promenljiva i t_i term različit od x_i, a sve promenljive x_i su međusobno različite. Svaki element t_i/x_i ovog skup je komponenta zamene i shvata se kao da je t_i *zamena za* x_i. *Primerak* literala L pri zameni θ (u oznaci $L\theta$) se dobija jednovremenom zamenom svih x_i odgovarajućim t_i u literalu L.

Primer 5.11.3 Za zamenu $\theta = \{a/x\}$ i literal $P(x)$, primerak $P\theta$ je $P(a)$. Za zamenu $\rho = \{f(z)/x, z/y\}$ i literal $P(x, f(y), a)$, $P\rho = P(f(z), f(z), a)$. ■

Slično primercima literala uvode se primerci klauza, formula, njihovih skupova itd. U postupku unifikacije javiće se potreba za kombinacijom više zamena, odnosno da se na objekat nad kojim je primenjena jedna primeni i druga zamena. Recimo, ako je L literal, a θ i ρ zamene, biće u jednom momentu potrebno konstruisati literal $L\theta$, a nešto kasnije i literal $(L\theta)\rho$, što se obično piše kao $L\theta\rho$. Kako je $\theta\rho$ zamena, moguće je i dalje slaganje, recimo sa zamenom λ, kada se dobija $\theta\rho\lambda$.

Definicija 5.11.4 Neka su $\theta = \{t_1/x_1, t_2/x_2, \ldots, t_n/x_n\}$ i $\rho = \{u_1/y_1, u_2/y_2, \ldots, u_k/y_k\}$ dve zamene. Njihova *kompozicija* (u oznaci $\theta\rho$)[10] je zamena koja se dobija iz skupa $\{t_1\rho/x_1, \ldots, t_n\rho/x_n, u_1/y_1, \ldots, u_k/y_k\}$ brisanjem svih elemenata oblika:

- $t_j\rho/x_j$, gde je $t_j\rho = x_j$ i
- u_i/y_i, gde je y_i iz skupa $\{x_1, x_2, \ldots, x_n\}$.

Kompozicija zamena je asocijativna operacija, a kompozicija zamene sa praznom zamenom je sama ta zamena.

Primer 5.11.5 Neka su $\theta = \{f(y)/x, z/y\}$ i $\rho = \{a/x, b/y, y/z\}$ zamene. Prema definiciji 5.4.3 iz skupa $\{f(b)/x, y/y, a/x, b/y, y/z\}$ briše se y/y, pošto je $z\rho = y$, kao i a/x i b/y, pošto su x i y promenljive zamenjene u zameni θ. Time je dobijena zamena $\theta\rho = \{f(b)/x, y/z\}$. ■

Unifikacija je posebna vrsta zamene. Ona ima ulogu da ujednači neke literale, klauze, formule ili njihove skupove.

Definicija 5.11.6 Zamena θ je *unifikator* skupa literala $\{L_1, L_2, \ldots, L_n\}$, ako je $L_1\theta = L_2\theta = \ldots = L_n\theta$. Skup literala je *unifikabilan* ako ima unifikator. Zamena θ je *najopštiji unifikator* nekog skupa literala, ako je unifikator tog skupa i za svaki drugi unifikator ρ postoji zamena λ takva da je $\rho = \theta\lambda$.

Primer 5.11.7 Skup literala $\{P(a, y), P(x, f(b))\}$ je unifikabilan, pošto se ujednačava zamenom $\theta = \{a/x, f(b)/y\}$ koja je unifikator za taj skup. ■

[10] U kompoziciji se najpre primeni zamena θ, pa onda zamena ρ.

Da bi se unifikovala dva literala, recimo $P(a)$ i $P(x)$, potrebno je najpre ispitati da li su jednaki, u kom slučaju je unifikacija trivijalna, tj. prazan skup. Međutim, pošto oni nisu jednaki, potrebno je pronaći prvo mesto na kome se literali razlikuju, i pokušati razrešiti tu razliku. Podizrazi literala, koji počinju na tom mestima, čine *skup različitosti Raz*.

Primer 5.11.8 Neka je $W = \{P(x, f(y, z)), P(x, a), P(x, g(h(x))\}$. Prvi simboli na kojima se literali razlikuju su redom f, a i g u prvom, drugom, odnosno trećem literalu. Zato je skup različitosti skupa W, $Raz(W) = \{f(y, z), a, g(h(x))\}$. ■

Teorema 5.11.9 (Teorema unifikacije) Svaki skup literala koji ima unifikator ima i najopštiji unifikator.

Dokaz. Sledeća procedura opisuje postupak koji pronalazi najopštiji unifikator skupa literala W.

procedure Unifikacija(W)
begin
0 **if** (neki literali u skupu su negirani, a neki nisu) ili
(nemaju svi literali isti relacijski simbol)
then
Skup W nije unifikabilan.
return
1 $k := 0$
$W_0 :=$ skup literala W koji se testira
$\theta_0 :=$ prazna zamena, tj. prazan skup
2 **while** (W_k nije jednočlan skup) **do**
begin
3 Raz_k je skup različitosti skupa W_k
4 **if** Raz_k sadrži promenljivu x_k i term t_k tako da se x_k ne javlja u t_k
then
5 $\rho_k := \{t_k/x_k\}$
$\theta_{k+1} := \theta_k \rho_k$
$W_{k+1} := W_k \rho_k$ (tj. $W_{k+1} := W\theta_{k+1}$)
$k := k + 1$
else
6 Skup W nije unifikabilan.
return
end
7 Skup W je unifikabilan i θ_k je najopštiji unifikator skupa w
end

Ako je skup unifikabilan, pomoću opisane procedure će se u konačnom broju koraka odrediti najopštiji unifikator, a ako skup literala nije unifikabilan, dobiće se odgovor da unifikator ne postoji. Najpre, očigledno je da ako je neki literal negiran, a neki nije, onda skup nije unifikabilan. Slično je i ako literali počinju različitim relacijskim simbolom. Ako se u koraku (4) u skupu različitosti pojave dva različita

funkcijska simbola, opet skup nije unifikabilan. Ista je situacija, ako se promenljiva x_k javlja u termu t_k, jer će svaka zamena proizvesti literale različite dužine. Primetimo i da korak (4) formalizuje ono što je već rečeno, naime samo se promenljiva može zamenjivati, pa ako se u skupu različitosti ne nalazi ni jedna promenljiva, unifikacija propada. Pošto svaki korak (5) smanjuje broj pojava promenljivih, koji je u polaznom skupu konačan, petlja (2) mora pre ili posle završiti. Ako se algoritam završi u koraku (7), jasno je da je skup W unifikabilan i da je neki θ_k njegov unifikator, pošto je $W\theta_k$ jednočlan skup. Konačno, pokažimo da je dobijeni unifikator najopštiji. Indukcijom po broju koraka k pokazaćemo da za svaki drugi unigikator σ skupa W postoji zamena λ_k takva da je $\theta_k\lambda_k = \sigma$. Za $k = 0$, θ_0 je prazna zamena, pa je $\lambda_0 = \sigma$. Neka je $k > 0$ i indukcijska pretpostavka važi za θ_{k-1}. Ako je $W_k = W\theta_{k-1}$ jednočlan skup, tvrđenje je dokazano, jer će se iz petlje (2) izaći. U suprotnom, pošto je skup W unifikabilan, mora se nakon ulaska u tu petlju doći do koraka (5). Neka su x_k i t_k promenljiva i term kojima se formira zamena $\rho_k = \{t_k/x_k\}$ i $\theta_k = \theta_{k-1}\{t_k/x_k\}$. Zamena λ_{k-1} koja komponovana sa θ_{k-1} daje unifikator σ mora unifikovati x_k i t_k. Ovu zamenu modifikujmo eliminisanjem zamene oblika $\{t/x_k\}$, gde je t neki term, a dobijenu zamenu označimo sa λ_k. Sada je $\theta_k\lambda_k = \theta_{k-1}\{t_k/x_k\}\lambda_k$. Pošto se x_k ne zamenjuje u λ_k i ne javlja u t_k, $\theta_{k-1}\{t_k/x_k\}\lambda_k = \theta_{k-1}\lambda_k\{t_k\lambda_k/x_k\} = \theta_{k-1}\lambda_k\{t_k\lambda_{k-1}/x_k\} = \theta_{k-1}\lambda_{k-1} = \sigma$. ■

Primer 5.11.10 Neka je procedura Unifikacija primenjena na skup literala $W = \{P(a, x, f(g(y))), P(z, f(z), f(y_1))\}$. Pošto $W_0 = W$ nije jednočlan skup, pronalazi se $Raz_0 = \{a, z\}$ i formira zamena $\rho_0 = \{a/z\}$. Tada je $\theta_1 = \{a/z\}$ i $W_1 = W\theta_1 = \{P(a, x, f(g(y))), P(a, f(a), f(y_1))\}$. Dalje, W_1 nije jednočlan, pa se izračunava $Raz_1 = \{x, f(a)\}$, $\rho_1 = \{f(a)/x\}$, $\theta_2 = \{a/z, f(a)/x\}$ i $W_2 = W_1\rho_1 = \{P(a, f(a), f(g(y))), P(a, f(a), f(y_1))\}$. Kako ni W_2 nije jednočlan, izračunavaju se $Raz_2 = \{g(y), y_1\}$, $\rho_2 = \{g(y)/y_1\}$, $\theta_3 = \theta_2\rho_2 = \{a/z, f(a)/x, g(y)/y_1\}$ i $W_3 = W_2\rho_2 = \{P(a, f(a), f(g(y)))\}$. Dakle, polazni skup je unifikabilan i θ_3 je njegov najopštiji unifikator. Sličnim razmatranjem ustanovilo bi se da skup $\{Q(f(a), g(x)), Q(y, y)\}$ nije unifikabilan. Na početku bi bilo $Raz_0 = \{f(a), y\}$, što bi dovelo do zamene $\theta_1 = \{f(a)/y\}$, kojom bi se dobilo $W_1 = \{Q(f(a), g(x)), Q(f(a), f(a))\}$. Ali, sada $Raz_1 = \{f(a), g(x)\}$ ne sadrži promenljivu i polazni skup nije unifikabilan. ■

Navedeni algoritam unifikacije može raditi eksponencijalno dugo u odnosu na dužinu ulaza, ali postoje i efikasnije implementacije. Provera da li se promenljiva x_k javlja u termu t_k u koraku (4) algoritma unifikacije se naziva *provera pojave*[11]. U nekim implementacijama, recimo u programskom jeziku Prolog, zbog efikasnosti se ne vrši provera pojave, što u opštem slučaju nije korektno.

Primer 5.11.11 Razmotrimo primenu algoritma unifikacije u kome se ne vrši provera pojave za literale $P(x)$ i $P(f(x))$. Najpre će skup Raz sadržati x i f i doći će do zamene $\{f(x)/x\}$ koja će proizvesti literale $P(f(x))$ i $P(f(f(x)))$. Dalje se postupak ponavlja i ulazi u beskonačnu petlju. ■

5.11.2 Pravilo rezolucije za predikatsku logiku

U iskaznoj logici pravilo rezolucije se primenjivalo na dve klauze, u kojima su se pronalazili komplementarni literali, nakon čega su se klauze spajale, a iz unije su se

[11] Occurrence check.

uklanjali komplementarni literali. U predikatskom slučaju, postupkom unifikacije omogućeno je ujednačavanje komplementarnih literala, nakon čega se iskazni postupak skoro istovetno sprovodi. Međutim, u opštem slučaju izbor literala čija se unifikacija pokušava nije jednoznačan, pa dve klauze mogu imati više rezolventi.

Definicija 5.11.12 Ako dva ili više literala iz klauze C imaju najopštiji unifikator θ, klauza $C\theta$ je *faktor klauze* C. Ako je $C\theta$ jedinstven literal, reč je o *jediničnom faktoru*.

Primer 5.11.13 Zamenom $\theta = \{f(y)/x\}$ unifikuju se literali $P(x)$ i $P(f(y))$, pa je $\{P(f(y)), \neg Q(f(y))\}$ faktor klauze $\{P(x), P(f(y)), \neg Q(x)\}$. ■

Definicija 5.11.14 (Binarna rezolucija) Neka su C_1 i C_2 dve klauze koje nemaju zajedničke promenljive, L_1 literal iz C_1 i L_2 literal iz C_2. Ako L_1 i $\neg L_2$ imaju najopštiji zajednički unifikator θ, *binarna rezolventa* klauza C_1 i C_2 po literalima L_1 i L_2 je klauza

$$Res(C_1, C_2, L_1, L_2) = (C_1\theta - \{L_1\theta\}) \cup (C_2\theta - \{L_2\theta\})$$

dobijena tako što je najpre iz $C_1\theta$ izbrisan $L_1\theta$, zatim $L_2\theta$ iz $C_2\theta$, nakon čega su preostali literali povezani disjunkcijom.

Razmotrimo ovde zahtev u definiciji 5.11.14 da klauze C_1 i C_2 nemaju zajedničke promenljive. To nije problem pošto su skupovi klauza konjunkcije klauza, a promenljive u klauzama univerzalno kvantifikovane. Kako kvantifikator $\forall$ prelazi preko znaka $\wedge$ i važi da je preimenovanje vezanih promenljivih valjana formula, lako se dolazi do klauza koje nemaju zajedničke promenljive.

Primer 5.11.15 Neka su $C_1 = \{P(x), Q(x)\}$ i $C_2 = \{\neg P(a), R(x)\}$ klauze. Pošto se x javlja u obe klauzule, vrši se preimenovanje promenljive x u drugoj klauzi u y, tako da će biti $C_2 = \{\neg P(a), R(y)\}$. Kako $P(x)$ i $P(a)$ imaju najopštiji unifikator $\{a/x\}$, klauza $\{Q(a), R(y)\}$ je binarna rezolventa klauza C_1 i C_2. ■

Sledeća definicija pokriva sve slučajeve u kojima dve klauze imaju više rezolventi.

Definicija 5.11.16 *Rezolventa* klauza C_1 i C_2 je jedna od binarnih rezolventi:

- binarna rezolventa od C_1 i C_2,
- binarna rezolventa od C_1 i nekog faktora od C_2,
- binarna rezolventa od C_2 i nekog faktora od C_1 i
- binarna rezolventa nekog faktora od C_1 i nekog faktora od C_2.

Vratimo se na početnu ideju da se rezolucijom predikatskih formula u kojima se javljaju promenljive odjednom simulira više primena pravila rezolucije na klauze iz Erbranove ekspanzije. Sledeće tvrđenje koje se naziva lema podizanja[12] iskazuje da smo to i postigli. Na dalje ćemo podrazumevati da primerci klauza ne sadrže promenljive i da se dobijaju sistematskom zamenenom promenljivih elementima odgovarajućeg Erbranovog univerzuma.

[12] Lifting lemma.

Teorema 5.11.17 Neka su C_1 i C_2 dve klauze i neka su C_1' i C_2' dva njihova primerka koje se mogu rezolvirati u smislu pravila iskazne rezolucije. Tada postoji klauza R, rezolventa klauza C_1 i C_2 u smislu definicije 5.11.16, takva da je neki njen primerka jednak rezolventi $Res(C_1', C_2')$ klauza C_1' i C_2'.

Dokaz. Pretpostavimo bez gubitka opštosti da klauze C_1 i C_2 nemaju zajedničkih promenljivih i da je zamena σ takva da $C_1' = C_1\sigma$ i $C_2' = C_2\sigma$. Dalje, neka je L literal bez promenljivih takav da je $R' = Res(C_1', C_2', L, \neg L)$. Literal L se dobija od jednog ili više literala iz klauze C_1 na koje je primenjena zamena σ, tj. važi $L_1^1, \ldots, L_1^m \in C_1$ i $L_1^1\sigma = \ldots = L_1^m\sigma = L$. Slično je za $\neg L$ i klauzu C_2, tj. $L_2^1, \ldots, L_2^n \in C_2$ i $L_2^1\sigma = \ldots = L_2^n\sigma = \neg L$. Odatle se i klauze C_1 i C_2 mogu rezolvirati u smislu definicije 5.11.16. Neka je θ najopštiji unifikator skupa literala $W = \{L_1^1, \ldots, L_1^m, \neg L_2^1, \ldots, \neg L_2^n\}$ koji prema teoremi unifikacije 5.11.9 postoji. Tada je R binarna rezolventa

$$((C_1 \setminus \{L_1^1, \ldots, L_1^m\}) \cup (C_2 \setminus \{\neg L_2^1, \ldots, \neg L_2^n\}))_\theta$$

Pošto je θ najopštiji unifikator, a i σ je unifikator skupa literala W, postoji zamena λ takva da je $\theta\lambda = \sigma$. Zato je

$$\begin{aligned} R' &= (C_1' \setminus \{L\}) \cup (C_2' \setminus \{\neg L\}) = (C_1\sigma \setminus \{L\}) \cup (C_2\sigma \setminus \{\neg L\}) \\ &= ((C_1 \setminus \{L\}) \cup (C_2 \setminus \{\neg L\}))_\sigma = ((C_1 \setminus \{L\}) \cup (C_2 \setminus \{\neg L\}))_{\theta\lambda} = R_\lambda \end{aligned}$$

■

Naredna teorema je osnovna teorema o rezoluciji u predikatskoj logici i predstavlja pandan teoreme 3.7.6.

Teorema 5.11.18 Neka je A rečenica u Skolemovoj standardnoj formi takva da je njena matrica A^* u konjunktivnoj normalnoj formi. Neka je $Cla(A^*)$ skup klauza iz A^*. Formula A je nezadovoljiva ako i samo ako je $\emptyset \in Res^*(Cla(A^*))$.

Dokaz. ($\Leftarrow$) Kao i u iskaznom slučaju važi da, ako je R rezolventa dve klauze C_1 i C_2, onda je skup klauza $\{C_1, C_2\}$ zadovoljiv ako i samo ako je zadovoljiv skup klauza $\{C_1, C_2, R\}$. Odatle, ako je $\emptyset \in Res^*(Cla(A^*))$, skup $Res^*(Cla(A^*))$ je nezadovoljiv, pa je nezadovoljiv i skup A^*, a time i formula A.

($\Rightarrow$) Ako je formula A nezadovoljiva, onda postoji konačno izvođenje C_1', C_2', $\ldots$, $C_n' = \emptyset$ za praznu klauzu iskaznom rezolucijom u nekom konačnom podskupu Erbranove ekspanzije formule. Svaka klauza C_i' u tom izvođenju je bilo primerak neke klauze $C \in Cla(A^*)$ ili rezolventa nekih prethodnih klauza. Konstruisaćemo novi dokaz rezolucijom klauza iz $Cla(A^*)$ koje sadrže promenljive i koje takođe dovodi do prazne klauze. Ako je klauza C_i' primerak klauze $C \in Cla(A^*)$, koristićemo upravo tu klauzu, tj. $C_i = C$. Ako je C_i' rezolventa nekih prethodnih klauza posmatraćemo, po teoremi 5.11.17, klauzu C_i koja je rezolventa nekih prethodnih klauza iz novog dokaza čija je C_i' primerak. Konačno, pošto je prazna klauza dobijena u starom izvođenju jedino sopstveni primerak, ona se dobija i u novom dokazu. Dakle, $\emptyset \in Res^*(Cla(A^*))$. ■

Procedurom PredikatskaRezolucija se za proizvoljnu formulu ispituje da li je valjana, odnosno zadovoljiva, ako se formula na početku ne negira.

procedure PredikatskaRezolucija
begin
1 Prevesti negaciju formule u Skolemovu standardnu formu u kojoj je matrica u konjuktivnoj normalnoj formi
$D_1 \wedge D_2 \wedge \ldots \wedge D_n$.
$F := \{D_1, D_2, \ldots, D_n\}$
2 $Res_0(F) := F$; $Res_1(F) := R(Res_0(F))$; $i := 1$
3 **while** $((Res_i(F) \neq Res_{i-1}(F))$ **and** $\emptyset \notin Res_i(F))$ **do**
begin
4 $i := i + 1$
$Res_i(F) := R(Res_{i-1}(F))$
end
5 **if** $(\emptyset \in Res_i(F))$
then Formula je valjana
else Formula nije valjana
end

Suštinska novost u odnosu na ranije opisanu proceduru IskaznaRezolucija je da ovde petlja (3) ne mora biti konačna, što je samo posledica neodlučivosti predikatske logike.

Primer 5.11.19 Kada se formula $((\forall x)(P(x) \rightarrow Q(x)) \wedge P(c)) \rightarrow Q(c)$ iz primera 5.1.1 negira i prevede u Skolemovu standardnu formu, pri čemu je matrica forme u konjunktivnoj normalnoj formi, dobija se formula $(\forall x)((\neg P(x) \vee Q(x)) \wedge P(c) \wedge \neg Q(c))$. Njoj odgovarajući skup klauza je $F = \{\{\neg P(x), Q(x)\}, \{P(c)\}, \{\neg Q(c)\}$. Unifikujući x sa c, pronalaze se komplementarni literali $\neg P(x)$ i $P(c)$, pa je rezolventa ove dve klauze $\{Q(c)\}$. Ova klauza se rezolvira sa trećom klauzom iz F čime se dobija prazna klauza. Odatle je polazna formula valjana. Analognom transformacijom od formule $(\forall x)(\exists y)(Q(x) \rightarrow Q(y)) \rightarrow \neg(\exists z)Q(z)$ se dobija skup klauza $\{\{\neg Q(x), Q(f(x))\}, \{Q(a)\}\}$ u kome se petlja (3) vrti neograničeno. Recimo, unifikujuću x sa a, pronalaze se u klauzama komplementarni literali $\neg Q(x)$ i $Q(a)$, pa se rezolucijom dobija klauza $\{Q(f(a))\}$. Dalje se unifikuju x i $f(a)$, što dovodi do klauze $\{Q(f(f(a)))\}$ itd. ■

Primena rezolucije

Rezolucija se primenjuje u sistemima koji se ponašaju inteligentno, recimo odgovaraju na pitanja, rešavaju probleme dajući uputstvo kako nešto treba uraditi i sl. Uobičajen je postupak da se u takvim situacijama stanje stvari u oblasti na koju se pitanje odnosi opiše bazom znanja, tj. skupom klauza S. Zatim se i pitanje opiše klauzom C. Pozivajući se na ranija tvrđenja, klauza C proizilazi iz skupa klauza S, ako se iz skupa $S \cup \{\neg C\}$ izvodi prazna klauza. Samo izvođenje koje se shvata kao rešavanje problema se prepušta programu.

Razmotrimo situaciju u kojoj se postavlja pitanja oblika: da li je nešto, ili nije, postoji li neko sa tom osobinom, ili ko je osoba sa takvom osobinom i sl. U ovom slučaju pitanje se može zapisati formulom $(\exists x)P(x)$ i tumačiti po potrebi, i ako formula jeste posledica baze znanja, ono sa čime se unifikuje promenljiva x će biti određenje objekta za koji je odgovor potvrdan.

Primer 5.11.20 Neka se na osnovu rečenica "Ko može da čita, taj je pismen", "Delfini nisu pismeni" i "Delfin Joca je inteligentan" kojima odgovaraju predikatske formule:

A_1 $(\forall x)(\text{Čita}(x) \rightarrow \text{Pismen}(x))$,

A_2 $(\forall x)(\text{Delfin}(x) \rightarrow \neg\text{Pismen}(x))$ i

A_3 $\text{Delfin}(Joca) \wedge \text{Inteligentan}(Joca)$

postavi pitanje da li "Postoji neko ko je inteligentan i ne ume da čita", kome odgovara predikatska formula:

A_4 $(\exists x)(\text{Inteligentan}(x) \wedge \neg\text{Čita}(x))$.

Iz skupa $\{\{\neg\text{Čita}(x), \text{Pismen}(x)\}, \{\neg\text{Delfin}(x), \neg\text{Pismen}(x)\}, \{\text{Delfin}(Joca)\}, \{\text{Inteligentan}(Joca)\}, \{\neg\text{Inteligentan}(x), \text{Čita}(x)\}\}$ koji odgovara formuli $A_1 \wedge A_2 \wedge A_3 \wedge \neg A_4$ izvodi se prazna klauza, pa je odgovor: "Da, postoji neko ko je inteligentan i ne ume da čita". Zapravo, taj neko je Joca, pošto je promenljiva x iz klauze koja odgovara pitanju upravo unifikovana sa Joca. ■

5.11.3 Strategije rezolucije

U proceduri PredikatskaRezolucija, u glavnoj petlji (3), primenjuje se strategija rada po širini, naime sistematski su generisani skupovi $Res_i(F)$ rezolviranjem klauza iz prethodnih skupova. Često su mnoge od tih klauza bespotrebne pošto nikako ne utiču na dobijanje (ili nedobijanje) prazne klauze, odnosno na dokaz formule sa kojom se radi. Broj potencijalnih rezolvenata je često toliki da je ovaj problem nazvan "kombinatorna eksplozija". Programi, ili ljudi, koji koristeći rezoluciju rešavaju neki problem su pretrpani nekorisnim klauzama što bitno usporava njihov rad, pošto je potrebno porediti veliki broj klauza, probati sprovodljivost unifikacije, primenu pravila rezolucije itd. Zbog toga su za efikasan rad razvijeni postupci koji uvek, ili bar često, smanjuju broj klauza sa kojima se radi čime se ceo postupak ubrzava. Naročito su interesantni postupci koji očuvavaju kompletnost, tj. oni u kojima se rezultat dobija barem kada i standardnom procedurom. Sledi prikaz nekoliko postupaka od kojih se neki mogu i međusobno kombinovati. Svi ovi postupci poboljšavaju performanse sistema koji su na njima zasnovani, ali u opštem slučaju ne smanjuju esencijalno prisutnu složenost ispitivanja valjanosti.

Postupak čišćenja[13] je jedan od principa koji očuvavaju kompletnost, a zasniva se na odbacivanju iz skupa klauza svih klauza koje sadrže bar jedan literal koji se ne može unifikovati sa negacijom bilo kog literala bilo koje druge klauze iz skupa klauza.

Postupkom brisanja[14] se iz skupa klauza odstranjuju suvišne klauze. Ako su klauze C_1 i C_2 u nekom skupu F i postoji zamena θ takva da je $C_1\theta$ sadržano u C_2, klauza C_2 se sme izbaciti iz skupa F. Takođe, iz skupa F se smeju izbaciti sve klauze koje sadrže komplementarne literale. Može se pokazati da brisanje klauza očuvava kompletnost.

[13] Purity principle.
[14] Deletion strategy.

U postupku *hiperrezolucije* skup klauza F se deli na dva skupa F_1 i F_2. U skupu F_1 se nalaze sve klauze čiji ni jedan literal nije negiran, a u skupu F_2 su sve ostale klauze iz F. Sada se prilikom rezolucije zahteva da je jedna od klauza koje se rezolviraju iz skupa F_1. Dokazano je da hiperrezolucija očuvava kompletnost. Ova vrsta hiperrezolucije se naziva pozitivna. Potpuno simetričan je postupak u slučaju negativne hiperrezolucije, tj. kada se u skup F_1 izdvoje sve klauze koje imaju samo negirane literale. Hiperrezolucija očuvava kompletnost.

Postupak zasnovan na *potpornom skupu* je uopštenje hiperrezolucije. Neke klauze se izdvoje u skup F_1, a sve ostale u skup F_2 i zahteva se da je pri rezoluciji bar jedna klauza iz F_1, takozvanog potpornog skupa. Ako je pri toj podeli ispunjeno da je skup F_2 zadovoljiv, i ovaj postupak očuvava kompletnost. Često se podrazumeva da pretpostavke nekog problema nisu kontradiktorne, pa se smeštaju u skup F_2, dok skup F_1 sadrži klauze koje odgovaraju upitu.

U postupku *linearne rezolucije* skup F_1 na početku sadrži samo jednu klauzu, recimo klauzu koja odgovara pitanju, a sve ostale klauze su u skupu F_2. Tokom rezolucije nove klauze se smeštaju u F_1, a insistira se da je u svakoj rezoluciji bar jedna klauza iz tog skupa. Dokazano je da linearna rezolucija očuvava kompletnost.

U programskom jeziku Prolog se koristi SLD[15] postupak koji ne očuvava kompletnost.

5.12 Metoda analitičkih tabloa u predikatskoj logici

Za predikatsku logiku prvog reda potrebno je proširiti metodu analitičkih tabloa tako da se razmatraju i formule koje uključuju kvantifikatore. Najpre se ujednačavajuća notacija proširuje sa dve nove vrste formula: γ i δ-formulama i njihovim komponentama koje su prikazane u tabeli 5.2, gde je a simbol konstante. γ-formula je tačna pri nekoj interpretaciji ako i samo ako je $\gamma(a)$ tačno za svaku konstantu a. δ-formula je tačna pri nekoj interpretaciji ako i samo ako postoji element domena interpretacije kojim se interpretira konstanta a i važi da je formula $\delta(a)$ tačna.

γ	$\gamma(a)$
$T\ (\forall x)A(x)$	$T\ A(a/x)$
$F\ (\exists x)A(x)$	$F\ A(a/x)$

δ	$\delta(a)$
$F\ (\forall x)A(x)$	$F\ A(a/x)$
$T\ (\exists x)A(x)$	$T\ A(a/x)$

Tabela 5.2. γ i δ formule.

Pri nekim konstrukcijama tabloa za predikatske formule prvog reda biće potrebno obezbediti da se pravila primene na sve čvorove. Jedan takav slučaj je opisan u primeru 5.12.2. Primetimo da se u iskaznom slučaju ovakva situacija nije mogla pojaviti pošto su iskazni tabloi uvek konačni. Zbog ovoga se, pored opisa pravila konstrukcije za nove vrste formula, mora opisati i postupak koji će garantovati obradu svakog čvora. Čvorove na koje je primenjeno pravilo konstrukcije

[15] Linear resolution of definite clauses with an election function.

a je bilo koji simbol konstante a je novi simbol konstante

Slika 5.1. γ i δ pravilo.

nazvaćemo *završenim*. Nova pravila konstrukcije koja odgovaraju γ i δ formulama su:

> γ-pravilo: ako je neka γ formula na grani i ako je a bilo koji simbol konstante takav da se $\gamma(a)$ ne nalazi na toj grani, tada se grana produžava formulama $\gamma(a)$ i γ, a polazni čvor se proglašava završenim i
>
> δ-pravilo: ako je neka δ formula na grani i simbol konstante a se nije do tog trenutka pojavio u tablou, tada se grana produžava formulom $\delta(a)$, a polazni čvor se proglašava završenim.

Nova pravila za konstrukciju tabloa se šematski prikazuju kao na slici 5.1.

Sistematski tablo se konstruiše tako što se pravila konstrukcije uvek primenjuju na nezavršeni čvor koji je najbliži korenu tabloa i time obezbeđuje da će svi čvorovi tabloa biti obrađeni. Prihvatljiv je i bilo koji drugi postupak koji to isto garantuje. Sistematska konstrukcija tabloa i jedna kraća varijanta su ilustrovane u primeru 5.12.1.

Primer 5.12.1 Slika 5.2.a prikazuje tablo za formulu $(\exists y)((\exists x)P(x) \rightarrow P(y))$. Najpre se primenom γ-pravila kreiraju čvorovi (2) i (3) i uvodi konstanta a pošto nema ni jedne konstante koja bi se iskoristila u pravilu. Čvor (1) je završen, što više nećemo ponavljati za čvorove na koje je primenjeno neko pravilo. Zatim se α-pravilom primenjenim na čvor (2) uvode dva nova čvora (4) i (5). Ponovna primena γ-pravila na čvor (3) uvodi čvorove (6) i (7) novu konstantu b. Pravilom δ primenjenim na čvor (4) dobija se čvor (8) i uvodi nova konstanta c. Čvor (5) sadrži atomsku formulu, pa se dalje α-pravilo primenjuje na čvor (6) i dobijaju se čvorovi (9) i (10). Primena pravila γ na čvor (7) pri čemu se koristi konstanta c daje čvorove (11) i (12). Čvor (8) sadrži atomsku formulu. Pravilom δ primenjenim na čvor (9) dobija se čvor (13) i uvodi nova konstanta d. Čvor (10) sadrži atomsku formulu. Pravilom α iz čvora (11) se dobijaju čvorovi (14) i (15). Konačno, pošto čvorovi (8) i (15) sadrže formulu $T\ P(c)$, odnosno njen konjungat, jedina grana tabloa se zatvara.

Na slici 5.2.b prikazana je jedna kraća konstrukcija tabloa za istu formulu takva da se u primeni γ-pravila ne kopira γ formula na kraj grane, već se pamti na koje konstante treba da se primeni. Konstrukcija započinje kao i malopre γ-pravilom, kreiranjem čvora (2) i uvođenjem konstante a, ali se pri tome ne kopira polazna γ-formula iz čvora (1). Na čvor (2) se zatim primeni pravilo α i dobijaju čvorovi (3) i (4). Primena δ-pravila na čvor (3) uvodi novu konstantu b i čvor (5). Čvor

(1)	$F\ (\exists y)((\exists x)P(x) \to P(y))$	(1)	$F\ (\exists y)((\exists x)P(x) \to P(y))$
(2)	$F\ (\exists x)P(x) \to P(a)$	(2)	$F\ (\exists x)P(x) \to P(a)$
(3)	$F\ (\exists y)((\exists x)P(x) \to P(y))$	(3)	$T\ (\exists x)P(x)$
(4)	$T\ (\exists x)P(x)$	(4)	$F\ P(a)$
(5)	$F\ P(a)$	(5)	$T\ P(b)$
(6)	$F\ (\exists x)P(x) \to P(b)$	(6)	$F\ (\exists x)P(x) \to P(b)$
(7)	$F\ (\exists y)((\exists x)P(x) \to P(y))$	(7)	$T\ (\exists x)P(x)$
(8)	$T\ P(c)$	(8)	$F\ P(b)$
(9)	$T\ (\exists x)P(x)$	(8)	$\times$
(10)	$F\ P(b)$		
(11)	$F\ (\exists x)P(x) \to P(c)$		
(12)	$F\ (\exists y)((\exists x)P(x) \to P(y))$		
(13)	$T\ P(d)$		
(14)	$T\ (\exists x)P(x)$		
(15)	$F\ P(c)$		
(16)	$\times$		

a) b)

Slika 5.2. Tablo dokaz za formulu $(\exists y)((\exists x)P(x) \to P(y))$.

(4) sadrži atomsku formulu. Primena γ-pravila na čvor (1) i konstantu b daje čvor (6). Čvor (5) sadrži atomsku formulu. Primenom pravila α na čvor (6) dobijamo čvorove (7) i (8) nakon čega je jedina grana tabloa zatvorena zbog čvorova (5) i (8). Ovako konstruisani tablo je očigledno kraći i sadrži manje konstanti od prvog tabloa. ■

Međutim ni skraćenje opisano u primeru 5.12.1.b, a ni bilo koje drugo, ne pomaže u slučaju svih formula, tj. za neke predikatske formule će u tablou uvek postojati beskonačne grane, što je posledica neodlučivosti problema valjanosti u predikatskoj logici prvog reda. U primeru 5.12.2 je prikazan jedan takav slučaj.

Primer 5.12.2 Deo tabloa za formulu $(\exists x)(\forall y)P(x, y)$ je prikazan na slici 5.3. Najpre se pravilo γ primeni na čvor (1) pri čemu se dobija čvor (2) i uvodi konstanta a. Time započinje ciklus u kome se δ-pravilom uvodi nova konstanta, za koju se ponovo primeni γ-pravilo na čvor (1). Tako se dobijaju čvorovi (3) i (4) i konstanta b, pa (5) i (6) i konstanta c itd. Lako je videti da se grana tabloa nakon bilo kog konačnog broja koraka neće zatvoriti, ali ni završiti. ■

Pod *završenim sistematskim tabloom* se podrazumeva sistematski tablo koji je ili beskonačan ili je konačan a primenom pravila za konstrukciju se ne može pojaviti ni jedna nova formula. Sada se mogu, kao i u iskaznom slučaju, formulisati teoreme korektnosti i kompletnosti. Dokaz teoreme o korektnosti 5.12.3 je analogan dokazu odgovarajuće iskazne teoreme 3.8.5, pa ga ovde ne navodimo. U dokazu teoreme kompletnosti 5.12.4 samo ćemo istaći novine u odnosu na dokaz teoreme 3.8.7.

Teorema 5.12.3 (Korektnost) Ako formula A ima tablo dokaz, ona je valjana.

$$
\begin{array}{ll}
(1) & F\ (\exists x)(\forall y)P(x,y) \\
(2) & \quad F\ (\forall y)P(a,y) \\
(3) & \quad\quad F\ P(a,b) \\
(4) & \quad F\ (\forall y)P(b,y) \\
(5) & \quad\quad F\ P(b,c) \\
(6) & \quad F\ (\forall y)P(c,y) \\
 & \quad\quad \ldots
\end{array}
$$

Slika 5.3. Tablo za formulu $(\exists y)(\forall y)P(x,y)$.

Teorema 5.12.4 (Kompletnost) Ako je formula A valjana, završeni sistematski tablo za A mora biti zatvoren posle konačno mnogo koraka.

Dokaz. Najpre, slično dokazu teoreme 3.8.6, uočimo da svaka završena grana θ sistematskog tabloa koja nije zatvorena može poslužiti za definisanje interpretacije koja će zadovoljavati sve formule sa te grane. Novi su samo slučajevi γ i δ formula. Ako je δ formula na grani, na grani je bar jedna $\delta(a)$ formula. Ako je γ formula na grani, na grani su sve formule $\gamma(a)$ za sve simbole konstanti iz nekog skupa konstanti. Interpretacija I_θ se definiše na sledeći način. Domen predstavljaju sve konstante, a zadovoljene su sve atomske formule bez promenljivih koje imaju znak T i nalaze se na grani. Dalje se indukcijom pokazuje da su zadovoljene sve formule sa grane, pa i formula u korenu tabloa. Primetimo da sistematičnost garantuje da samo konačne grane tabloa mogu biti zatvorene, jer se zatvaranje grane vrši u nekom fiksiranom koraku konstrukcije u kome je celi tablo konačan, nakon čega se zatvorena grana više ne menja. Prema tome, ako je formula valjana, a tablo za nju ima bilo konačnu, bilo beskonačnu otvorenu granu, rezonujući kao i u dokazu teoreme 3.8.7 izveli bismo kontradikciju da je i formula valjana i njena negacija zadovoljiva, odakle sledi tvrđenje. ■

U slučaju da je tablo završen, a neka grana nije zatvorena, na osnovu teoreme kompletnosti 5.12.4, moguće je konstruisati kontramodel definišući interpretaciju koja zadovoljava sve atomske formule bez promenljivih sa prefiksom T.

Primer 5.12.5 Razmotrimo formulu $(\exists x)(\forall y)P(x,y)$ čiji tablo je opisan u primeru 5.12.2. Čitajući atomske formule sa grana dolazimo do interpretacije u kojoj je domen prebrojivo beskonačan skup $\{a,b,c,\ldots\}$ i za koju važi $I \not\models P(a,b)$, $I \not\models P(b,c)$, $\ldots$ za koji očigledno nije $I \models (\exists x)(\forall y)P(x,y)$. Primetimo da se analizom ponekad kontramodel može skratiti, na primer jedan kontramodel posmatrane formule ima dvočlani domen $\{a,b\}$ u kome je $I \not\models P(a,b)$, $I \not\models P(b,a)$. ■

5.13 Logičko programiranje

Istraživanja vezana za obradu prirodnih jezika i automatsko dokazivanje teorema dovela su do stvaranja oblasti koja je nazvana logičko programiranje. Neposredno po nastanku rezolucije njena efikasna primenljivost je bila bitno ograničena pošto nije bila poznata prirodna strategija kojom bi se izvođenje učinilo efikasnim. Oko

1970. godine predložena je[16] jedna takva strategija koja je, međutim, primenljiva samo na posebnu vrstu klauza, takozvane *Hornove klauze*. Implementacijom ove ideje pojavio se programski jezik Prolog[17], 1972. godine.

5.13.1 Elementi logičkog programiranja

Logika je pre pojave programskog jezika Prolog upotrebljavana za opisivanje problema. Međutim, kada je pokazano da skup formula može imati i proceduralnu interpretaciju, logika je iskorištena i kao programski jezik.

Osnovna ideja logičkog programiranja izražena je takozvanom jednačinom Kovalskog:

$$\text{algoritam} = \text{logika} + \text{kontrola}$$

pri čemu je pod logikom podrazumevan opis problema dat formulama, dok je kontrola formalni mehanizam izvođenja, tj. zaključivanja, koji ostvaruje sam računar (odnosno, odgovarajući program). Zadatak programera je da opiše šta je problem, a na računaru je da odgovori kako da se taj problem reši. Odatle, logički jezici po pravili imaju dva posebna dela:

- deo za prihvatanje opisa problema, tj. korisnički interfejs i
- deo za automatsko izvršavanje programa, tj. mehanizam izvođenja.

Očito je da je ideja logičkog programiranja bliska veštačkoj inteligenciji. Logičko programiranje omogućava rešavanje problema i pisanje programa u raznim oblastima, ali i sam razvoj jezika logičkog programiranja je značajno polje istraživanja u računarstvu.

Ideja formulisana jednačinom Kovalskog nije u potpunosti postignuta, pošto jezici koji pretenduju da su jezici logičkog programiranja poseduju i neke nelogičke elementa, na primer kontrolne komponente. Pa ipak i ti postojeći jezici donose niz pogodnosti:

- sintaksa i semantika jezika su relativno jednostavne (logički jezici nemaju naredbu **goto**, naredbe pridruživanja, naredbe ciklusa i **if-then-else** naredbe),
- rezonovanje o logičkim programima je lakše nego o proceduralnim,
- veliki broj problema se prirodno rešava metodologijom logičkog programiranja (ekspertni sistemi, baze podatake, obrada prirodnih jezika itd.),
- postoje značajne olakšice u paralelizaciji logičkih u odnosu na proceduralne programe,
- razvoj logičkih programa je obično znatno brži nego razvoj proceduralnih programa itd.

[16] Jedan od najzaslužnijih za to je Robert Kowalski (Kovalski).

[17] Prolog je skraćenica nastala iz reči programiranje u logici, *programming in logic*.

Glavni problem logičkog programiranja je efikasnost izračunavanja. Kreatori jezika logičkog programiranja pokušavaju da taj problem reše na razne načine, pre svega korištenjem nekih elemenata kontrole i paralelizacijom izvršavanja.

Sintaksa jezika u logičkom programiranju je veoma jednostavna. Termini koji se koriste su većinom vezani za vrste klauza i pretpostavlja se i da su sve promenljive univerzalno kvantifikovane, tako da se simbol $\forall$ nigde ne piše.

Definicija 5.13.1 *Klauza Kovalskog* je formula oblika

$$A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n$$

gde su sve formule A_i i B_j atomske. Prazna klauza (u oznaci $\leftarrow$) je sinonim za kontradikciju.

Definicija 5.13.2 *Definitna klauza* je formula oblika

$$A \leftarrow B_1 \wedge \ldots \wedge B_n$$

gde su A i sve formule B_j atomske. Za $n = 0$ definitna klauza ima oblik $A \leftarrow$ i naziva se jedinična klauza. U definitnoj klauzi atomska formula A je *glava klauze*, a formula $B_1 \wedge \ldots \wedge B_n$ *telo klauze*.

Definicija 5.13.3 *Ciljna klauza* je formula oblika

$$\leftarrow B_1 \wedge \ldots \wedge B_n$$

gde su svi B_j atomske formule.

Definicija 5.13.4 *Hornova kluza* je ili definitna ili ciljna klauza.

Definicija 5.13.5 *Logički program* je skup Hornovih klauza. *Procedura* je skup definitnih klauza čije glave sadrže isti predikatski simbol.

Klauza Kovalskog se može napisati i u obliku $A_1 \vee \ldots \vee A_m \vee \neg B_1 \vee \ldots \vee \neg B_n$, odnosno da su svi B_j negativni, a svi A_i pozitivni literali. Slično, definitna klauza se može pisati u obliku $A \vee \neg B_1 \vee \ldots \vee \neg B_n$, jedinična klauza u obliku A a ciljna klauza u obliku $\neg B_1 \vee \ldots \vee \neg B_n$, odnosno kao $\neg(B_1 \wedge \ldots \wedge B_n)$.

Po pravilu, definitne klauze predstavljaju bazu znanja, tj. opis problema, a ciljna klauza upit, odnosno zadatak koji treba rešiti. Ako su $C_1, C_2, \ldots, C_k$ sve definitne klauze jednog logičkog programa i ciljna klauza logičkog programa

$$G = \leftarrow B_1 \wedge \ldots \wedge B_n = \neg(B_1 \wedge \ldots \wedge B_n),$$

onda je izvršenje logičkog programa pokušaj izvođenja prazne klauze iz skupa klauza $\{C_1, C_2, \ldots, C_k, G\}$. Ukoliko se u tome uspe, formula $\neg G = B_1 \wedge \ldots \wedge B_n$ je posledica skupa klauza $\{C_1, C_2, \ldots, C_k\}$. Samo izvođenje se sastoji od primena pravila rezolucije. Očigledno, izvršenje logičkog programa je postupak opovrgavanja. Ono što je posebno značajno je da se prilikom izvođenja prazne klauze izračunavaju i vrednosti promenljivih koje se pojavljuju u upitu. Time se dobijaju konkretne vrednosti u odgovoru na pitanje za koje vrednosti promenljivih je formula $B_1 \wedge \ldots \wedge B_n$ posledica baze znanja logičkog programa.

Jezici logičkog programiranja se međusobno razlikuju u izboru klauza koje se rezolviraju. Međutim, po pravilu, jedna od klauza je ili ciljna klauza, ili klauza dobijena prethodnim rezolviranjem, što znači da se jedna klauza u rezolviranju sastoji isključivo od negativnih literala. Prema tome, strategija koja se u logičkim jezicima primenjuje je jedna vrsta linearne negativne hiperrezolucije. Kako definitne klauze imaju tačno jedan pozitivan literal, u logičkom programiranju je pojednostavljeno ispitivanje mogućnosti primene pravila rezolucije na dve klauze.

Primetimo da se definitna klauza $A \leftarrow B_1 \wedge \ldots \wedge B_n$ može čitati na dva načina:

- deklarativno: A je tačno, ako su B_1 i B_2 i $\ldots$ i B_n tačni i
- proceduralno: da bi se dokazao cilj A (izvršila procedura A, rešio problem A) moraju se dokazati potciljevi (izvršiti procedure, rešiti problemi) B_1 i B_2 i $\ldots$ i B_n.

Deklarativno i proceduralno čitanje klauza su u osnovi logičkog, odnosno kontrolnog aspekta algoritma pomenutog u jednačini Kovalskog. Zajedničko za logičko programiranje i automatsko dokazivanje teorema je da se:

- logički jezik koristi za opis znanja i
- formalno izvođenje koristi pri nalaženju rešenja problema.

Međutim, logičko programiranje se ne svodi na automatske dokazivače, pošto:

- logičko programiranje koristi formalni logički jezik i za definisanje izračunljivih funkcija i
- logičko programiranje koristi proceduru izvođenja koja je vođena ciljem za izvršavanje takvih definicija kao programa.

5.13.2 Prolog

Prolog je najviše korišten programski jezik inspirisan idejom logičkog programiranja. Osnovnu celinu jezika čine relacije koje se definišu procedurama, tj. skupovima Hornovih klauza koje u glavi imaju isti predikat. Unifikacija predstavlja glavni mehanizam za manipulaciju podacima, kojom se vrši prenos parametara i selektovanje i konstrukciju objekata. Prolog nije čist logički jezik jer sadrži i kontrolne elemente, ali ga to čini pogodnim da se prihvati kao programski jezik opšte namene, pogodan za razvoj ekspertnih sistema, relacionih baza podataka itd.

5.14 Za dalje proučavanje

Literatura spomenuta u poglavlju o iskaznoj logici se većinom odnosi i na predikatsku logiku prvog reda, tako da je nećemo ponovo navoditi. Opis strukture steka u okviru predikatske logike prvog reda potiče iz [76]. Konstrukcija definicione forme i poređenja različitih formalnih sistema za predikatsku logiku prvog reda su detaljno opisani u [26]. Problemi odlučivosti i neodlučivosti velikog broja teorija razmatraju se u [2]. Kompletan dokaz teoreme 5.9.4 može se pronaći u [12]. U [60] matematičkoj logici se pristupa sa stajališta veštačke inteligencije. Takođe, analizirani su elementi logičkog programiranja i programskog jezika Prolog.

6

Opisne logike

Opisne logike, poznate i kao terminološke logike, predstavljaju grupu srodnih logika koje služe za opisivanje baza znanja. U njima se prvo definišu koncepti domena, a zatim se oni koriste za specifikaciju osobina objekata i individua koji se pojavljuju u domenu. Opisne logike su izvedene od mreža sa strukturnim nasleđivanjem da bi se prevazišli nedostaci semantičkih mreža i sistema okvira pri opisivanju baza znanja kao i izvodjenju zakjučaka nad tom bazom. U ovom odeljku opisaćemo ukratko osnovne opisne logike i ukazati na logičke osnove zaključivanja koje se primenjuje u semantičkom web-u, optimizaciji upita kod objektno orijentisanih baza podataka, zatim opisu i klasifikaciji multimedijalnih podataka kao i izvođenju zaključaka u robotici.

6.1 Definicije osnovnih pojmova opisnih logika

Jezik opisnih logika je prebrojiv skup koji se sastoji od simbola atomskih koncepta[1], zatim simbola atomskih uloga[2], simbola konstanti, logičkih veznika, kao i pomoćnih simbola leve i desne zagrade i zareza. Konceptne formule, ili kratko koncepti, se grade od atomskih koncepata, dok se formule uloga, ili kraće uloge, grade uz pomoć atomskih uloga. Sa N_c, N_R, N_I, F ($F \subseteq N_R$) redom označavamo skup svih koncepata, uloga, konstanti i atributa, dok sa A_c, A_p, A_t redom označavamo proizvoljan atomski koncept, atomsku ulogu i atomski atribut. Sa C i D, odnosno R i S, odnosno A i B redom označavamo koncepte, uloge i atribute.

Simbol $\bot$ označava kontradikciju, na primer $A \sqcap \neg A$, dok sa $\top$ kraće označavamo tautologije tipa $\neg(A \sqcap \neg A)$.

Definicija 6.1.1 Koncepti, uloge i atributi se definišu na sledeći način:

- Svaki atomski koncept A_c i njegova negacija $\neg A_c$, kao i $\top$ i $\bot$ su koncepti jezika opisnih logika.
- Svaka atomska uloga A_p (uključujući i atomski atribut A_t) su uloge

[1] Atomic concept.
[2] Atomic role.

- Svaki atomski atribut A_t je atribut
- Ako su C i D proizvoljni koncepti, R i S proizvoljne uloge i A i B atributi onda su:
  - $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists R.C$, $\forall R.C$, $\geq nR.C$, $\leq nR.C$, $A = B$, $A \neq B$ koncepti, gde je n pozitivan ceo broj,
  - $\neg R$, $R \sqcap S$, $R \sqcup S$, $R \circ S$, $id(C)$, R^{-1}, R^{+}, R^{*}, uloge i
  - $A \sqcap B$, $A \circ B$ su atributi,
- koncepti, uloge i atributi se dobijaju konačnom primenom prethodnih pravila.

Koncepti se interpretiraju kao unarni predikati, definisani na skupu zvanom domen, a uloge i atributi kao binarni predikati na domenu, s tim što atributi moraju da zadovoljavaju dodatne semantičke uslove navedene u sledećoj definiciji kojom se uvodi značenje koncepata, uloga i atributa.

Definicija 6.1.2 Interpretacija I je uređeni par $(\triangle^I, \cdot^I)$ gde je $\triangle^I$ neprazan skup nazvan domen interpretacije dok je $\cdot^I$ funkcija koja:

- Svaki atomski koncept A_c preslikava u unarnu relaciju A_c^I skupa $\triangle^I$ ($A_c^I \subseteq \triangle^I$),
- Svaki atomski atribut A_t preslikava u binarnu relaciju A_t^I nad $\triangle^I$, ($A_t^I \subseteq \triangle^I \times \triangle^I$) uz ograničenje: za svaki $a \in \triangle^I$ postoji najviše jedan $b \in \triangle^I$ tako da je $(a, b) \in A_t^I$,
- Svaku atomsku ulogu A_p preslikava u binarnu relaciju A_p^I nad $\triangle^I$, ($A_p^I \subseteq \triangle^I \times \triangle^I$)

i za koju važi:

- $(\top)^I = \triangle^I$
- $(\bot)^I = \emptyset$
- $(\neg C)^I = (\triangle)^I / (C)^I$,
- $(C \sqcap D)^I = C^I \cap D^I$
- $(C \sqcup D)^I = C^I \cup D^I$
- $(\forall R.C)^I = \{a \in \triangle^I : (\forall b)((a, b) \in R^I \Longrightarrow b \in C^I)\}$
- $(\exists R.C)^I = \{a \in \triangle^I : (\exists b)((a, b) \in R^I \wedge b \in C^I)\}$
- $(\exists_{\geq n} R.C) = \{a \in \triangle^I : |\{b \in \triangle^I : (a, b) \in R^I, b \in C^I\}| \geq n\}$
- $(\exists_{\leq n} R.C) = \{a \in \triangle^I : |\{b \in \triangle^I : (a, b) \in R^I, b \in C^I\}| \leq n\}$
- $(A = B)^I = \{d \in \triangle^I : \{a \in \triangle^I : A(d, a)\} = \{b \in \triangle^I : B(d, b)\}\}$
- $(A \neq B)^I = \{d \in \triangle^I : \{a \in \triangle^I : A(d, a)\} \neq \{b \in \triangle^I : B(d, b)\}\}$

- $(R \sqcap S)^I = R^I \cap S^I$
- $(R \sqcup S)^I = R^I \cup S^I$
- $(\neg R)^I = (\Delta^I \times \Delta^I) \setminus R^I$
- $(R \circ S)^I = \{(a,c) \in \Delta^I \times \Delta^I : (\exists b)((a,b) \in R^I \wedge (b,c) \in C^I)\}$
- $id(C)^I = \{(a,a) \in \Delta^I \times \Delta^I : a \in C^I\}$
- $(R^+)^I = \bigcup_{i \geq 1}(R^I)^i$, tj. $(R^+)^I$ je tranzitivno zatvaranje od (R^I).
- $(R^*)^I = \bigcup_{i \geq 0}(R^I)^i$ je tranzitivno refleksivno zatvaranje, gde je $(R^I)^0 = \{(a,a) : a \in \triangle^I\}$
- $(R^{-1})^I = \{(b,a) \in \Delta^I \times \Delta^I : (a,b) \in R^I\}$

Na osnovu 6.1.2 atomski atributi se od atomskih uloga razlikuju po tome što za svaki $a \in \triangle^I$ postoji najviše jedan $b \in \triangle^I$ tako da je $(a,b) \in A_t^I$. Lako se pokazuje da isto važi za sve atribute i uloge. Očigledno je da važi $R^+ \stackrel{def}{=} R \circ R^*$. Ponekad se umesto $\exists_{\geq n} R.C$, odnosno $\exists_{\leq n} R.C$, redom piše $\geq nR.C$, $\leq nR.C$, odnosno $\geq nR$,$\leq nR$, za $C \equiv \top$.

Kao skraćenice uvode se oznake za sledeće uloge: $self \stackrel{def}{=} id(\top)$ i $R|C \stackrel{def}{=} R \circ id(C)$.

Primer 6.1.3 Pretpostavimo da imenice Osoba i Žensko predstavljaju atomske koncepte, tada $(Osoba \sqcap \check{Z}ensko)$ i $(Osoba \sqcap \neg \check{Z}ensko)$ su koncepti, pa za prvi koncept, intuitivno, možemo reći da predstavlja osobe koje su ženskog pola, dok drugi koncept predstavlja osobe koje nisu ženskog pola. Neka DeteOd predstavlja simbol jedne uloge. Na osnovu nje možemo graditi sledeće složenije koncepte, pa na primer ($Osoba \sqcap \exists$ DeteOd.$\top$), što predstavlja one osobe koje imaju dete. Koncept (Osoba $\sqcap \forall$ DeteOd.Žensko) opisuje one osobe čija su sva deca ženskog pola. Neka je $\triangle^I = \{Jelena, Marko\}$, onda konstanta $Jelena \in (Osoba \sqcap \check{Z}ensko)^I$, a konstanta $Marko \in (Osoba \sqcap \neg \check{Z}ensko)^I$. ■

Primer 6.1.4 Neka su SinOd i CerkaOd atomske uloge. Njihova unija u oznaci $(SinOd \sqcup CerkaOd)$ predstavlja ulogu DeteOd. Tranzitivno zatvaranje uloge u oznaci $(DeteOd^+) = \bigcup_i (DeteOd)^i$ predstavlja ulogu PotomakOd. Neka je $\triangle^I = \{Sanja, Marko\}$ i $(DeteOD)^I = \{(Sanja, Marko)\}$ tj. Sanja je dete od Marka. Tada inverzija uloge DeteOd u oznaci $(DeteOd^{-1}) = \{(Marko, Sanja)\}$ predstavlja ulogu RoditeljOd. ■

Definicija 6.1.5 Kažemo da su dva koncepta C i D ekvivalentna i pišemo $C \equiv D$ ako je $C^I = D^I$ za sve interpretacije I. Kažemo da je koncept zadovoljiv ako postoji interpretacija I tako da je $C^I \neq \varnothing$. Koncept C obuhvata koncept D ($D \sqsubseteq C$) ako i samo ako za svaku interpretaciju I važi $D^I \subseteq C^I$. Odgovarajući pojmovi se analogno definišu za uloge i atribute.

Osnovna opisna logika označava se sa ALC[3]. Ostale logike iz ove klase opisnih logika, shodno prethodnim definicijama sintakse i semantike, dobijamo proširenjem ALC opisne logike. U tabelama 6.1 i 6.2 dajemo pregled nekih opisnih logika koje su dobijene na taj način.

Teorema 6.1.6 U opisnim logikama koje sadrže ALC važi:

- $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$
- $C \sqcap D \equiv \neg(\neg C \sqcup \neg D)$
- $\exists R.C \equiv \neg\forall R.\neg C$
- $\forall R.C \equiv \neg\exists R.\neg C$

	Koncepti										
Logika	$\sqcap$	$\sqcup$	$\neg$	$\exists R.C$	$\forall R.C$	$(\geq_n,\leq_n)R$	$(\geq_n,\leq_n)R.C$	$\exists A$	$\forall A$	$=$	$\neq$
ALC	x	×	×	×	×						
$ALCN$	×	×	×	×	×	×					
$ALCR$	×	×	×	×	×						
$ALCNR$	×	×	×	×	×	×					
$ALCF$	×	×	×	×	×			×	×	×	×
$ALCFN$	×	×	×	×	×	×		×	×	×	×
$ALCFNR$	×	×	×	×	×	×		×	×	×	×
$ALCN(o)$	×	×	×	×	×	×					
ALC_+	×	×	×	×	×						
ALC_{R^+}	×	×	×	×	×						
$ALC_\oplus$	×	×	×	×	×						
TSL	×	×	×	×	×						
CIQ	×	×	×	×	×	×	×				
$ALCH_{R^+}$	×	×	×	×	×						
$ALCHF_{R^+}$	×	×	×	×	×			×	×		

Tabela 6.1. Koncepti za neka proširenja ALC opisne logike.

Nedostatak klasa logika uvednih prethodnim definicijama je što nije moguće opisati neke osobine karatkeristične za određene objekte nekog domena. Tako na primer ako želimo da opišemo sve muškarce koji su stariji od svojih sestara, to možemo da uradimo samo koristeći unarni predikat starijiOdSestre na sledeći način: $Muskarac \sqcap \exists sestraOd.Zensko \sqcap starijiOdSestre$. Ali ako nas zanima da li je brat stariji od sestre dve ili pet godina, moramo uvesti unarne predikate: StarijiOdSestre2god i StarijiOdSestre5god. Kod mnogih opisnih logika u ovakvim slučajevima se javljaju poteškoće pri zaključivanju, recimo kod ispitivanja zadovoljivosti koncepata. Suština problema je kako integrisati brojeve i ostale tipove podataka poput stringova u opisne logike. Da bi se ovo prevazišlo uvode se konkretni domeni, odnosno nova klasa opisnih logika sa konkretnim domenima. Te klase opisnih logika su proširenje postojećih klasa.

[3]Atribute Language with Complemente

	Uloge							
Opisne logike	$\sqcap$	$\sqcup$	$\circ$	id	$^{-1}$	$+$	$*$	$R \circ S$
$ALCR$	$\times$							
$ALCNR$	$\times$							
$ALCF$								$\times$
$ALCFN$								$\times$
$ALCFNR$	$\times$							$\times$
$ALCN_{(}o)$			$\times$					
ALC_{+}		$\times$	$\times$			$\times$		
$ALC_{R^{+}}$								
$ALC_{\oplus}$								
TSL		$\times$	$\times$	$\times$	$\times$		$\times$	
CIQ		$\times$	$\times$	$\times$	$\times$		$\times$	
$ALCH_{R^{+}}$								
$ALCHf_{R^{+}}$								

Tabela 6.2. Uloge za neka proširenja ALC opisne logike.

Definicija 6.1.7 Konkretni domen D je uređeni par (Δ_D, Φ_D), gde je Δ_D proizvoljan skup, dok je Φ_D proizvoljan skup predikata i funkcija definisanih nad skupom Δ_D. Predikati i funkcije imaju određenu arnost.

Primeri konkretnih domena su:

- Δ_D je skup prirodnih brojeva N, a Φ_D je $\{\geq, =, +\}$
- Δ_D je skup reči preko nekog alfabeta σ, a Φ_D je {praznaRec, prefiksOd, konkatenacija}

Integracija konkretnih domena u opisne logike se ostvaruje dodavanjem:

- abstraktnih osobina[4], koje su funkcionalne uloge[5] a ponekad se nazivaju i atributi, (na primer: majkaOd),
- konkretnih osobina[6], koje shvatamo kao pridruživanje vrednosti iz konkretnog domena logičkim objektima (veličina, tempretatura),
- novih konstruktora koji opisuju ograničenja na konkretnom domenu. Konstruktori su oblika $\exists_{u_1,\ldots u_n}.P$ gde je svaki u_i oblika $f_1 \ldots f_{k_i} g$ od k_i abstraktnih osobina $f_1, \ldots, f_{k_i}$ iza kojih sledi jedna konkretna osobina g, a P je n-arni predikat iz konkretnog domena.

Proširenjem definicije 6.1.1, nove konstrukcije bogatijeg jezika se uvode na sledeći način:

[4]abstract features
[5]functional roles
[6]concrete features

Definicija 6.1.8 Sa N_{aF}, N_{cF} redom označavamo skup svih abstraktnih osobina i konkretnih osobina. Put u je kompozicija $f_1 \dots f_k g$ od k abstraktnih osobina $f_1 \dots f_k (n \geq 0)$ i jedne konkretne osobine g. Ako je $\triangle^D$ konkretan domen tada su:

- $\exists_{u_1,\dots u_k}.P$,
- $g \uparrow$

koncepti.

Značenje novih formula se dobija proširenjem definicije 6.1.2 na sledeći način:

Definicija 6.1.9 Interpretacija I je uređeni par $(\triangle^I, \cdot^I)$ gde je $\triangle^I$ neprazan skup nazvan domen interpretacije dok je $\cdot^I$ funkcija koja:

- svaku abstraktnu osobinu f preslikava u parcijalnu funkciju $f^I : \triangle^I \longrightarrow \triangle^I$
- svaku konkretnu osobinu g preslikava u parcijalnu funkciju $g^I : \triangle^I \longrightarrow \triangle^D$
- ako je $u = f_1 \dots f_k g$ i $d \in \triangle^I$ tada je $u^I(d)$ definisano sa $g^I(f_k^I \dots (f_1^I(d)) \dots)$.

i za koju važi:

- $(\exists_{u_1,\dots u_k}.P)^I = \{d \in \triangle^I : \exists x_1 \dots x_n \in \triangle^D : u_i^I(d) = x_i \text{ i } (x_1, \dots, x_n) \in P^D\}$,
- $(g \uparrow)^I = \{d \in \triangle^I : g^I(d) \text{ je nedefinisano}\}$

Uvođenjem konkretnih domena 6.1.7 proširuju se logike u tabeli 6.1. Nove logike se označavaju dodavanjem (D) na kraju imena, recimo ALC(D) ili ALCF(D) tipa.

Primer 6.1.10 Neka su $Muskarac$ atomski koncept, konkretna osobina *godine*, $\geq_n$ unarni predikat i neka je skup prirodnih brojeva konkretni domen, tada koncept $Muskarac \sqcap \exists godine.\, \geq_{18}$ predstavlja sve punoletne muškarce. ■

6.1.1 Opisne logike i predikatska logika

Opisne logike se mogu predstaviti kao fragmenti logike prvog reda. Logika prvog reda je poluodučiva, ali opisne logike prestvljaju odlučive fragmente logike prvog reda. Podsetimo se da se koncepti interpretiraju kao unarne relacije tako da njihovim imenima možemo pridružiti unarne relacijske simbole. Uloge se interpretiraju kao binarne relacije tako da njihovim imenima možemo pridružiti binarne relacijske simbole.

Definicija 6.1.11 Neka konceptna formula C i njen prevod $\pi(C)$ su ekvivalentni ako za sve interpretacije $(\triangle^I, \cdot^I)$ i za svaki $a \in \triangle^I$ važi: $a \in C^I$ akko $I \vDash \pi(C)(a)$. Prevođenje je očuvavajuće ako je prevod svake formule ekvivalentan formuli. Odgovarajući pojmovi se analogno definišu za uloge i atribute.

Teorema 6.1.12 Za svaku od navedenih logika postoji očuvavajući prevod u fragment logike prvog reda.

Konceptima $\top$, $\bot$ odgovaraju valjane formule, odnosno tautologije i kontradikcije u logici prvog reda. Sada možemo svaku konceptnu formulu C transformisati u formulu predikatske logike $\pi_C(x)$ sa jednom slobodnom promenljivom x. Atomski koncept A može biti transformisan u formulu oblika $A(x)$. Logičkim veznicima (presek, unija, negacija) u opisnim logikama, u predikatskoj logici odgovaraju takođe logički veznici konjukcije, disjunkcije i negacije respektivno. Sledi primer jednog prevođenja za ALC koncepte.

Primer 6.1.13 Ako sa A i R redom označimo atomski koncept, odnosno ulogu, a sa C i D označimo proizvoljne koncepte, tada je:

- $\pi_x(A) = A(x)$,
- $\pi_x(C \sqcap D) = \pi_x(C) \wedge \pi_x(D)$,
- $\pi_x(C \sqcup D) = \pi_x(C) \vee \pi_x(D)$,
- $\pi_x(\exists R.C) = (\exists y)(R(x,y) \wedge \pi_y(C)$,
- $\pi_x(\forall R.C) = (\forall y)(R(x,y) \supset \pi_y(C))$,
- $\pi_y(A) = A(y)$,
- $\pi_y(C \sqcap D) = \pi_y(C) \wedge \pi_y(D)$,
- $\pi_y(C \sqcup D) = \pi_y(C) \vee \pi_y(D)$,
- $\pi_y(\exists R.C) = (\exists x)(R(y,x) \wedge \pi_x(C)$,
- $\pi_y(\forall R.C) = (\forall x)(R(y,x) \supset \pi_x(C))$,

gde su π_x i π_y funkcije koje prevode ALC koncepte u formule predikatske logike sa jednom slobodnom promenljivom.

Za logike koje imaju brojevna ograničenja ($\geq nR$, $\leq nR$) prevođenje se obavlja na sledeći način:

- $\pi_x(\geq nR) = (\exists^{\geq n} y)R(x,y) = (\exists y_1, \ldots, y_n)(\bigwedge_{i \neq j} y_i \neq y_j \wedge \bigwedge_i R(x, y_i))$
- $\pi_x(\leq nR) = (\exists^{\leq n} y)R(x,y) = (\exists y_1, \ldots, y_{n+1})(\bigwedge_{i \neq j} y_i \neq y_j \supset \bigvee_i \neg R(x, y_i))$. Ξ

Primer 6.1.14 Neka su dati sledeći atomski koncepti $Osoba$, $Muskarac$, $Zena$, $Student$. Neka $LJUDI$ predstavlja jedan model, tada ove koncepte svahatamo kao podskupove modela ljudi. U predikatskoj logici ove atomske koncepte možemo predstaviti kao formule sa jednom slobodnom promenljivom, na sledeći način: $Osoba(x)$, $Muskarac(x)Zena(x)$, $Student(x)$. Neka sada $Osoba \sqcap Student$ predstavlja presek koncepata koji označava sve one osobe koje su studenti. Ovaj koncept ima svoju reprezentaciju u predikatskoj logici na sledeći način: $Osoba(x) \wedge Student(x)$. Uniji koncepata $Muskarac \sqcup Student$ odgovara formula $Muskarac(x) \vee Student(x)$ i opisuje sve one individue koje su ili muškarci ili studenti, kao podskupove datog modela. Negacija koncepta ($\neg Student$) opisuje sve one individue koji nisu studenti. Koncept ($\exists Prijatelj.Zena$) opisuje sve one individue u modelu

koji imaju za prijatelja ženu. Ovde vidimo da postoji i jedna atomska uloga Prijatelj. Poslednja formula takođe predstavlja koncept koji je definisan korišćenjem binarne relacije između individua u datom modelu. Ovaj koncept ima odgovarajuću formulu u predikatskoj logici, sa jednom slobodnom promenljivom, napisanu na sledeći način: $(\exists y)(Prijatelj(x, y) \wedge Zena(x))$.

Ako atomsku ulogu sada napišemo uz univerzalni kvantifikator $(\forall Prijatelj.\ Zena)$ dobijamo koncept koji opisuje sve individue koji imaju samo žene za prijatelja. Odgovarajuća formula ovog koncepta u predikatskoj logici prvog reda je $(\forall y)(Prijatelj(x, y) \rightarrow Zena(y))$. Ξ

Na sličan način se mogu formule opisnih logika prevesti u formule dinamičke logike.

6.1.2 TBox i ABox delovi baze znanja

Sistemi bazirani na opisnim logikama omogućava opis dve vrste znanja na nekom domenu:

- Terminološko zanje[7] koje se sastoji od opštih definicija koncepata i znanja o njihovim vezama
- Znanje o specifičnim situacijama[8] koje govori o konkretnim elementima u domenu.

Ove dve vrste znanja opisuju se formulama koje se grupišu u TBox, za terminološko znanje, i ABox za znanje o specifičnoj situaciji.

Terminološko znanje

Terminološko znanje se opisuje formulama u kojima učestvuju koncepti i uloge. Različiti sistemi bazirani na opisnim logikama podržavaju različite vrste TBox-a. Ovde dajemo opštu definiciju TBox-a kao i definiciju nekih ograničenja ovog pojma.

Definicija 6.1.15 Opšte terminologije su formule oblika:

- $(C \sqsubseteq D)$
- $(C \doteq D)$

gde su C i D dati koncepti. Opšti TBox T je konačan skup opštih terminologija.

Definicija 6.1.16 Interpretacija I zadovoljava opštu terminologiju $C \sqsubseteq D (C \doteq D)$ akko $C^I \subseteq D^I$ $(C^I = D^I)$. Interpretacija I zadovoljava TBox T ako i samo ako zadovoljava svaku terminologiju u T. U tom slučaju T se naziva zadovoljiv, a I je u tom slučaju model od T i pišemo $I \models T$.

Ilustrujmo sada primerom jedan opšti TBox.

[7] Terminological knowledge
[8] Assertional knowledge

Primer 6.1.17 Neka su Osoba i Žensko atomski koncepti tada jedan TBox može biti oblika:

$Zena \doteq Osoba \sqcap Zensko$
$Muskarac \doteq Osoba \sqcap \neg Žena$
$Majka \doteq Zena \sqcap \exists DeteOd.Osoba$
$Otac \doteq Muskarac \sqcap \exists DeteOd.Osoba$
$Roditelj \doteq Majka \sqcup Otac.$ Ξ

Definicija 6.1.18 Koncept C je zadovoljiv, u odnosu na TBox T, ako i samo ako postoji model I od T tako da je $C^I \neq \emptyset$. Koncept C je obuhvaćen konceptom D u donosu na TBox T $C \sqsubseteq_T D$ako i samo ako $C^I \subseteq D^I$ za svaki model I od T. Dva koncepta C i D su ekvivalentna i pišemo $C^I \equiv_T D^I$, ako i samo ako $C^I = D^I$ za svaki model I od T.

Dužina TBox T, u oznaci $|T|$ se računa na sledeći način:

$$|T| := \sum_{C \sqsubseteq D \in T} |C| + |D|$$

Znanje o specifičnim situacijama

Drugi deo baze znanja bazirane na opisnim logikama je znanje o specifičnim situacijama, čija je kraća oznaka ABox.

Definicija 6.1.19 Za imena konstanti x, y, atomski koncept C, atomsku ulogu R, aksiome oblika $x : C$, $(x, y) : R, i\ x \neq y$ su aksiome tvrđenja. Jedan ABox A je konačan skup aksioma tvrđenja.

Definicija 6.1.20 Interpretacija I zadovoljava aksiomu tvrđenja $x : C$ ako i samo ako $x^I \in C^I$, zadovoljava aksiomu tvrđenja $(x, y) : R$ako i samo ako $(x^I, y^I) \in R^I$, i zadovoljava aksiomu tvrđenja $x \neq y$ ako i samo ako $x^I \neq y^I$. Interpretacija I zadovoljava ABox A ako i samo ako zadovoljava svaku aksiomu tvrđenja u A. Ako jedna takva interpretacija I postoji, onda je A zadovoljiv. Tada je I model od A u oznaci $I \mathit{fffl}\ A$.

Ilustujmo sada primerom jedan ABox

Primer 6.1.21 Koristeći primer 6.1.2u kome smo opisali jedan TBox rodbinskih veza, ovde ćemo prikazati drugi deo baze znanja (Abox) kojim se opisuju rodbinske veze jedne konkretne grupe osoba

$$(Marija, Petar) : deteOd$$
$$(Marija, Pavle) : deteOd$$
$$(Petar) : Otac$$
$$(Petar, Branko) : deteOd$$

U datom ABox-u konstante su: Marija, Petar, Pavle, Branko. Formula $(Petar) : Otac$ znači da je Petar otac, dok formula $(Marija, Pavle) : deteOd$ znač i da je Pavle Marijino dete. Ξ

Konačno možemo dati definiciju baze znanja zapisane jezikom opisnih logika.

Definicija 6.1.22 Baza znanja (K) je uređeni par (T, A), gde je T TBox a A je ABox. Jedna interpretacija I zadovoljava bazu znanja K ako i samo ako $I \models A$ i $I \models T$.

6.2 Automatsko zaključivanje

Imajući u vidu opisne logike, standardni problemi automatskog zaključivanja mogu se podeliti u nekoliko kategorija i to:

1. Zadovljivost koncepta[9]. Neka je dat neki koncept C. Koncept C je zadovoljiv (u odnosu na TBox T) ako i samo ako postoji interpretacija I tako da je $I \models T$ i $C^I \neq \emptyset$. Ovaj problem se može svesti na problem da li su neki koncepti kontradiktorni.

2. Obuhvatanje koncepata[10]. Neka su data dva koncepta C i D. Postavlja se pitanje da li je koncept C obuhvaćen konceptom D (u odnosu na TBox T). Koristeći relaciju obuhvatanja, koncepti definisani u TBox-u mogu biti uređeni jednim kvazi poretkom u kome se vidi specijalizacija/generalizacija [11] hijerarhije koncepata. Računanje ove hijerarhije jedan od glavnih problema zaključivanja zasnovanog na opisnim logikama, a koji je korišćen u mnogim sistemima baziranim na opisnim logikama (FACT).

3. Zadovoljivost baze znanja[12]. Za datu bazu znanja K proverava se da li je znanje sadržano u toj bazi kontradiktorno ili ne. Zadovoljivost koncepata može biti zamenjen problemom zadovoljivosti baze znanja K.

4. Provera instanci[13]. Data je baza znanja K, ime konstante (konstanta, individua) x, i neki koncept C. Postavlja se pitanje da li je $x^I \in C^I$ za svaki model I od K. U ovom slučaju x se naziva primerak koncepta C u odnosu na bazu znanja K. Moguće je čak izvesti zaključak da je x primerak koncepta C u svakom modelu baze znanja iako se eksplicitno ne pojavljuje u ABox-u. Problem može biti zamenjen i sledećim pitanjem: Ako je data baza znanja $K = (T, A)$, x je primerak koncepta C u odnosu na bazu znanja K ako i samo ako je baza znanja $K(\{T, A \cup \{x : \neg C\}\})$ nezadovoljiva.

Za svaku opisnu logiku prve dve metode automatskog zaključivanja se mogu formulisati na drugi način. Dati koncept C je nezadovojiv u odnosu na TBox T ako i samo ako je $C \sqsubseteq_T \bot$, a sa druge strane $C \sqsubseteq_T D$ ako i samo ako je koncept $C \sqcap \neg D$ nezdovoljiv u odnosu na TBox T. Ove dve tehnike zaključivanja se koriste samo u odnosu na TBox. Razlog leži u tome što ABox ne izvodi zaključke na problemima ovakvog tipa.

Prethodno pomenute tehnike zaključivanja možemo opisati sledećom definicijom:

[9] Concept satisfiability.
[10] Concept subsumption
[11] Specialistion/Generalisation.
[12] Knowledge base satisfiability
[13] Instance Checking

Definicija 6.2.1 Za datu bazu znanja $K = (T, A)$, dva koncepta C i D i konstantu a, kažemo da je:

- zadovoljivost koncepta[14], u oznaci $K \nvDash C \equiv \bot$, problem proveravanja da li postoji model I baze znanja K tako da je $C^I \neq \emptyset$.
- obuhvatanje koncepata[15], napisano kao $K \models C \sqsubseteq D$,je problem određivanja da li je $C^I \subseteq D^I$ u svakom modelu I baze znanja K.
- konzistentnost[16], napisana kao $K \nvDash$, je problem određivanja da li je baza znanja K zadovoljiva.
- provera instanci [17], napisana kao K *ffl* $C(a)$, je problem određivanja, da li je formula $C(a)$ zadovoljiva u svakom modelu baze znanja K.

Za navedene tehnike zaključivanja razvijeno je više procedura među kojima je i adaptivna metoda tabloa. U grupi rezultata povezanih sa kompleksnošću nalaze se i:

- za logike ALC i $ALCNR$ obuhvatanje koncepata je PSPACE-kompletan problem i
- za logike ALC i $ALCNR$ zadovoljivost koncepata je NP-kompletan problem.

6.3 Primena opisne logike u semantičkom web-u

Za razliku od postojećeg weba gde su podaci prvenstveno namenjeni upotrebi od strane ljudi, semantički web obezbeđuje takve podatke koje mogu obraditi i mašine. Time se obezbeđuje podrška širokom opsegu inteligentnih usluga poput informativnih brokera, pretraživačkih agenata, usluga filtriranja informacija. Obzirom da je trenutno web strogo orijentisan ka obradi od strane ljudi, nemogućnost obrade sadržaja informacija od strane mašina ozbiljno remeti njegovu funkcionalnost. Računari su ograničeni na prenos i prikaz informacija na webu, i nemaju stvarnu ulogu u obradi ovih informacija. Vizija semantičkog weba omogućuje kreiranje takvog weba u kome bi sve informacije bile razumljive mašinama podjednako kao i ljudima. Da bi se to postiglo, neophodno je da informacije budu date sa tačnim značenjem, kodirane i strukturirane sa mašinski čitljivim opisom svojih sadržaja, čime bi se olakšana automatska obrada i integrisanje informacija dostupnih na webu. Da bi "mašine" mogle pravilno da zaključuju tokom obrade ovih podataka, jeziku mora prethoditi osnovna semantika. Takode, cilj sematičkog weba je i podizanje nivoa efikasnosti weba u odnosu na korisnike. Ovo pre svega znači - omogućiti na webu neke servise koji trenutno nisu zastpljeni: lociranje informacija, uporedivanje i ukrštanje informacija, zaključivanje na osnovu pronađenih informacija iz dva ili više odvojenih izvora. Mašinski čitljive i kodirane strukture resursa koje koristi semantički web eksplicitno se nazivaju ontologijama čije značenje možemo opisati sledećom definicijom:

[14]Concept satisfiability
[15]Subsumption.
[16]Consistency.
[17]Instance checking

Definicija 6.3.1 Ontologija O je apstrakcija domena D izražena trojkom $O = (C, R, isa)$ gde je:

- $C = \{c_i | i = 1, n\}$ skup koncepata domena D
- $R = \{r_i | i = 1, n\}$ skup relacija između koncepata
- isa-predstavlja skup veza nasleđivanja između koncepata

Korišćenjem ontologija pretraživanje podataka na internetu prelazi sa tekućeg pretraživanja po ključnoj reči na pronalaženje informacija koje su sintaktički različite ali predstavljaju semantički bliske reči. Kao rezultat istraživačkih napora u oblasti semantičkog Web-a postoji veći broj različiih jezika i alata koji se koriste za prikaz ontologija koji je skladu sa postojećim web standardima. Korišćenje ontologija takođe zahteva i dobro definisane, dobro dizajnirane alate za zaključivanje. Sintaksa ontoloških jezika(OWL, AOL) mora biti intuitivno razumljiva za korisnike i kompatibilna sa postojećim web standardima . Ovi jezici treba da budu toliko izražajni da se relevantni koncepti mogu definisati u dovoljno detalja ali ne toliko izražajni da zaključivanje bude neefikasno. Zaključivanje je veoma bitno za utvrđivanje kvaliteta pri projektovanju ontologije. Za vreme projektovanja ontologije veoma je važno utvrditi koji su koncepti kontradiktorni. Posebno je zanimljivo pri projektovanju ontologije uz pomoć zakljlučivača utvrditi hijerarhiju koncepata tj. koji je koncept specijalizacija nekog drugog koncepta. Zaključivači su veoma važni pri utvrđivanju koji koncepti su sinonimi. Veoma važna osobina automatskog zaključivanja na Web-u je i mogućnost integracije različitih ontologija.

Primer 6.3.2 Pretpostavimo da su nam na raspolaganju ontologije koje se odnose na automobile i da želimo da dobijemo informaciju o svim automobilima koji imaju određene osobine i cenu u određenom intervalu. Naš upit će tada sadržati opis karakteristika automobila i prihvatljivi opseg cene. U klasičnom pretraživanju web-a pomoću sistema kao što je google, dobili bismo niz adresa internet stranica koje sadrže reči iz našeg upita. Sa druge strane mehanizam koji podržava semantički web trebalo bi da pruži odgovor u skaldu sa sadržajem upita.

Slično pretpostavimo da imamo ontologije rečnika i da nas interesuje prevod neke naše reči, kao i njeni sinonimi na srpskom ali i na jezicima sa kojih dobijamo prevod. Klasičan pretraživač interneta nam obezbeđuje sve stranice na kojima se nalaze reč i njeni prevodi. Međutim, on nije u stanju da poveže sinonime naše reči i njihove prevode i da nam i tu listu vrati kao odgovor. Ξ

6.3.1 Za dalje proučavanje

Osnovna referenca za oblast opisnih logika je [8]. Logike sa konkretnim domenima analizirane su u [75, 83]. O primeni opisnih logika u bazama znanja govori se u [6, 24], a kod semantičkog weba u [7]. Veza opisnih logika i predikatske logike prvog reda data je u [11]. Adaptacija metode tabloa za potrebe opisnih logika, kao i primar implementacije dokazivača mogu se pronaći u [52].

7

Neklasične logike

U mnogim situacijama je korisno prevazići ograničenja koja nameće klasična logika, kako po pitanju formalnog jezika, tako i zbog pretpostavke da su svi iskazi bilo tačni bilo netačni. Pod nazivom neklasične (ili nestandardne) logike podrazumevaju se sve logike koje su razvijene kao odgovor na taj zahtev. Neklasične logike se dele u dve grupe:

- proširenja klasične logike i
- zamene za klasičnu logiku.

U prvu grupu spadaju logike koje koriste bogatiji formalni jezik nego klasične logike čime se povećava izražajnost, dok se očuvavaju klasični način definisanja istine i odgovarajuće valjane formule. Glavni predstavnici ove grupe su modalne logike i razne njihove varijante, poput temporalnih logika, dinamičkih logika, logika znanja i/ili verovanja, verovatnosnih logika itd. U drugoj grupi se nalaze logike koje koriste isti formalni jezik kao i klasične logike, ali se od njih razlikuju drugačijim tretiranjem istinitosnih vrednosti ili odbacivanjem nekih valjanih formula. Logike koje su svojevrsni suparnici klasične logike su: viševrednosne logike, fazi logika[1], intuicionistička logika, logike za nemonotono rezonovanje itd.

Razne neklasične logike se razlikuju u tretiranju pojedinih situacija, što može izazvati zabunu. Ponovićemo da zadatak matematičke logike nije da između ovih raznovrsnih logika odredi najbolju, već da proučava mehanizme na kojima te logike počivaju.

Do kraja poglavlja će biti prikazane neke neklasične logike i njihove primene, dok će u poglavlju 8 biti detaljno analizirane verovatnosne logike kao posebno interesantna grupa neklasičnih logika.

[1] Izvorni engleski naziv za fazi logike je *fuzzy logic*. Značenje reči fuzzy je nejasno, razmazano, pa su ovo logike koje barataju sa informacijama za koje nam potpun istinitosni status nije poznat, a koje su izrečene nepreciznim, višeznačnim, prirodnim jezikom. Zbog nepostojanja odgovarajućeg termina na našem jeziku, reč fazi se odomaćila i široko je prihvaćena, tako da ćemo je u nastavku teksta i mi koristiti.

7.1 Modalne logike

Modalne logike su logike iznijansirane istine. Pored iskaza koji su tačni ili netačni postoje i iskazi koji su nužno tačni, dokazivi, obavezni, iskazi koji su mogući itd. Recimo, iskazi 'Mesec su osvojili vanzemaljci ili Divac je košarkaš' i 'Danas je petak ili danas nije petak' su po formi slični i mogu se zapisati pomoću disjunktivne formule $\alpha \vee \beta$. Međutim, između njih postoji suštinska razlika. Za prvi iskaz se zna da je tačan na osnovu iskustva. Svako ko je pratio utakmice košarkaške reprezentacije čuo je za Vladu Divca. Drugi iskaz je tačan na osnovu svoje forme $\alpha \vee \neg\alpha$, pri čemu nije potrebno nikakvo iskustvo, odnosno ne mora se znati koji je danas dan. Ovaj drugi iskaz je primer iskaza koji je nužno tačan. Slično, sve tautologije, odnosno u predikatskom slučaju valjane formule, su nužno tačne.

Jedan od zadataka modalnih logika je da opišu načine rezonovanja sa običnim i kvalifikovanim, recimo nužnim ili mogućim, istinama. Primeri pitanja sa kojima se modalne logike susreću su: ako je nešto nužno tačno, da li je nužno da je to nužno tačno, ili da li postoji mogućnost da neki iskaz bude tačan i sl.

Jezik modalnih logika nastaje proširivanjem jezika klasične logike unarnim operatorom $\Box$, dok se operator $\Diamond$ definiše sa

$$\Diamond\alpha =_{def} \neg\Box\neg\alpha.$$

tako da je omogućena dekompozicija iskaza poput 'Moguće je živeti bez kiseonika'.

Formule $\Box\alpha$ i $\Diamond\alpha$ se mogu shvatiti na više načina:

- $\Box\alpha$ se shvata kao 'α je logički nužno', u kom slučaju se $\Diamond\alpha$ čita kao 'α je logički moguće i ne protivreči logičkim zakonima',
- $\Box\alpha$ se shvata kao 'agent zna α', dok bi $\Diamond\alpha$ značilo da 'α nije u kontradikciji sa onim što se zna',
- slična je i interpretacija pri kojoj se $\Box\alpha$ shvata kao 'agent veruje u α',
- $\Box\alpha$ se shvata kao 'α je dokazivo u nekom formalnom sistemu', a $\Diamond\alpha$ kao 'α nije u kontradikciji sa formalnim sistemom',
- $\Box\alpha$ se shvata kao 'α je zakonska obaveza' dok se $\Diamond\alpha$ shvata kao 'α je dozvoljeno',
- $\Box\alpha$ se shvata kao 'α je tačno zauvek u budućnosti, a $\Diamond\alpha$ kao 'α će se ostvariti u budućnosti' itd.

Da bismo olakšali izražavanje u nastavku ovog odeljka ćemo operatore $\Box$ i $\Diamond$ redom zvati nužno i moguće iako se nećemo ograničiti na njihovu prvu interpretaciju. U kasnijim odeljcima ćemo, međutim, razmotriti i neke specifične interpretacije. Uglavnom ćemo razmatrati iskazne modalne logike, i to ne sve, već takozvane *normalne*, mada će se značajan deo izložene materije odnositi kako na preostale iskazne modalne logike, tako i na predikatske modalne logike.

7.1.1 Modalni operatori

Raznovrsnost interpretacija čini da su neke modalne formule prihvatljive u jednom pristupu, dok to nisu u nekom drugom. Na primer, formula $\Box\alpha \to \alpha$ je prihvatljiva kada operator $\Box$ shvatimo kao 'znati', pri čemu se formula čita kao 'ako znam da je α, onda je α tačno', dok to nije slučaj ako operatoru damo vremensku interpretaciju 'ako je α tačno uvek u budućnosti, onda je tačno i sada.' Međutim, postoje principi, koji su opšteprihvaćeni bez obzira na tumačenje koje dajemo operatoru $\Box$. Neki od tih principa su:

- modalni operatori nisu istinitosno funkcionalni[2] pa ni jedna od sledećih formula ne sme biti valjana:

  $\Box\alpha \leftrightarrow \neg\alpha$,

  $\Box\alpha \leftrightarrow \alpha$,

  $\Box\alpha \leftrightarrow (\alpha \vee \neg\alpha)$,

  $\Box\alpha \leftrightarrow (\alpha \wedge \neg\alpha)$ i

  slično za operator $\Diamond$,

- primerci klasično valjanih formula su nužno istiniti i

- ako iskaz β nužno sledi iz iskaza α, tj. važi $\Box(\alpha \to \beta)$ i izkaz α je nužan, onda je nužan i iskaz β, odnosno valjana je formula

  $$(\Box(\alpha \to \beta) \wedge \Box\alpha) \to \Box\beta,$$

  koja se shvata i na način da sve što nužno proizilazi iz nužne istine takođe je nužna istina.

Drugi uslov u nužno istinite formule ubraja sve primerke klasično valjanih formula. Kada bi se tu spisak nužno istinitih formula završio, razvoj modalnih logika ne bi ni bio potreban. Međutim, na osnovu prethodnih uslova ne možemo ništa reći o formulama kao što su: $\Box\alpha \to \alpha$ ili $\Box\alpha \to \Box\Box\alpha$, dakle o formulama koje govore o odnosu modaliteta. Prva od ove dve formule kaže da ako je nešto nužno tačno, onda je i tačno, a druga da, ako je nešto nužno tačno, onda je nužno da je nužno tačno.

Kao što ćemo videti, postoje sistemi u kojima su neke ovakve modalne formule valjane, dok te iste formule to nisu u drugim sistemima.

7.1.2 Iskazni Kripkeovi modeli za modalne logike

U preciznom davanju značenja modalnim formulama pre svega se koristi pristup sa mogućim svetovima koji se naziva i relaciona semantika.

Kao i u slučaju klasične logike, iskazi su ili tačni ili netačni. Recimo, razumno je iskaz 'Osobi A kiseonik nije potreban za život' proglasiti za netačan, ali precizno govoreći, iskaz je takav u zemaljskim uslovima. U nekom drugačijem svetu u kome se životne forme razlikuju od ovdašnjih, moguće je da iskaz bude istinit. Modeli za

[2]Istinitosna vrednost formule koja počinje nekim od modalnih operatora nije funkcija istinitosnih vrednosti podformula te formule.

modalne logike se upravo baziraju na istovremenom postojanju više svetova koji svaki za sebe predstavlja klasičnu iskaznu interpretaciju, a između kojih postoji relacija dostižnosti. Preciznije,

Definicija 7.1.1 Neka je Φ skup iskaznih slova. Uređena trojka $\langle W, R, v\rangle$ je *iskazni Kripkeov*[3] *model*, ako je ispunjeno:

- W je neprazan skup čije se elementi nazivaju *(modalni) svetovi* ili *stanja*,
- R je binarna relacija nad W ($R \subset W \times W$) koja se naziva *relacija dostižnosti* ili *relacija vidljivosti* i
- v je *valuacija*, funkcija $v : W \times \Phi \to \{\top, \bot\}$ koja svakom svetu i svakom iskaznom slovu pridružuje jednu od istinitosnih vrednosti $\top$ i $\bot$.

Umesto iskazni Kripkeov model koristi se i nazivi *iskazni modalni model*, *modalni model* ili samo *model*.

Primetimo da je za svaki svet w nekog modela $v(w)$ upravo klasična iskazna interpretacija, jer za svako iskazno slovo $p \in \Phi$, $v(w)(p) \in \{\top, \bot\}$. Ako za dva sveta w i u nekog iskaznog Kripkeovog modela $\langle W, R, v\rangle$ važi da je wRu, kažemo da je svet u dostižan (ili vidljiv) iz sveta w. Takođe, uočimo i da se u klasičnom slučaju reč model koristila u smislu interpretacije pri kojoj formula važi što se značajno razlikuje od značenja iz prethodne definicije u kojoj je model semantička struktura u kojoj formule dobijaju značenje u skladu sa sledećom definicijom.

Definicija 7.1.2 Neka je $M = \langle W, R, v\rangle$ iskazni Kripkeov model. Relacija (*modalne*) *zadovoljivosti* (u oznaci $\models$) je binarna relacija između svetova modela i formula takva da za svaki svet $w \in W$ važi:

- ako je $p \in \Phi$, $w \models p$ ako i samo ako $v(w)(p) = \top$,
- $w \models \neg\alpha$ ako i samo ako nije $w \models \alpha$,
- $w \models (\alpha \wedge \beta)$ ako i samo ako $w \models \alpha$ i $w \models \beta$ i
- $w \models \Box\alpha$ ako i samo ako za svaki svet u koji je dostižan iz w važi $u \models \alpha$.

Ako je $w \models \alpha$, formula α je *zadovoljena* u svetu w.

Prva tri uslova za relaciju zadovoljivosti su u skladu sa odgovarajućom relacijom u klasičnoj logici, dok je poslednji zahtev karakterističan. Da bi formula $\Box\alpha$ bila zadovoljena u svetu w, tražimo da formula α mora biti zadovoljena u svakom od svetova dostižnih iz sveta w. Ako je $w \models \alpha$, kaže se i α *važi* u svetu w ili α je *tačno* u svetu w. Oznaka $w \not\models \alpha$ se koristi ako α ne važi u svetu w.

Primer 7.1.3 Neka je $\langle\{w_1, w_2\}, \{(w_1, w_1), (w_1, w_2), (w_2, w_1)\}, v\rangle$ iskazni Kripkeov model takav da je skup svetova dvočlan, drugi svet je dostižan iz prvog, a prvi iz drugog i iz sebe samog, dok je valuacija v takva da je $v(w_1)(p) = \top$, $v(w_1)(q) = \top$,

[3]Ime je dato u čast Saul-a Kripke-a (1940) koji je prvi na zadovoljavajući način semantički opisao modalne logike i pokazao kompletnost različitih modalnih logika u odnosu na odgovarajuće klase ovih modela.

$v(w_2)(p) = \top$, $v(w_2)(q) = \bot$. U skladu sa definicijom 7.1.2, $w_1 \models p$, $w_1 \models q$, $w_1 \models p \wedge q$, $w_2 \models p$, $w_2 \models \neg q$, $w_2 \not\models p \wedge q$. Dalje je $w_1 \models \Box p$, jer p važi u svim svetovima dostižnim iz w_1 i $w_2 \models \Box q$, dok nije $w_1 \models \Box q$. Slično $w_1 \not\models \Box(p \wedge q)$ i $w_2 \models \Box(p \wedge q)$. Primetimo da je, recimo, $w_2 \models \Box q$ iako nije $w_2 \models q$. Ξ

Primer 7.1.4 Neka je model M određen skupom stanja $W = \{w, u, t\}$, relacijom $R = \{(w,w),(w,u),(u,w),(u,u),(t,t)\}$ i valuacijom $v(w)(p) = v(t)(p) = \top$, $v(u)(p) = \bot$. Sada je $w \not\models \Box p$ i $t \models \Box p$. Ξ

U primeru 7.1.4 se uočava da postoje stanja modela u kojima se valuacije poklapaju, ali koja ipak nisu ista, pošto se razlikuju po pitanju dostižnih svetova. Recimo, u svetovima w i t jednaka je istinitosna vrednost iskaznog slova p, ali pošto je iz w dostižan svet u u kome p ne važi, $\Box p$ važi u t, ali ne i u svetu w.

Na osnovu definicije modalnog operatora $\Diamond$ ($\Diamond =_{def} \neg\Box\neg$) lako se proverava da je:

- $w \models \Diamond\alpha$ ako i samo ako u modelu postoji svet u dostižan iz sveta w tako da je $u \models \alpha$.

Slično je i sa preostalim klasičnim iskaznim operatorima $\vee$, $\rightarrow$ i $\leftrightarrow$.

Primer 7.1.5 U primeru 7.1.3 važi $w_1 \models \Diamond p$, $w_1 \models \Diamond q$, $w_1 \models \Diamond\neg q$, $w_1 \models \Diamond q \vee \Diamond\neg q$ itd. Ξ

Definicija 7.1.6 Formula α je *zadovoljiva* ako postoji neki svet w nekog iskaznog Kripkeovog modela za koji je $w \models \alpha$. Ako takav svet ne postoji, formula je *nezadovoljiva*. Formula α je *valjana* u nekom iskaznom Kripkeovom modelu ako je zadovoljena u svakom svetu tog modela.

7.1.3 Klase iskaznih Kripkeovih modela

Zavisno od tipa relacije dostižnosti u iskaznim Kripkeovim modelima razmatraju se razne klase modela. Za neki model $\langle W, R, v\rangle$ kažemo da ima neko svojstvo ako je to slučaj sa odgovarajućom relacijom dostižnosti. Na primer, model je simetričan ako je relacija dostižnosti između svetova simetrična. U tabeli 7.1 su prikazane najpoznatije klase iskaznih Kripkeovih modela[4].

Definicija 7.1.7 Formula α je *valjana u klasi iskaznih Kripkeovih modela* C ili *C-valjana* ako je valjana u svakom modelu iz te klase.

Prethodne klase modela se očigledno ne poklapaju. Na primer, lako je zamisliti model koji jeste tranzitivan, ali nije tranzitivan i refleksivan. Klasa modela C_1 sadrži sve modele klase C_2 u kojoj su uslovi za relaciju dostižnosti strožiji, tako da su svi $S4$-modeli istovremeno i T-modeli itd. Ako za dve klase modela C_1 i C_2 važi $C_2 \subset C_1$, onda je svaka C_1-valjana formula istovremeno i C_2-valjana. Recimo, svaka T-valjana formula je i $S4$-valjana. Sledeći primer na slučaju klasa T i $S4$ ilustruje da u opštem slučaju obrnuto ne važi, tj. ako je $C_2 \subset C_1$ i $C_1 \not\subset C_2$, onda ne mora svaka C_2 valjana formula biti i C_1 valjana.

[4] Atribut *idealnosti* ima ona relacija dostižnosti u kojoj za svaki svet w modela postoji bar jedan svet u tako da je wRu, tj. postoji bar jedan svet u koji je dostižan iz w.

Naziv klase modela	Uslovi za relaciju dostižnosti
K	bez uslova
T	refleksivnost
$K4$	tranzitivnost
KB	simetričnost
$S4$	refleksivnost i tranzitivnost
B	refleksivnost i simetričnost
$S5$	refleksivnost, simetričnost i tranzitivnost
D	idealizacija
$D4$	idealizacija i tranzitivnost
DB	idealizacija i simetričnost

Tabela 7.1. Klase iskaznih Kripkeovih modela.

Primer 7.1.8 Razmotrimo formulu $\Box p \to \Box\Box p$, T-model $\langle W, R, v\rangle$, u kom je $W = \{w_1, w_2, w_3\}$, $R = \{(w_1, w_1), (w_1, w_2), (w_2, w_2), (w_2, w_3), (w_3, w_3)\}$ i $w_1 \models p$, $w_2 \models p$ i $w_3 \not\models p$. Na osnovu definicije relacije $\models$ zaključujemo da: $w_1 \models \Box p$, jer p važi u svim svetovima dostižnim iz w_1, $w_2 \not\models \Box p$, jer p ne važi u svetu w_3 i $w_1 \not\models \Box\Box p$, jer $\Box P$ ne važi u svetu w_2. Odatle nije $w_1 \models \Box p \to \Box\Box p$, jer $\Box p$ važi u svetu w_1, a $\Box\Box p$ ne, pa razmatrana formula nije T-valjana.

Neka je sada $\langle W', R', v'\rangle$ proizvoljni $S4$-model. Pretpostavimo da formula $\Box p \to \Box\Box p$ nije valjana u tom modelu i neka je $w \in W'$ jedan svet modela u kome ne važi formula. To znači da $w \models \Box p$ i $w \not\models \Box\Box p$. Odatle postoji bar jedan svet u dostižan iz w u kome važi p (jer $w \models \Box p$), a ne važi $\Box p$ (jer $w \not\models \Box\Box p$). To, opet, znači da postoji i svet t dostižan iz u u kome ne važi p (jer $u \not\models \Box p$). No, kako je relacija dostižnosti tranzitivna i zbog toga svet t dostižan zbog toga iz sveta w, a p ne važi u t, to $\Box p$ ne može važiti u w, što je suprotno polaznoj pretpostavci. Dakle, formula $\Box p \to \Box\Box p$ jeste S4-valjana, ali nije T-valjana. Ξ

Činjenica da formula $\Box p \to \Box\Box p$ nije T-valjana, ne znači da ne postoje refleksivni modeli koji nisu tranzitivni u kojima je formula valjana. Da bi se to videlo, dovoljno je razmotriti bilo koji model iz klase $T \setminus S4$ u čijim svim svetovima važi iskazno slovo p. Međutim, pokazuje se da je na izvestan način (o kome ovde neće biti precizno govoreno[5]) svaka od navedenih osobina relacije dostižnosti karakterisana nekom posebnom formulom. Tako formula $\Box\alpha \to \Box\Box\alpha$ upravo odgovara tranzitivnosti relacije dostižnosti.

Možemo zaključiti da je klasa K-modela najširа klasa među navedenim, dok joj odgovara najuži skup valjanih formula. Obrnuto, najužoj klasi modela, klasi $S5$-modela odgovara najširi skup valjanih formula.

[5]Umesto klasa modela razmatraju se *modalni okviri*, uređeni parovi koji se sastoje od skupa svetova i relacije dostižnosti, i relacija zadovoljivosti definisana na njima.

7.1.4 Najpoznatije iskazne normalne modalne logike

Iskazne *normalne modalne logike* spadaju u one modalne logike koje se najpravilnije ponašaju[6]. To se odnosi na pogodnu sintaksnu i semantičku definabilnost, relativno jednostavne dokaze potpunosti i odlučivosti, postupke za ispitivanje valjanosti itd. Ako modalnu logiku shvatimo kao skup valjanih formula, onda su sve logike koje odgovaraju klasama modela opisanim u tabeli 7.1 normalne. Lako se pokazuje sledeća teorema.

Teorema 7.1.9 Ako je C bilo koja klasa modela definisana uslovima za relaciju dostižnosti datim u tabeli 7.1, onda skup svih C-valjanih formula:

1. sadrži sve tautologije, tj. sve primerke shema koje su tautologije,
2. sadrži sve primerke sheme $\Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta)$,
3. sadrži β kad god sadrži α i $\alpha \to \beta$ i
4. sadrži $\Box\alpha$ kad god sadrži α.

Dokaz. Razmotrimo slučaj (2). Neka je w proizvoljan svet proizvoljnog modela neke od razmatranih klasa modela. Ako $w \not\models \Box(\alpha \to \beta)$, onda je trivijalno $w \models \Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta)$. Zato pretpostavimo da je $w \models \Box(\alpha \to \beta)$. Ako je dalje $w \not\models \Box\alpha$, direktno se ustanovljava da traženo važi, jer je desna strana implikacije tačna. Neka je zbog toga $w \models \Box(\alpha \to \beta) \wedge \Box\alpha$. To znači da u svakom svetu u dostižnom iz w važi $u \models \alpha \to \beta$ i $u \models \alpha$. Koristeći definiciju klasičnog iskaznog operatora $\to$ imamo da je $u \models \beta$. Kako je u proizvoljan svet dostižan iz w, onda je $w \models \Box\beta$. Prema tome, za proizvoljan svet w važi $w \models \Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta)$, odakle je posmatrana formula C-valjana. Ξ

Zapravo, pod *normalnim modalnim logikama* se podrazumevaju one logike koje ispunjavaju sve zahteve iz teoreme 7.1.9. Ova definicija zasnovana je na semantičkim kriterijumima.

Drugi način za opisivanje normalnih modalnih logika je sintaksni, pomoću aksiomatskih sistema. Sledeći skup shema aksioma:

svi primerci klasičnih iskaznih teorema i

(**K**)[7] $\Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta)$

i pravila izvođenja

iz α i $\alpha \to \beta$ izvesti β i

(**N**)[8] iz α izvesti $\Box\alpha$

[6]Pored normalnih modalnih logika postoje i one koje to nisu, a nazivaju se ne-normalne modalne logike. Na primer, *regularne logike* zadovoljavaju sve uslove iz teoreme 7.1.9, sem uslova 2, umesto koga se zahteva da logika koja sadrži formulu $\alpha \to \beta$ sadrži i formulu $\Box(\alpha \to \beta)$.

[7]Ova aksioma se naziva aksioma **K**, u čast Kripkea.

[8]Ovo pravilo izvođenja se naziva *necesitacija*.

predstavlja pandan uslovima iz teoreme 7.1.9. Označimo ovaj aksiomatski sistem sa K, kao što je to bilo urađeno i za najopštiju klasu modela. Ovo nije slučajno. Naime, sredstvima koja sliče onima koja su korištena u odeljku 3.6 pokazuje se teorema 7.1.10, tj. teorema potpunosti za klasu K-valjanih formula. Pre iskaza ove teoreme uočimo jednu značajnu razliku između klasične i modalne logike: teorema dedukcije 3.6.3 koja je bila osnova dokaza kompletnosti u odeljku 3.6 u modalnom slučaju ne važi. Obrazloženje za to je sledeće. Pošto je $\alpha \vdash \alpha$, primenom pravila **N**, dobili bismo $\alpha \vdash \Box\alpha$. Ako bi, pri tom, važila teorema dedukcije, važilo bi i $\vdash \alpha \to \Box\alpha$. Međutim, formula $\alpha \to \Box\alpha$ nije valjana ni u jednoj od normalnih modalnih logika, pa teorema dedukcije ne važi.

Teorema 7.1.10 Formula je K-valjana ako i samo ako je K-teorema.

Dokaz. Uz izvesne izmene koje su posledice činjenice da ne važi teorema dedukcije, dokaz se svodi na postupak opisan u odeljku 3.6. Najpre se definišu konzistentni i nekonzistentni skupovi formula:

- skup formula $\{\alpha_1, \ldots, \alpha_n\}$ nije konzistentan ako je $\vdash \neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$,
- beskonačan skup formula nije konzistentan ako bar jedan njegov konačan podskup nije konzistentan i
- skup je konzistentan ako i samo ako nije nekonzistentan.

Zatim se pokazuje da se svaki konzistentan skup može proširiti do maksimalno konzistentnog skupa formula. Svi maksimalni konzistenti skupovi formula predstavljaće svetove takozvanog kanonskog modela u kome se valuacija definiše tako da za iskazno slovo $p \in \Phi$, $v(w)(p) = \top$ ako i samo ako $p \in w$. Relacija dostižnosti se uvodi sa wRu ako i samo ako $\{\alpha : \Box\alpha \in w\} \subset u$. dokaz se završava tako što se pokaže da za svaku formulu α i svaki svet w važi $\alpha \in w$ ako i samo ako $w \models \alpha$. Time je pokazano da je svaki konzistentan skup formula zadovoljiv, odakle kao i u slučaju klasične iskazne logike sledi traženo tvrđenje. □

Karakteristične aksiome za pojedine normalne modalne logike su:

(**D**) $\Box\alpha \to \Diamond\alpha$

(**T**) $\Box\alpha \to \alpha$

(**4**) $\Box\alpha \to \Box\Box\alpha$

(**B**) $\alpha \to \Box\Diamond\alpha$

(**5**) $\Diamond\alpha \to \Box\Diamond\alpha$

U tabeli 7.2 su prikazani aksiomatski sistemi koji odgovaraju spominjanim normalnim modalnim logikama. Potpunost se za ove sisteme pokazuje u skladu sa postupkom iz teoreme 7.1.10, pri čemu karakteristične aksiome obezbeđuju da relacija dostižnosti ima potrebna svojstva. Na primer, pošto aksioma (**T**) pripada svim maksimalnim konzistentnim skupovima, tj. svim svetovima kanonskog modela, za svaku formulu $\Box\alpha$ koja pripada nekom svetu w kanonskog modela imamo da je i

T	sistem K + aksioma **T**
$K4$	sistem K + aksioma **4**
KB	sistem K + aksioma **B**
$S4$	sistem K + aksiome **T** i **4** = sistem T + aksioma **4**
B	sistem K + aksiome **T** i **B** = sistem T + aksioma **B**
$S5$	sistem K + aksiome **T** i **5** = sistem $S4$ + aksioma **B** ili **5**
D	sistem K + aksioma **D**
$D4$	sistem K + aksiome **D** i **4** = sistem D + aksioma **4**
DB	sistem K + aksiome **D** i **B** = sistem D + aksioma **B**

Tabela 7.2. Aksiomatski sistemi za normalne modalne logike.

$\alpha \in w$. Odatle je $\{\alpha : \Box\alpha \in w\} \subset w$, pa je relacija dostižnosti u kanonskom modelu refleksivna.

Za spomenute normalne modalne logike se može dokazati još jedno važno svojstvo. Sve ove logike su odlučive, tj. postoji postupak koji u konačnom broju koraka za proizvoljnu formulu utvrđuje da li je valjana ili nije, odnosno da li je zadovoljiva ili nije. U odeljku 7.2 prikazaćemo metod prefiksiranih tabloa koji rešava ove probleme.

7.1.5 Modalne logike sa više verzija modalnog operatora $\Box$

Praksa je pokazala da je ponekad potrebno istovremeno razmatrati više verzija modalnog operatora $\Box$. Na primer, ako je reč o vremenskoj interpretaciji jedna vrsta operatora se shvata kao 'uvek u budućnosti', a druga kao 'uvek u prošlosti'. Ako se operator $\Box$ interpretira kao 'znati', razne verzije operatora $\Box$ se odnose na znanje raznih osoba.

Neka su u modalnom jeziku verzije operatora $\Box$ označene sa $\Box_1, \ldots, \Box_n$. Na semantičkom nivou njima odgovaraju posebne relacije dostižnosti, tako da je modalni model oblika $M = \langle W, R_1, \ldots, R_n, v\rangle$ i za svaki svet $w \in W$ važi:

- $w \models \Box_i\alpha$ ako i samo ako za svaki svet u za koji je wR_iu važi $u \models \alpha$.

Na sintaksnom nivou relacijama odgovaraju verzije karakterističnih modalnih aksioma. Na primer, ako je relacija R_i refleksivna, onda je aksioma

$(\mathbf{T}_i)$ $\Box_i\alpha \to \alpha$.

Dodavanjem ovakvih aksioma dobijaju se odgovarajući korektni i potpuni aksiomatski sistemi. Takođe, dobijene logike ostaju odlučive, ali se u pojedinim situacijama povećava njihova složenost.

7.1.6 Predikatske modalne logike

Modalne logike prvog reda nastaju proširivanjem klasične predikatske logike modalnim operatorima, kao što je to rađeno i u iskaznom slučaju. Na semantičkom nivou se svakom svetu modalnog modela pridružuje domen, skup objekata koji postoje u

tome svetu. Ovo komplikuje situaciju u odnosu na iskazni slučaj, jer je potrebno izabrati jednu od sledećih mogućnosti:

- svi svetovi imaju isti domen, u kom slučaju se radi sa modelima sa konstantnim domenom,
- domen raste, tj. ako je wRu, onda je domen sveta w podskup domena sveta u u kom slučaju se radi sa modelima monotonim domenima i
- ne postoji nikakvo ograničenje za domene svetova modela.

Ako modeli nisu konstantni, potrebno je definisati šta se dešava sa istinitosnom vrednošću formula u nekom svetu, ako se pri interpretaciji formule pojavljuje objekat koji ne postoji u domenu tog sveta. Tada je moguće vrednost formule proglasiti:

- uvek netačnom,
- nedefinisanom ili
- bez obzira na sve dati joj neku istinitosnu vrednost.

Konačno, potrebno je u svakom od prethodnih slučajeva odrediti da li su funkcijski simboli:

- rigidni (u svim svetovima imaju isto značenje) ili
- nerigidni (u raznim svetovima mogu imati različito značenje).

Izbor svake od navedenih mogućnosti dovodi do posebne vrste modalnih modela prvog reda, posebne aksiomatizacije itd. Recimo, ako se dozvole nerigidni termi, neki primerci klasično valjanih formula više nisu valjani, kao što je ilustrovano u primeru 7.1.11.

Primer 7.1.11 Formula $(\forall x)A(x) \to A(t/x)$, gde je term t slobodan za promenljivu x u formuli A, iako primerak jedne aksiome klasične predikatske logike prvog reda, nije modalno valjana ako se dozvole termi koji nisu rigidni. Neka je formula $A(x) = \Diamond P(x)$, gde je P unarni relacijski simbol, i neka je term t konstanta c. Neka za modalni model važe sledeće pretpostavke:

- skup svetova W modalnog modela je dvočlan, $W = \{w_1, w_2\}$, a domeni i u jednom i u drugom svetu su jednaki skupu $\{d_1, d_2\}$,
- relacija dostižnosti je puna relacija, tj. $R = W^2$,
- konstanta c se u svetu w_1 interpretira kao d_2, a u svetu w_2 kao d_1 i
- u svetu w_1 relacijski simbol P se interpretira kao relacija $\{d_1\}$, a u svetu w_2 kao relacija $\{d_2\}$.

Sada u svetu w_1 važi formula $(\forall x)\Diamond P(x)$, jer kakvu god vrednost dodelili promenljivoj x formula $\Diamond P(v(x)/x)$ je tačna u w_1. Sa druge strane, formula $\Diamond P(c)$ nije tačna u w_1 jer $P(c)$ ne važi ni u w_1, ni u w_2, pa $w_1 \not\models (\forall x)\Diamond P(x) \to \Diamond P(c)$. Ξ

U najrestriktivnijem slučaju modela sa konstantnim domenima i rigidnim termima aksiomatizacija se ostvaruje dodavanjem na aksiomatski sistem klasične logike prvog reda modalnih aksioma i pravila izvođenja iskaznih logika i Barkan-formule

$$(\forall x)\Box\alpha \rightarrow \Box(\forall x)\alpha$$

koja karakteriše konstantnost domena.

Još jednu zanimljivost modalnih logika prvog reda predstavlja i činjenica da, za razliku od klasične predikatske logike, unarna modalna predikatska logika prvog reda nije odlučiva.

7.1.7 Temporalne logike

Temporalne, tj. vremenske, logike su modalne logike koje omogućavaju da se rezonuje o promeni istinitosti nekog iskaza tokom vremena što je jedna od karakteristika prirodnog jezika. Da bi se to postiglo klasični jezik se proširuje modalnim operatorima koji u vremenu karakterišu istinitosnu vrednost formula na koje se primenjuju. Po pravilu, koriste se dva oblika operatora $\Box$ koji se označavaju sa G i H a tumače se kao *uvek u budućnosti* i *uvek u prošlosti*. Pored njih se upotrebljavaju i operatori $F = \neg G\neg$ *biće u budućnosti* i $P = \neg H\neg$ *bilo je u prošlosti*. Jedna tipična formula temporalnih logika je:

$$Pp \wedge PH\neg p \rightarrow P(GPp \wedge H\neg p)$$

koja se čita kao 'ako je u nekom momentu u prošlosti važilo p i ako je u nekom momentu u prošlosti važilo da je uvek pre tog momenta važilo $\neg p$, onda je u nekom momentu u prošlosti važilo da je za svaki momenat u budućnosti postojao momenat koji mu je prethodio i u kome je važilo p i pre koga je uvek važilo $\neg p$'.

Modeli za temporalne logike su Kripkeovi modeli oblika $\langle W, R, v\rangle$ u kojima je W skup vremenskih trenutaka, a R se tumači kao relacija 'biti ranije' ili u nekim pristupima kao 'ne biti kasnije'[9]. Relacija zadovoljivosti se definiše kao i u kod modalnih logika, recimo:

- $w \models G\alpha$ ako i samo ako za svaki svet u za koji je wRu važi $u \models \alpha$,
- $w \models F\alpha$ ako i samo ako za postoji svet u za koji je wRu i važi $u \models \alpha$,
- $w \models H\alpha$ ako i samo ako za svaki svet u za koji je uRw važi $u \models \alpha$ i
- $w \models P\alpha$ ako i samo ako za postoji svet u za koji je uRw i važi $u \models \alpha$.

Klase temoralnih modela se razlikuju u izboru strukture skupa vremenskih trenutaka koja može biti izomorfna skupu prirodnih, celih, racionalnih, realnih brojeva, pozitivnih realnih brojeva, kružna itd., zavisno od oblasti rezonovanja koja se formalizuje. Svakom od ograničenja odgovaraju karakteristične aksiome. Recimo, formuli

$$G\alpha \rightarrow F\alpha$$

[9] U prvom slučaju relacija nije refleksivna, a u drugom to jeste.

odgovara zahtev da ne postoji poslednji vremenski trenutak.

U računarstvu se najviše koriste temporalne logike u kojima postoji početni trenutak, a vreme je diskretno i linearno. U tom slučaju se pre svega razmatraju operatori koji govore o budućnosti. Pored već spomenutih, koriste se i operatori $\bigcirc$ i U koji se tumače kao 'u sledećem trenutku', odnosno 'sve dok nije'[10]. Ako su vremenski trenuci označeni nizom w_0, w_1, ..., onda važi:

- $w_i \models \bigcirc\alpha$ ako i samo ako $w_{i+1} \models \alpha$ i
- $w_i \models \beta U \alpha$ ako i samo ako postoji $k \geq i$ tako da $w_k \models \alpha$ i za svako j za koje je $i \leq j < k$, $w_j \models \beta$.

Operatorom U je moguće definisati operatore F i G:

- $F\alpha \leftrightarrow (\top U \alpha)$, odnosno
- $G\alpha \leftrightarrow \neg(\top U \neg\alpha)$.

U slučaju da se u obzir uzima i prošlost, koristi se i operator S koji se tumači kao 'od kada je'[11] i za koga je

- $w_i \models \beta S \alpha$ ako i samo ako postoji $k \leq i$ tako da $w_k \models \alpha$ i za svako j za koje je $k < j \leq i$, $w_j \models \beta$.

Ova vrsta temporalnih logika se primenjuje u specifikaciji i verifikaciji sistema. Specifikacija predstavlja opis željenog ponašanja sistema, dok se verifikacijom ustanovljava da sistem ispoljava određene osobine. Linearna diskretna vremenska logika je pogodna za opisivanje skupa svih izvršavanja nekog programa i proučavanje osobina programa koje su univerzalne, tj. uniformno važe pri svim izvršavanjima. Prikaz jednog postupka verifikacije opširnije je dat u odeljku 9.8.1.

Primer 7.1.12 Formule koje se u verifikaciji često pojavljuju su:

- formula $\alpha \rightarrow F\beta$ znači da će, ako je α zadovoljena u nekom momentu, formula β eventualno biti zadovoljena nakon tog momenta,
- formulom $G(\alpha \rightarrow F\beta)$ se pojačava prethodni iskaz, tj. kaže se da prethodna formula važi u svakom svetu, pa će posle svakog trenutka u kome važi α postojati trenutak u kome će važiti β,
- formula $GF\alpha$ znači da će α važiti u beskonačno mnogo trenutaka,
- formula $FG\alpha$ znači da će α eventualno, u nekom budućem momentu, početi da zauvek važi,
- formula $G(\alpha \rightarrow G\alpha)$ znači da ako α bude važila u nekom trenutku, onda će nakon njega važiti zauvek,
- formula $\neg\beta U\alpha$ znači da β neće važiti u bilo kom momentu koji prethodi prvom momentu u kome važi α itd.

[10] Until.
[11] Since.

Na primer, ako formulu α shvatimo kao zahtev za resursom, a formulu β kao realizaciju zahteva, onda formula $G(\alpha \to F\beta)$ kaže da će svaki zahtev za resursom biti opslužen, što je jedno od bitnih svojstava svakog operativnog sistema. Ξ

Jedna od zanimljivih struktura vremena koja se takođe primenjuje u računarstvu se dobija pretpostavkom da svaki vremenski trenutak ima linearno uređenu prošlost, dok se budućnost razgranava, odnosno da se vreme može prikazati u obliku drveta. I ovde je prevashodno interesantan slučaj u kome je vreme diskretno, mada postoje i drugačije logike. Drvo kojim se predstavlja vreme, ako se pretpostavi da ne postoje početni i završni trenuci, nema ni koren ni listove. Kod ovakvih logika posebno se proučava pitanje značenja operatora F, odnosno da li je $w \models F\alpha$ ako α važi:

- u bar jednom trenutku na svakoj grani kojoj je w koren ili
- u bar jednom trenutku na bar jednoj grani čiji je koren w.

Jedan od načina da se prevaziđe ova dilema je upotreba nešto drugačijih vremenskih operatora ($\forall G$, $\exists G$, $\forall F$, $\exists F$, $\forall \bigcirc$ i $\exists \bigcirc$) prilagođenih drvoidnoj strukturi vremena. Njihovo značenje se definiše sa:

- $w \models \forall G\alpha$ ako α važi u svim vremenskim trenucima u poddrvetu čiji je koren trenutak w,
- $w \models \exists G\alpha$ ako α važi u svim vremenskim trenucima u bar jednoj grani koja počinje trenutkom w,
- $w \models \forall F\alpha$ ako na svakoj grani koja počinje trenutkom w postoji bar jedan trenutak u kome važi α,
- $w \models \exists F\alpha$ ako postoji bar jedna grana koja počinje trenutkom w na kojoj u bar jednom trenutku važi α,
- $w \models \forall \bigcirc \alpha$ ako α važi u svakom sledećem trenutku vremenskog trenutka w i
- $w \models \exists \bigcirc \alpha$ ako α važi u bar jednom sledećem trenutku vremenskog trenutka w.

Lako se vidi da važi $\exists F = \neg\forall G\neg$, $\forall F = \neg\exists G\neg$ i $\exists\bigcirc = \neg\forall\bigcirc\neg$.

Razgranata (drvoidna) diskretna vremenska logika je pogodna za opisivanju drveta izvršavanja nekog nedeterminističkog programa. Pri tome se analiziraju egzistencijalne osobine, poput korektnog završetka rada bar jednog od mogućih izvršavanja programa za proizvoljan ulaz.

Primer 7.1.13 Neka su P nedeterministički program i B i H redom iskazi koji znače da je program u polaznom stanju, odnosno u završnom stanju. Neka su I i O iskazi koji opisuju korektan ulaz i izlaz programa. Pretpostavlja se da program, ako se zaustavi, beskonačno ostaje u tom završnom stanju. Sledeće formule opisuju različite koncepte korektnosti programa.

Formula $(B \wedge I) \to \exists G(H \to O)$ znači da, ako je program počeo izvršavanje nad korektno datim ulaznim podacima, onda postoji izvršavanje koje može biti beskonačno, ali ako je konačno, onda program završava rad dajući korektan izlaz.

Formula $(B \wedge I) \rightarrow \exists F(H \wedge O)$ znači da, ako je program počeo izvršavanje nad korektno datim ulaznim podacima, onda postoji izvršavanje koje završava rad dajući korektan izlaz.

Formula $(B \wedge I) \rightarrow \forall G(H \rightarrow O)$ znači da, ako je program počeo izvršavanje nad korektno datim ulaznim podacima, onda svako konačno izvršavanje daje korektan izlaz.

Formula $(B \wedge I) \rightarrow \forall F(H \wedge O)$ znači da, ako je program počeo izvršavanje nad korektnim ulaznim podacima, onda je svako izvršavanje konačno i korektno. Ξ

Pored primena u specifikaciji i verifikaciji, temporalne logike se koriste i u drugim oblastima u kojima je prisutan vremenski aspekt u zaključivanju. Na primer, u dijagnostici u medicini značajno je opisati istoriju bolesti. Takođe, u planiranju akcija koje treba da se izvrše u okolini koja trpi promene bitno je upravljati znanjem koje se menja u vremenu itd.

Sve spominjane iskazne temporalne logike su aksiomatizibilne, pri čemu se klasična iskazna aksiomatizacija proširuje karakterističnim aksiomama za vremenske operatore. Ove logike su takođe i odlučive.

Pregled temporalnih logika zaključimo primedbom da je prikazan pristup sa tačkastom interpretacijom vremena u kome su osnovni elementi vremena trenuci. Postoji i druga vrsta logika u kojima su osnovni elementi vremena intervali.

7.1.8 Modalne logike znanja i/ili verovanja

Jedna od spominjanih interpretacija modalnog operatora $\Box$ je *znati*, u kom slučaju je uobičajeno da se umesto $\Box$ koristi znak[12] K. Tada se modalne aksiome

(**T**) $K\alpha \rightarrow \alpha$,

(**4**) $K\alpha \rightarrow KK\alpha$ i

(**5**)[13] $\neg K\alpha \rightarrow K\neg K\alpha$

redom interpretiraju kao 'ako znam α, onda je α tačno', 'ako znam α, onda znam da znam α' i 'ako ne znam α, onda znam da ne znam α'. Pravilo necesitacije se interpretira sa 'zna se sve što je tačno'. Aksiome (**4**) i (**5**) se nazivaju i aksiome introspekcije, odnosno samospoznaje, jer garantuju da se poznaje kompletno sopstveno znanje. Aksioma (**T**) i pravilo (**N**) obezbeđuju konzistentnost i kompletnost znanja jer ne dozvoljavaju da se zna nešto što nije tačno, odnosno garanutuju da se sve što je tačno zna. Logike znanja se primenjuju u ekonomiji, pri donošenju odluka i sklapanju pogodbi, u računarskim naukama, pri analizi protokola u distribuiranim sistemima, u veštačkoj inteligenciji itd.

Često je potrebno istovremeno razmatrati znanje više agenata[14] pa se u u modalnom jeziku javlja lista unarnih modalnih operatora $K_1, \ldots, K_n$ sa značenjem 'agent i zna'. Tako je moguće govoriti ne samo o znanju jednog agenta, nego i o znanju agenta o znanju nekog drugog agenta itd. Nije uobičajeno da se u

[12] K potiče od engleskog termina *knowledge*, znanje.

[13] Ovo je samo drugačiji zapis formule $\Diamond\alpha \rightarrow \Box\Diamond\alpha$, pri čemu se koristi definicija $\Diamond = \neg\Box\neg$.

[14] Agent može biti proces koji se izvrašava na računarskom sistemu, formalni model za simulaciju neke osobe, robot ili neki drugi elektronski uređaj, program na Internetu itd.

jezik uključuju pandani operatora $\Diamond$, mada se $\neg K_i \neg \alpha$ shvata kao 'moguće je α'. Značenje modalnim formulama se daje uopštenjem Kripkeovih modela u kojima se javlja više relacija dostižnosti, po jedna za svaki operator K_i. U modelu $M = \langle W, \mathcal{K}_1, \ldots, \mathcal{K}_n, v \rangle$ za svet $w \in W$, važi

- $w \models K_i \alpha$ ako i samo ako za svaki svet u za koji je $w \mathcal{K}_i u$ važi $u \models \alpha$.

Pri tome je pogodno svetove modela shvatati kao stanja aktuelnog sveta. Ako je $w \mathcal{K}_i u$, agent i ne može razlikovati stanja w i u i, ako je w aktuelno stanje sveta, agent i će i stanje u smatrati mogućim.

Uglavnom se prihvata da su relacije $\mathcal{K}_i$ relacije ekvivalencije, pa su modeli za logiku znanja uopštenja S5-modela, dok se odgovarajuća kompletna aksiomatizacija dobija zamenom operatora $\Box$ u aksiomama (**K**), (**T**), (**4**) i (**5**) i u pravilu (**N**) sa K_i, za $i = 1, n$.

Pored operatora K_i u logici znanja se upotrebljavaju i operatori E_G i C_G koji se tumače kao 'svi u grupi G znaju'[15], odnosno 'u grupi G opšte je poznato'[16] gde se to sveopšte shvata u smislu 'svi znaju', 'svi znaju da svi znaju', 'svi znaju da svi znaju da svi znaju da svi znaju da itd.'. Formalno,

- $w \models E_G \alpha$, $G \subset \{1, \ldots, n\}$ ako i samo ako za svaki $i \in G$, $w \models K_i \alpha$ i
- $w \models C_G \alpha$, $G \subset \{1, \ldots, n\}$ ako i samo ako za svaki $k = 1, 2, \ldots$, $w \models E_G^k \alpha$, gde je $E_G^1 \alpha =_{def} E_G \alpha$, $E_G^{k+1} \alpha =_{def} E_G E_G^k \alpha$.

Ovi operatori povećavaju izražajnost logika, ali i značajno podižu njihovu složenost.

Sledeći primeri ilustruju primene modalne logike znanja.

Primer 7.1.14 Osobe 1 i 2 sede jedna naspram drugoj i imaju kape na glavama. Osoba 1 vidi kapu osobe 2 i obrnuto, ali ni jedna osoba ne vidi svoju kapu. Iskaz p_i, $i = 1, 2$, označava da je na kapi osobe i zalepljena nalepnica. Ispitivač na početku javno obaveštava osobe da se na nečijoj kapi nalazi nalepnica, što zapisujemo u formi $K_1(p_1 \vee p_2)$, odnosno $K_2(p_1 \vee p_2)$. Pošto je saopštenje javno, svaka osoba zna da i druga osoba zna $p_1 \vee p_2$, što zapisujemo sa $C_{\{1,2\}}(p_1 \vee p_2)$. Svaka osoba zna i da ona druga osoba zna da li je nalepnica na kapi prve osobe, tj. $K_1(K_2 p_1 \vee K_2 \neg p_1)$ i $K_2(K_1 p_2 \vee K_1 \neg p_2)$. Ispitivač postavlja pitanje da li neka osoba zna da li je nalepnica na njenoj kapi. Pošto ni jedna osoba ne zna odgovor, osoba 1 zna da osoba 2 ne zna da li na svojoj kapi ima nalepnicu i obrnuto, tj. $K_1 \neg K_2 p_2$ i $K_2 \neg K_1 p_1$. Razmatrajući opis može se pokazati da nakon ovakvog odgovora osobe znaju da su nalepnice zalepljene na obe kape. Koristeći prethodno razvijenu sintaksnu i semantičku karakterizaciju znanja, neki dokazivač teorema bi mogao formalno izvesti da je $\vdash C \to K_1 p_1 \wedge K_1 p_2$, i $\vdash C \to K_2 p_1 \wedge K_2 p_2$, gde je C konjunkcija formula iz postavke zadatka.

Isti zaključak se može izvesti i analizom modela odgovarajućih stanja sveta u odnosu na označavanje kapa nalepnicama. Najpre, stanja sveta predstavimo uređenim parovima oblika (x_1, x_2), gde je $x_i = 1$ ako se na kapi osobe i nalazi nalepnica, inače je $x_i = 0$. Očigledno, postoje četiri moguća stanja predstavljena sa $(0, 0)$, $(0, 1)$, $(1, 0)$ i $(1, 1)$. Svakoj od osoba odgovara relacija dostižnosti koja

[15] E potiče od engleskog termina *everyone*, svako.
[16] C potiče od engleskog termina *common*, opšte, zajedničko.

prikazuje koja stanja osoba ne može razlikovati s obzirom da ne vidi sopstvenu kapu. Zato je relacija $\mathcal{K}_1$ najmanja relacija ekvivalencije koja sadrži $\{((0,1),(1,1)),((0,0),(1,0))\}$, a $\mathcal{K}_2$ najmanja relacija ekvivalencije koja sadrži $\{((0,0),(0,1)),((1,0),(1,1))\}$. Iskazi p_i definisani kao u prethodnom pasusu u nekom svetu važe ako i samo ako je $x_i = 1$. Na ovaj način je određen jedan Kripkeov model u kome, na primer, važi:

- $(0,1) \models \neg K_2 p_2$, jer $(0,0) \models \neg p_2$, a $((0,1),(0,0)) \in \mathcal{K}_2$,
- $(1,0) \models \neg K_1 p_1$, jer $(0,0) \models \neg p_1$, a $((1,0),(0,0)) \in \mathcal{K}_1$ i
- $(0,0) \models \neg(p_1 \vee p_2)$.

Nakon što ispitivač saopšti da se na nečijoj kapi nalazi nalepnica, to postaje opšte znanje grupe, tj. važi $C_{\{1,2\}}(p_1 \vee p_2)$. Promena opšteg znanja grupe $\{1,2\}$ se u modelu reflektuje tako što se odstrani stanje označeno sa $(0,0)$ jer u njemu važi $\neg(p_1 \vee p_2)$. Odbacivanje sveta $(0,0)$ dovodi do promena u relacijama $\mathcal{K}_1$ i $\mathcal{K}_2$, tako da u novom modelu važi:

- $(0,1) \models K_2 p_2$, jer $(0,1) \models p_2$, a u novoj relaciji $\mathcal{K}_2$ svet $(0,1)$ je u relaciji samo sa samim sobom, i slično
- $(1,0) \models K_1 p_1$, jer je $(1,0) \models p_1$.

Odgovor 'ne znam' osoba na pitanje ispitivača da li se nalepnica nalazi na njihovoj kapi dalje menja opšte znanje grupe i izgled modela. Ne primer, takav odgovor osobe 1 eliminiše stanje $(1,0)$, jer $(1,0) \models K_1 p_1$. Slično se eliminiše i stanje $(0,1)$. U jedinom preostalom stanju $(1,1)$ važi $K_1 p_1$ i $K_2 p_2$. Ξ

Primer 7.1.15 Neka je dat sistem koji se sastoji od n agenata, recimo računarska mreža u kojoj se izvršava n procesa koji međusobno komuniciraju. Svaki od agenata se nalazi u nekom lokalnom stanju u svakom vremenskom trenutku. Lokalno stanje opisuje informacije kojima agent ima pristup, poput inicijalnog stanja, liste poruka koje je agent razmenio itd. Globalno stanje je uređena n-torka $(s_1, \ldots, s_n)$ lokalnih stanja agenata u nekom trenutku. Jedno izvršavanje r sistema se sastoji od niza globalnih stanja kroz koja sistem prolazi. Tačka je uređeni par (r,m) koji se sastoji od izvršavanja r i vremenskog trenutka m. U tački (r,m) sistem se nalazi u globalnom stanju $r(m) = (s_1, \ldots, s_n)$, dok sa $r_i(m)$ označavamo lokalno stanje s_i agenta i u tom trenutku. Opis R celog sistema je dat skupom izvršavanja, tj. opisom ponašanja sistema, dok je interpretirani sistem I uređeni par (R,v) koji se sastoji od opisa sistema i valuacije v koja u svakom globalnom stanju svakom iskaznom slovu dodeljuje istinitosnu vrednost. Iskazna slova odgovaraju osnovnim činjenicama o sistemu, kao što su 'sistem je zaglavljen', 'proces i je poslao poruku u k-tom koraku izvršavanja' i sl.

Interpretiranom sistemu $I = (R,v)$ odgovara jedan Kripkeov model $M_I = \langle S, \mathcal{K}_1, \ldots, \mathcal{K}_n, v\rangle$, gde su S skup svih tačaka iz I, v valuacija iz I, a relacije $\mathcal{K}_i$ se definišu tako da $(r,m)\mathcal{K}_i(r',m')$ ako $r_i(m) = r'_i(m')$, tj. ako se poklapaju lokalna stanja agenta i. Lako se vidi da su sve relacije $\mathcal{K}_i$ relacije ekvivalencije. Navešćemo uslove za tačnost u nekoj tački samo za formule koje počinju operatorima G i K_i, dok su preostali slučajevi jednostavna uopštenja već opisanih situacija. Tačnost formula u tački (r,m) interpretiranog sistema I se definiše na sledeći način:

- $(I, r, m) \models G\alpha$ ako i samo ako $(I, r, m') \models \alpha$ za svako $m' \geq m$, gde je G temporalni operator uvek u budućnosti i
- $(I, r, m) \models K_i\alpha$ ako i samo ako za svako (r', m') za koje je $(r, m)\mathcal{K}_i(r', m')$ važi $(I, r', m') \models \alpha$, gde je K_i modalni operator znanja.

Kombinovanjem temporalnih operatora i operatora znanja mogu se praviti iskazi o razvoju znanja u vremenu.

Opisani pristup se koristi u dizajniranju i verifikaciji distribuiranih protokola, tj. izračunavanja u kojima radi više procesa koji međusobno komuniciraju. Primer takvog protokola je koordinirani napad u kome dva procesa A i B treba da se dogovore oko koordiniranog rada na nekom problemu. Ni jedan proces neće započeti obradu pre nego što je siguran da će to, zajedno sa njim, učiniti i drugi proces. Procesi se dogovaraju razmenom poruka koje, ako stignu na cilj, to čine u nekom jediničnom vremenu, ali se može dogoditi i da se zagube tokom prenosa. Pitanje koje se analizira je koliko vremena je procesima potrebno da se dogovore? Jedan scenario događanja je sledeći. Proces A šalje poruku 'započnimo obradu u trenutku t' koju proces B dobija. Proces B ne može započeti obradu, jer proces A ne zna da li je on primio poruku, zbog čega proces A neće početi obradu. Zato proces B šalje poruku sa potvrdom prijema prve poruke. Pretpostavimo da i ona stiže na cilj. Da li će sada proces A započeti obradu? Ne, jer proces B ne zna da li je proces A poruku primio, pa proces B ne zna da proces A zna da je on primio prvu poruku, zbog čega proces A ne zna da li će proces B započeti obradu. Postupak razmene poruka se može nastaviti proizvoljno dugo, ali se pokazuje da ni u jednom momentu procesi A i B neće postići konsenzus oko početka obrade. Zapravo, razmenom poruka procesi povećavaju svoje znanje, ali ne postižu opšte znanje o početku obrade. Ξ

U nekim pristupima se kritikuje aksiomatizacija rezonovanja o znanju kakva je ovde opisana. Aksiome (**4**) i (**5**) se odbacuju odgovarajućim izmenama zahteva za relaciju dostižnosti. Aksioma (**T**) i pravilo (**N**) su u velikoj meri prirodni u pristupu zasnovanom na Kripkeovim modelima, pa se tu ne dovode u pitanje. Međutim, oni karakterišu idealnog mislioca, logičkog sveznalicu, koji zna valjane formule i sve posledice svog znanja i nisu realni ni u slučajevima modeliranja ljudi, niti kada se u obzir uzmu ograničeni resursi izračunavanja.

U logikama verovanja operator $\Box$ se interpretira kao *verovati*. Pošto je verovanje manje precizno od znanja, pojedini sistemi dopuštaju verovanje u neistinito tvrđenje. U logikama verovanja se takođe koriste modalne aksiome, ali se ne uključuje aksioma (T), što se tumači kao: može se verovati i u nešto što nije tačno.

7.1.9 Dinamička logika

Dinamička logika je jedna vrsta modalnih logika čija je prevashodna namena da pruži prirodan formalizam za proučavanje ponašanja programa.

Skup svetova modela ovde predstavlja skup stanja u kojima se program može naći tokom izvršavanja. Stanja programa opisujemo vrednostima promenljivih. Recimo, u programu za izračunavanje vrednosti n-tog člana Fibonačijevog niza

$$(x, y, z) \leftarrow (n, 1, 1)$$

Naredba	Zapis u dinamičkoj logici
if p then α else β	$(p?;\alpha) \cup (\neg p?\beta)$
while p do α	$(p?;\alpha)^*;\neg p?$
repeat α until p	$\alpha(\neg p?;\alpha)^*;p?$

Tabela 7.3. Zapis standardnih programskig konstrukcija.

while $x \neq 0$ **do** $(x, y, z) \leftarrow (x - 1, z, y + z)$

značajne su tri promenljive x, y i z koje redom sadrže brojač ciklusa, vrednost prethodnog i vrednost tekućeg člana niza. Pretpostavimo da je α oznaka nekog nedeterminističkog programa. Tada su dva stanja s i t u relaciji R_α, tj. $(s,t) \in R_\alpha$, ako je t moguće završno stanje programa α koji je izvršavanje započeo u stanju s. Za svaki program α razmatraju se operatori $[\alpha]$ i $\langle\alpha\rangle$ kao varijante operatora $\Box$, odnosno $\Diamond$.

Primer 7.1.16 Primeri formula koje se ovde proučavaju su:

- $n \geq 0 \rightarrow [\alpha](z = F_n)$ koja je tačna u nekom stanju s, u kom je $n \geq 0$, ako je u svakom završnom stanju programa (koja ne moraju postojati) vrednost promenljive z jednaka n-tom članu Fibonačijevog niza,
- $n \geq 0 \rightarrow \langle\alpha\rangle(z = F_n)$ koja je tačna u nekom stanju s, u kom je $n \geq 0$, ako postoji završno stanje programa u kom je vrednost promenljive z jednaka n-tom, članu Fibonačijevog niza ili
- $\langle\alpha\rangle p \leftrightarrow \langle\beta\rangle p$ kojom tvrdimo su programi α i β ekvivalentni u odnosu na p u smislu da za svako početno stanje s postoji stanje t u kom program α završava rad i važi p ako i samo ako za s postoji i stanje u u kom rad završava program β i važi p.

U dinamičkoj logici je dozvoljena manipulacija složenim programima kojom svaki program dobija posebne modalne operatore. Program $\alpha \cup \beta$ predstavlja program u kome se nedeterministički izabere i potom izvrši bilo program α, bilo program β. Relacija dostižnosti je sada $R_{\alpha\cup\beta} = R_\alpha \cup R_\beta$.

Primer 7.1.17 U dinamičkoj logici izrazive su i formule:

- $\langle\alpha \cup \beta\rangle p \leftrightarrow (\langle\alpha\rangle p \vee \langle\beta\rangle p)$ i
- $[\textbf{while}(p \vee q)\ \textbf{do}\ \alpha]r \leftrightarrow$
  $[(\textbf{while}(p)\ \textbf{do}\ \alpha)(\textbf{while}(q)\ \textbf{do}\ (\alpha\ \textbf{while}(p)\ \textbf{do}\ \alpha)))]r$ Ξ

Pored unije dva programa, koriste se i nadovezivanje, u oznaci $\alpha;\beta$, ponavljanje programa nula ili više puta, u oznaci α^*, i uslovno izvršavanje, u oznaci $p?$. Opis standardnih programskih konstrukcija ovakvim sredstvima je prikazan u tabeli 7.3.

Značenje formula se, kao što je već nagovešteno, daje uopštenjem Kripkeovih modela u kome za svaki elementarni program P postoji posebna relacija dostižnosti. Pri tome za složene programe važi:

- $R_{\alpha;\beta} = \{(s,t) : (\exists u)((s,u) \in R_\alpha \wedge (u,t) \in R_\beta)\}$,
- $R_{\alpha^*} = \{(s,t) : (\exists s_0, s_1, \ldots, s_k)(s_0 = s \ \wedge \ s_k = t \ \wedge \ (\forall i)((0 \leq i < k) \rightarrow (s_i, s_{i+1}) \in R_\alpha))\}$ i
- $R_{p?} = \{(s,s) : s \models p\}$.

Ako su formule koje se javljaju iskazne, dobija se iskazna dinamička logika koja je aksiomatizibilna i odlučiva. Takođe se proučava i dinamička predikatska logika u kojoj je, kao u primerima 7.1.16 i 7.1.18, dozvoljena upotreba promenljivih, funkcijskih i relacijskih simbola.

Primer 7.1.18 Formula:

$$(\forall x)\langle \textbf{while}(x \neq 1) \textbf{ do if}(parno(x)) \textbf{ then } x \leftarrow \tfrac{x}{2} \textbf{ else } x \leftarrow 3x+1\rangle \textbf{ true}$$

tvrdi da uzastopno množenje neparnog broja sa 3 i uvećanje za 1 i polovljenje parnog broja, polazeći od bilo kog prirodnog broja redukuje taj broj na 1. Istinitost ove formule nije poznata. Ξ

7.1.10 Provera modela

Ispitivanje da li je neka formula posledica skupa formula očigledno modelira ljudsko zaključivanje, a može se primeniti u veštačkoj inteligenciji pri donošenju odluka, u proveri korektnosti programa i/ili elektronskih kola itd. Ovaj postupak bi mogao biti sproveden tako što bi se nekom dokazivaču teorema postavilo pitanje u formi 'da li je valjana formula $KB \rightarrow \alpha$?', gde je KB formula koja opisuje bazu znanja, tj. trenutno raspoložive podatke, dok je α formula koja predstavlja sam upit. Problem u tom pristupu predstavlja često velika složenost postupka dokazivanja o čemu ćemo detaljno govoriti u glavi 10. Kao odgovor na ovaj problem razvija se drugačiji pristup, takozvana provera modela[17] kojom se neke situacije efikasnije razrešavaju.

Ako se za zaključivanje u kome se koriste dokazivači teorema može reći da je sintaksnog karaktera, onda je provera modela zasnovana na semantičkom pristupu. Naime, umesto da se traži dokaz za α iz KB, u proveri modela se, kao što i sam naziv postupka sugeriše, proverava da li formula α važi u modelu koji određuje baza znanja. Provera modela se primenjuje na probleme koji se mogu opisati Kripkeovim modelima. Tipični problemi sa kojima se provera modela sreće su oni u kojima je kontrolni aspekt, a ne obrada složenih podataka, u prvom planu, kao što je slučaj sa verifikacijom elektronskih kola, komunikacijskim protokolima, sistemima za kontrolu procesa itd. Osobine sistema koje se najčešće proveravaju dele se na[18]:

- osobine dostižnosti kojima se iskazuje da se neka situacija može ostvariti, na primer da će vrednost promenljive biti manja od nule,

[17] Model checking.

[18] Engleski nazivi za navedene osobine su redom reachability property, safety property, liveness property i fairness property.

- osobine sigurnosti kojima se iskazuje da se, pod određenim uslovima, nešto neće nikada dogoditi, na primer da proces A neće započeti rad ako se proces B ne izvrši,
- osobine dogodivosti kojima se iskazuje da se, pod određenim uslovima, nešto ostvaruju pre ili posle, pri čemu se ne zna tačno kada, na primer da će, ako se uputi zahtev za resursom, on biti zadovoljen, i
- osobine pravičnosti kojima se iskazuje da se, pod određenim uslovima, nešto ostvaruju (ili ne ostvaruje) beskonačno često, na primer da će proces neprestano zahtevati neki resurs.

Ove osobine se pogodno opisuju sredstvima temporalne logike.

Primer 7.1.19 Osobina sigurnosti kojom se iskazuje da završetak rada procesa B mora da prethodi početku rada procesa A može se jezikom linearne temporalne logike zapisati kao $(\neg\text{start}(A))U$ kraj(B). Slično, eventualna osobina da će upućeni zahtev za resursom uvek biti zadovoljen može se zapisati sa $G(\text{zahtev} \rightarrow F \text{ ispunjen})$. Ξ

Provera modela se u osnovi sastoji iz sledećih koraka:

- modeliranje sistema Kripkeovim modelom, odnosno konačnim automatom (o čemu će biti reči u poglavlju 9),
- zapisivanje osobine koja se proverava jezikom temporalne logike i
- ispitivanje da li model zadovoljava tu formulu,

za koje postoje više načina realizacije.

Jedna kombinacija tih varijanti ilustrovana je u analizi Kripkeovog modela u primeru 7.1.14, dok ćemo drugi vid provere modela opisati u odeljku 9.8.1.

7.2 Metoda prefiksiranih tabloa za modalne logike

Kao što je ranije već rečeno, metoda tabloa za klasičnu logiku se uz izvesne modifikacije može primeniti i na modalne logike. Načina da se to izvede ima više, a ovde ćemo prikazati metodu *prefiksiranih tabloa*.

7.2.1 Proširenje ujednačavajuć notacije

U podeli formula koja je činila ujednačavajuću notaciju potrebno je dodati formule koje se odnose na modalne operatore. Novi slučajevi su prikazani u tabeli 7.4.

Sledeća razlika u odnosu na klasičnu logiku se ogleda u potrebi da se označe svetovi modela pošto nije svejedno u kom svetu se javlja koja formula. To se sprovodi pomoću prefiksa koji predstavljaju imena svetova.

Definicija 7.2.1 *Prefiks* je konačan niz prirodnih brojeva. Prefiks σ' je *prosto proširenje* prefiksa σ ako je $\sigma' = \sigma, n$, za neki prirodan broj n. Ako je σ prefiks i X označena modalna formula, tada je $\sigma\ X$ *prefiksirana označena modalna formula*.

Primer 7.2.2 $\sigma = 1, 2, 2, 1$ je jedan prefiks, dok su $\sigma\ T\ A$ i $\sigma\ F\ A$ prefiksirane označene modalne formule. Ξ

ν	ν_0
$T\ \Box A$	$T\ A$
$F\ \Diamond A$	$F\ A$

π	π_0
$T\ \Diamond A$	$T\ A$
$F\ \Box A$	$F\ A$

Tabela 7.4. Modalne formule u ujednačavajućoj notaciji.

Naziv klase modela	Uslovi za relaciju dostižnosti
K	opšti uslov
T	opšti uslov, refleksivnost
$K4$	opšti uslov, tranzitivnost
KB	opšti uslov, inverznost
$S4$	opšti uslov, refleksivnost, tranzitivnost
B	opšti uslov, refleksivnost, inverznost
$S5$	potpuna relacija
D	opšti uslov
$D4$	opšti uslov, tranzitivnost
DB	opšti uslov, inverznost

Tabela 7.5. Uslovi za relaciju dostižnosti između prefiksa.

7.2.2 Pravila za konstrukciju prefiksiranih tabloa

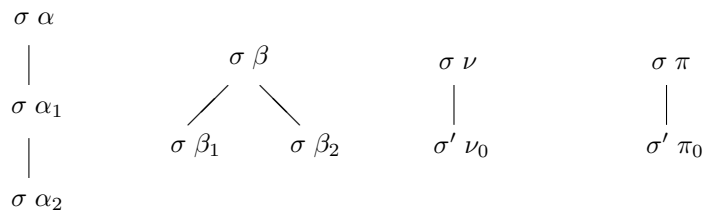
Slično kao i u klasičnom iskaznom slučaju, izgradnja tabloa započinje postavljanjem formule $1\ F\ A$ u koren drveta, nakon čega se primenjuju pravila konstrukcije. Grana tabloa je *zatvorena* ako se na njoj nađu formule oblika $\sigma\ F\ A$ i $\sigma\ T\ A$. Tablo je *zatvoren* ako mu je zatvorena svaka grana. Zatvoreni tablo za formulu $1\ F\ A$ je *dokaz* za formulu A.

Prefiks σ se *koristi na grani* tabloa ako se na grani nalazi bar jedna prefiksirana formula oblika σX, za neku označenu formulu X. Prefiks σ je *bez restrikcija* na nekoj grani tabloa ako nije početni deo bilo kog prefiksa na toj grani.

Između prefiksa definisaćemo binarnu relaciju dostižnosti. Ova relacija zadovoljava:

- opšti uslov, ako je σ, n dostižno iz σ, za svaki prirodan broj n i svaki prefiks σ,
- refleksivnost, ako je σ dostižno iz σ za svaki prefiks σ,
- inverznost, ako je σ dostižno iz σ, n za svaki svaki prirodan broj n i svaki prefiks σ i
- tranzitivnost, ako je σ, σ' dostižno iz σ za svaki konačan neprazan niz prirodnih brojeva σ' i svaki prefiks σ.

U tabeli 7.5 je dat pregled svojstava koje zadovoljava relacija dostižnosti između prefiksa za spominjane modalne logike.



Slika 7.1. α, β, ν i π-pravila.

Pravila za konstrukciju tabloa se uvode na sledeći način. Neka je čvor Y kraj neke grane do tog momenta konstruisanog drveta. Zavisno od formula na grani koja sadrži Y, nastavak konstrukcije je moguće izvesti nekim od pravila:

- α-pravilo: ako je neka $\sigma\ \alpha$ formula na grani, tada se grana produžava sa dva nova čvora od kojih jedan sadrži $\sigma\ \alpha_1$ formulu, a drugi $\sigma\ \alpha_2$ formulu,
- β-pravilo: ako je neka $\sigma\ \beta$ formula na grani, tada se grana u čvoru Y grana, pri čemu levi naslednik sadrži $\sigma\ \beta_1$, a desni $\sigma\ \beta_2$ formulu.
- ν-pravilo: ako je neka $\sigma\ \nu$ formula na grani, tada se grana produžava čvorom $\sigma'\ \nu_0$, gde je σ' dostižno iz σ i dodatno zadovoljava:
  - za K, KB i $K4$, σ' je već korišten na grani i
  - za D, T, DB, $D4$, $S4$ i $S5$, σ' je korišten na grani ili je prosto proširenje prefiksa σ i
- π-pravilo: ako je neka $\sigma\ \pi$ formula na grani, tada se grana produžava čvorom $\sigma'\ \pi_0$, gde je σ' prosto proširenje bez restrikcija prefiksa σ.

Pravila za konstrukciju tabloa su šematski prikazana na slici 7.1. Razlika u primeni ν-pravila je posledica činjenice da kod logika K, $K4$ i KB za proizvoljan svet ne mora postojati svet koji je dostižan iz njega. Uočimo sličnost između ν-pravila i γ-pravila, odnosno π-pravila i δ-pravila, tabloa za klasičnu predikatsku logiku. Prvi par pravila koristi uvedene prefikse, odnosno konstante, dok drugi par uvodi nove prefikse, odnosno konstante. Ovakva paralela nije slučajna, već je posledica činjenice da se u definisanju zadovoljivosti ν-formula koristi univerzalna kvantifikacija mogućih svetova, dok π formulama odgovara egzistencijalna kvantifikacija.

Primer 7.2.3 Razmotrimo T-tablo za formulu $\Box A \to A$ dat na slici 7.2. Prema prethodnim definicijama, u slučaju logike T iz nekog prefiksa su dostižni on sam i sva njegova prosta proširenja. Čvorovi (2) i (3) se dobijaju iz (1) α-pravilom. (4) se dobija iz (2) ν-pravilom (prefiks 1 je korišten na grani). Grana, a tim i tablo se zatvara zbog (3) i (4). Međutim, u logici K ν-pravilo ne bismo mogli primeniti na (2), pa tablo ne bi bio zatvoren. Ξ

(1) $1\ F\ \Box A \to A$
(2) $1\ T\ \Box A$
(3) $1\ F\ A$
(4) $1\ T\ A$
$\times$

Slika 7.2. T-tablo za formulu $\Box A \to A$.

7.2.3 Korektnost i Kompletnost metode prefiksiranih tabloa

Dokaz korektnosti i kompletnosti se sprovodi slično kao i u odeljku 3.8.3.

Teorema 7.2.4 Neka je L neka od normalnih modalnih logika. Formula A ima L-tablo dokaz ako i samo ako je L-valjana.

Dokaz. ($\Rightarrow$) Deo koji se odnosi na korektnost se pokazuje definisanjem preslikavanja I koje prefikse tabloa preslikava u svetove modalnog modela pri čemu, ako je prefiks σ' dostižan iz prefiksa σ, onda je u modalnom modelu $I(\sigma')$ svet dostižan iz sveta $I(\sigma)$, pod uslovom da $I(\sigma)$ uopšte ima dostižan svet. Dalje važi da, ako su sve formule na jednoj grani tabloa zadovoljene pri preslikavanju I, tj. za svaku formulu $\sigma\ X$ koja se javlja na grani, $I(\sigma) \models X$, onda to isto važi i nakon primene bilo kog pravila konstrukcije tabloa. Recimo, razmotrimo logiku T i slučaj ν-pravila. Neka je $\langle W, R, v\rangle$ jedan T-model, $w \in W$, $\sigma\ T\ \Box A$ na grani tabloa i važi $w = I(\sigma)$, $w \models \Box A$. Primena ν-pravila dodaje novu formulu $\sigma'\ T\ A$ grani tabloa, pri čemu postoje dve mogućnosti. Pri prvoj od njih σ' je korišteno na grani i dostižno iz σ, pa je $I(\sigma')$ definisano i važi $I(\sigma)RI(\sigma')$. Prema definiciji relacije $\models$, direktno sledi $I(\sigma') \models A$. Pri drugoj mogućnosti je $\sigma' = \sigma, n$ je prosto proširenje prefiksa σ. Preslikavanje I dodefinišemo tako da je $I(\sigma')$ bilo koji svet modela dostižan iz $I(\sigma)$. Tada ponovo važi $I(\sigma') \models A$. Slično se tvrđenje dokazuje i za preostale vrste formula.

Konačno, recimo da je tablo zadovoljiv ako je je pri nekom preslikavanju I zadovoljiva bar jedna njegova grana. Prethodni dokazom se zapravo pokazuje da je naslednik u konstrukciji jednog zadovoljivog tabloa, takođe zadovoljiv. Prema tome, ako formula ima zatvoreni tablo, to znači da ni on ni bilo koji njegov prethodnik, pa ni tablo $1\ F\ A$, nije zadovoljiv, odakle je formula $F\ A$ nezadovoljiva, a formula A valjana.
($\Leftarrow$) U dokazu kompletnosti potrebno je uvesti sistematsku proceduru izgradnje tabloa koja će garantovati da će biti obrađeni svi potrebni čvorovi. Slično predikatskom tablou za klasičnu logiku, pored dosadašnjih prvila uvode se zahtevi:

- obrađuje se uvek najlevlji čvor najbliži korenu tabloa,
- nakon primene pravila čvor je završen,
- prilikom primene ν-pravila na kraj grane se pored formule $\sigma'\ \nu_0$ dopisuje i formula $\sigma\ \nu$; pravilo se u slučaju logika K, T, B, $S4$ i $S5$ primenjuje na sve, ukoliko takvih ima, prefikse dostižne iz σ; u slučaju logika D, $D4$ i DB pravilo se primenjuje na sve, ukoliko takvih ima, prefikse dostižne iz σ, ako dostižnih prefiksa nema, σ' je novi prefiks bez restrikcije i $\sigma' = \sigma, n$ i

- prilikom uvođenja novog prefiksa dostižnog iz σ oblika $\sigma' = \sigma, n$ (primenom ν ili π-pravila), n je najmanji prirodan broj takav da je σ' bez restrikcija.

Ove izmene obezbeđuju da svaka grana tabloa koja nije zatvorena, a na kojoj su primenjena sva pravila izvođenja, ima sledeće osobine:

- ni za jedno iskazno slovo p nisu $\sigma\ T\ p$ i $\sigma\ F\ p$ istovremeno na grani,
- ako je neka $\sigma\ \alpha$-formula na grani, tada su na grani i odgovarajuće $\sigma\ \alpha_1$ i $\sigma\ \alpha_2$ formule,
- ako je neka $\sigma\ \beta$-formula na grani, onda je na grani bar jedna od formula $\sigma\ \beta_1$ ili $\sigma\ \beta_2$,
- ako je neka $\sigma\ \nu$-formula na grani, onda su na grani sve formule $\sigma'\ \nu_0$, za sve prefikse σ' dostižne iz σ (u slučaju logika K, T, $S4$, B i $S5$), odnosno za bar jedan prefiks σ' dostižan iz σ (u slučaju logika D, $D4$ i DB) i
- ako je neka $\sigma\ \pi$-formula na grani, onda je na grani i formula $\sigma'\ \pi_0$, za neki prefiks σ' dostižan iz σ

koje garantuju da je svaka otvorena grana na kojoj su obrađeni svi čvorovi zadovoljiva. Da bi se to videlo, dovoljno je posmatrati model čiji su svetovi prefiksi tabloa, relacija dostižnosti je upravo dostižnost u tablou, a za svako iskazno slovo važi $v(\sigma, p) = \top$ ako i samo ako je $\sigma\ T\ p$ na grani. Sada pretpostavimo da formula A nema dokaz, pa ni sistematski postupak ne dovodi do zatvorenog tabloa. Taj tablo ima barem jednu otvorenu granu[19] pomoću koje se konstruiše model za sve formule sa te grane. Kako se i formula $1\ F\ A$ nalazi na grani, znači da formula A nije valjana. Ξ

7.2.4 Primena tabloa u odlučivosti modalnih logika

Donekle izmenjena sistematska procedura konstrukcije tabloa se koristi kao osnova za utvrđivanje odlučivosti spominjanih modalnih logika.

U slučaju kada odgovarajuća relacija dostižnosti u klasi modalnih modela ne mora biti tranzitivna, jedina modifikacija je:

- formula se dodaje na kraj grane samo ako već nije prisutna na grani.

Na primer, prilikom primene ν-pravila na formulu $\sigma\ \nu$, kada god se pojavi novi prefiks σ' dostižan iz σ, na kraju grane se dodaje samo formulu $\sigma'\ \nu_0$ ukoliko se već ne nalazi na grani. Pretpostavka da se u konstrukciji dobila beskonačna grana dovodi do kontradikcije. Razmotrimo, na primer, logiku K. Pošto je broj podformula polazne formule konačan, a svaki prefiks se može pojaviti najviše jednom uz jednu podformulu, broj pojava svakog prefiksa na grani je konačan. Kako je grana beskonačna, sledi da se na njoj mora pojaviti beskonačno mnogo prefiksa. Za to postoje dve mogućnosti:

[19] Ovo je jasno ukoliko se konstrukcija izvede u konačnom broju koraka. Ako je konstrukcija beskonačna, prema lemi Königa, beskonačno drvo u kome svi čvorovi imaju konačan stepen grananja mora imati beskonačnu granu.

(1)	$1\ F\ \Box A \to \Box\Box A$
(2)	$1\ T\ \Box A$
(3)	$1\ F\ \Box\Box A$
(4)	$1,1\ F\ \Box A$
(5)	$1,1,1\ F\ A$
(6)	$1\ T\ A$
(7)	$1,1\ T\ A$

Slika 7.3. T-tablo za formulu $\Box A \to \Box\Box A$.

- postoji beskonačno mnogo prefiksa iste dužine ili
- postoji beskonačno mnogo različitih dužina prefiksa.

U prvom slučaju neka je n najmanja dužina prefiksa za koju se na grani javlja beskonačno mnogo prefiksa. Ne može biti $n = 1$, pošto je $\sigma = 1$ jedini prefiks dužine jedan. Jedini način da se uvede prefiks dužine n je da se primeni π-pravilo na prefiks dužine $n-1$. Kako se pravilo konstrukcije na svaku π-formulu primenjuje samo jednom, to znači da postoji beskonačno mnogo različitih prefiksa dužine $n-1$, što je kontradikcija. Ni drugi slučaj nije moguć jer se duži prefiksi dobijaju jedino primenom π-pravila, a broj π-formula koje su podformule polazne formule je konačan. Dakle, u slučaju netranzitivnih logika sistematski konstruisan tablo je uvek konačan.

Primer 7.2.5 Razmotrimo T-tablo za formulu $\Box A \to \Box\Box A$ dat na slici 7.3. Prema prethodnim definicijama, u slučaju logike T iz nekog prefiksa su dostižni on sam i sva njegova prosta proširenja. Čvorovi (2) i (3) se dobijaju iz (1) α-pravilom, a (4) se dobija iz (3) π-pravilom. Tom prilikom je uveden novi prefiks $1,1$. (5) se dobija iz (4) π-pravilom, pri čemu se uvodi novi prefiks $1,1,1$. Čvorovi (6) i (7) se dobijaju iz (2) ν-pravilom. Grana i tablo su završeni, ali ne i zatvoreni. Jedan T-model u kome polazna formula ne važi ima tri sveta. Pored refleksivnosti, prvi svet je dostižan iz drugog, a drugi iz trećeg. U prvom i drugom svetu važi A, a u trećem $\neg A$. Zbog toga u prvom svetu važi $\Box A$, u drugom $\neg\Box A$ i ponovo u prvom $\neg\Box\Box A$. Ξ

Prethodni dokaz konačnosti konstrukcije tabloa, međutim, ne prolazi u slučaju tranzitivnih logika. Recimo, na slici 7.4 je prikazan deo jednog $S4$-tabloa. Lako se uočava da se uvođenje novih prefiksa produžava unedogled.

Pa ipak, i tranzitivne modalne logike su odlučive. Najpre treba uočiti da se na bilo kojoj beskonačnoj grani, kao ni ranije, ne može javiti beskonačno mnogo prefiksa iste dužine, već samo beskonačno mnogo različitih dužina prefiksa. Svaki od prefiksa može stajati samo uz po jednu pojavu (označenu bilo sa T, bilo sa F) bilo koje podformule polazne formule. Na bilo kojoj beskonačnoj grani postoji bar jedan beskonačan niz prefiksa u kome je svaki sledeći prefiks dostižan iz prethodnog. U tom nizu moraju postojati dva prefiksa koji stoje uz iste označene formule. Zbog sistematičnosti konstrukcije, to isto važi i za njihove naslednike, čime se dobija jedan oblik periodičnog ponašanja. Pojava prva dva prefiksa u nizu koja stoje uz iste označene formule se mora dogoditi u konačnom broju koraka, pošto je broj

(1)	$\sigma\ F\ \Diamond\Box A$	
(2)	$\sigma\ F\ \Box A$	(ν-pravilo na (1))
(3)	$\sigma, 1\ F\ A$	(π-pravilo na (2))
(4)	$\sigma, 1\ F\ \Box A$	(ν-pravilo na (1))
(5)	$\sigma, 1, 1\ F\ A$	(π-pravilo na (4))
(6)	$\sigma, 1, 1\ F\ \Box A$	(ν-pravilo na (1))
	...	

Slika 7.4. Deo beskonačnog $S4$-tabloa za formulu $\Diamond\Box A$.

označenih podformula polazne formule konačan. Svaki beskonačni niz prefiksa se može prekinuti kod prvi put uočene pojave periodičnosti ove vrste. Ako bi se grana zatvorila u nizu prefiksa nakon pojave periodičnosti, očigledno je da se to mora dogoditi i pre te pojave. Zbog toga, dodavanje provere periodičnosti obezbeđuje da su i sistematski tabloi kod tranzitivnih logika uvek konačni.

Konačno, pošto je sistematska konstrukcija tabloa konačna, jednostavnom proverom se utvrđuje da li je tablo zatvoren, u kom slučaju je polazna formula valjana, inače tablo nije zatvoren i polazna formula nije valjana. Sve se ovo može obaviti u konačnom broju koraka, pa su razmatrane modalne logike odlučive.

7.3 Dualni tablo i rezolucija za modalne logike

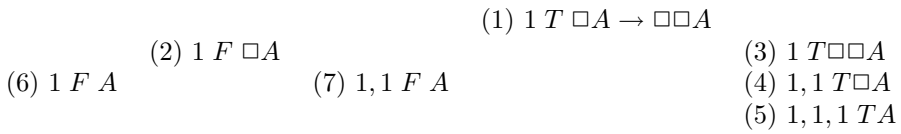
Ranije je rečeno da je postupak rezolucije primenljiv na logike kod kojih se proizvoljne formule mogu prevesti u oblik konjunktivne normalne forme. Zbog postojanja modalnih operatora to, u opštem slučaju, nije moguće kod modalnih logika, pa su i primene varijanti rezolucije retke. U ovom odeljku ćemo opisati pristup u kome se jedna varijanta rezolucije kombinuje sa metodom koja je dualna prefiksiranim tabloima iz odeljka 7.2. U nastavku ćemo pre svega ukazati na razlike između ovih postupaka. Razmatraćemo modalnu logiku T, dok se ostali slučajevi rešavaju analognim izmenama kao i kod prefiksiranih tabloa. Koristićemo većinu pojmova čije se definicije nalaze u odeljku o prefiksiranim tabloima: ujednačavajuća notacija, prefiksirane označene modalne formule, dostižnost između prefiksa itd.

7.3.1 Konstrukcija dualnih tabloa za modalne logike

Razlike u metodama prefiksiranih i dualnih tabloa se najpre javljaju u koracima konstrukcije tabloa.

U koren tabloa se sada postavlja formula $1\ T\ A$, a ne $1\ F\ A$. Ovo će za posledicu imati to da više nećemo razmatrati zatvorenost grana tabloa, pošto, intuitivno, polazimo od pretpostavke da je formula A tačna u svetu označenom prefiksom 1. Dalje, neka je čvor Y kraj neke grane do tog momenta konstruisanog drveta. Zavisno od formula na grani koja sadrži Y, nastavak konstrukcije je moguće izvesti nekim od pravila:

- α-pravilo: ako je neka $\sigma\ \alpha$ formula na grani, tada se grana produžava sa dva nova čvora od kojih jedan sadrži $\sigma\ \alpha_1$ formulu, a drugi $\sigma\ \alpha_2$ formulu,



Slika 7.5. Dualni T-tablo za formulu $\Box A \to \Box\Box A$.

- β-pravilo: ako je neka $\sigma\ \beta$ formula na grani, tada se grana u čvoru Y grana, pri čemu levi naslednik sadrži $\sigma\ \beta_1$, a desni $\sigma\ \beta_2$ formulu.

- ν-pravilo: ako je neka $\sigma\ \nu$ formula na grani i ako ovo pravilo nije do sada primenjeno na istu prefiksiranu formulu (bilo gde u tablou), tada se grana produžava čvorom u kome se nalazi odgovarajuća $\sigma'\ \nu_0$-formula, gde je σ' prosto proširenje bez restrikcija prefiksa σ; ako je negde u tablou ovo pravilo već primenjeno na istu prefiksiranu formulu pri čemu je uveden čvor koji sadrži prefiksiranu formulu $\sigma''\ \nu_0$, grana na kojoj je razmatrani čvor se produžava čvorom koji sadrži upravo $\sigma''\ \nu_0$.

- π-pravilo: neka je formula $\sigma\ \pi$ formula na grani i neka je σ' prefiks dostižan iz σ takav da π-pravilo nije do sada primenjeno na posmatranu formulu i prefiks σ' (u suprotnom se pravilo ne primenjuje); ako ovo pravilo nije do sada uopšte primenenjeno na posmatranu formulu, grana se produžava čvorom u kome se nalazi formula $\sigma'\ \pi_0$ (ovakav čvor nazvaćemo prvim π_0 potomkom razmatranog π-čvora); ako je ovo pravilo već primenjeno na razmatrani π-čvor, grana se razgranava u čvoru Y, prethodniku prvog π_0 potomka razmatranog π-čvora, a novi čvor sadži formulu $\sigma'\ \pi_0$.

Svaki čvor na jednoj grani se redukuje najviše jednom, nakon čega je završen na toj grani, sem π-čvorova koji se ponašaju nešto drugačije: čvor koji sadrži formulu $\sigma\ \pi$ je završen ako ne postoji i neće postojati ni jedan novi prefiks σ'' dostižan iz σ, a π-pravilo je primenjeno na sve prefikse σ' dostižne iz σ. Čvorovi koji sadrže atomske formule su takođe završeni, pošto se na njih ne može primeniti ni jedno od nabrojanih pravila.

Ovde treba obratiti pažnju na ν i π-pravila u ovom sistemu i njihovu dualnost u odnosu na istoimena pravila kod metode prefiksiranih tabloa. Naime, u metodi dualnih tabloa novi prefiksi se uvode ν-pravilom, dok kod prefiksiranih tabloa tu ulogu ima π-pravilo. Sledeći primer pokazuje kako u metodi dualnih tabloa izgleda prefiksirani tablo prikazan u primeru 7.2.5.

Primer 7.3.1 Dualni tablo za formulu $\Box A \to \Box\Box A$ je dat na slici 7.5. Najpre, u čvoru (1) se nalazi formula sa prefiksom T, a ne F kao što je prethodno bio slučaj. Čvorovi (2) i (3) se dobijaju iz (1) β-pravilom. Čvorovi (4) i (5) se dobijaju iz (3), odnosno (4), primenama ν-pravila, dok se čvorovi (6) i (7) dobijaju iz (2) primenama π-pravila. Nakon dobijanja čvora (7) konstrukcija (dualnog) tabloa je završena. Ξ

7.3.2 Pravilo dualne rezolucije

Kao što je ranije rečeno, metoda dualnih tabloa kombinuje tablo pristup i jednu varijantu rezolucije. Zapravo, i ovde ćemo koristiti atribut dualna uz rezoluciju, pošto se pod klauzom smatra konjunkcija (a ne disjunkcija kao u definiciji 3.7.2) označenih atomskih formula.

Definicija 7.3.2 *Dualna klauza* je konjunkcija označenih atomskih formula. Dualna klauza je *prazna* ako je konjunkcija nula označenih atomskih formula.

Definicija 7.3.3 Neka su C_1 i C_2 dualne klauze i neka je A atomska formula takva da $T\ A \in C_1$ i $F\ A \in C_2$, tada je *dualna rezolventa* klauza C_1 i C_2, klauza

$$Res(C_1, C_2, A) = (C_1 \setminus \{T\ A\}) \cup (C_2 \setminus \{F\ A\})$$

dobijena tako što je iz C_1 izbrisana formula $T\ A$, a $F\ A$ iz C_2, i preostale označene atomske formule povezane konjunkcijom.

Slično proceduri rezolucije iz odeljka 3.7 i ovde se pokušava izvođenje prazne klauze, ali je u slučaju uspeha zaključak da je skup dualnih klauza valjan, tj. tačan pri svakoj interpretaciji. Dokaz tvrđenja se sprovodi u potpunosti dualno dokazu 3.7.6.

Teorema 7.3.4 Skup klauza F je tačan pri svakoj interpretaciji ako i samo ako je prazna klauza u skupu $Res^*(F)$.

7.3.3 Potpunost metode dualnih tabloa

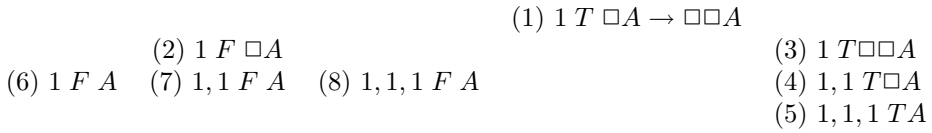
Potpunost metode dualnih tabloa se dokazuje u dva koraka. Najpre se pokazuje kakav je odnos posmatrane formule i skupova prefiksiranih označenih atomskih formula sa grana dualnog tabloa, a potom se u proveri eventualne valjanost skupa tih skupova koristi dualna rezolucija.

Teorema 7.3.5 Neka je dat završeni dualni tablo u čijem korenu je formula $1\ T\ A$. Formula A je T-valjana akko za svaki T-model M i svaku interpretaciju I koja slika prefikse tabloa u svetove iz M postoji bar jedna grana tabloa sa koje je skup svih atomskih označenih formula sa prefiksima zadovoljen pri interpretaciji I.

Dokaz. Dokaz se sprovodi analogno dokazu teoreme 7.2.4. Ξ

Da bismo primenili dualnu rezoluciju na skup skupova prefiksiranih označenih atomskih formula sa grana dualnog tabloa, potrebno je nekako preći sa označenih atomskih modalnih formula na označene klasične atomske formule. U tom cilju se uvodi pojovim indukovane dualne klauze i indukovane valuacije.

Definicija 7.3.6 Neka je $\{\sigma_1\ X_1\ A_1, \ldots, \sigma_n\ X_n\ A_n\}$ skup prefiksiranih označenih atomskih formula sa neke završene grane dualnog tabloa, gde je $X_i \in \{T, F\}$. *Indukovana dualna klauza* je konjunkcija $\wedge_{i=1}^{n} X_i\ A_{i,\sigma_i}$, gde su A_{i,σ_i} iskazna slova. *Indukovani skup dualnih klauza* je skup svih dualnih klauza indukovanih skupovima prefiksiranih označenih atomskih formula sa grana nekog završenog tabloa. Za neki T-model M i T-interpretaciju I koja prefikse nekog završenog tabloa preslikava u svetove modela M, *indukovana valuacija* I_v se definiše tako da važi: $I_v \models X_i\ A_{i,\sigma_i}$ akko $I \models \sigma_i\ X_i\ A_i$.



Slika 7.6. Dualni $S4$-tablo za formulu $\Box A \to \Box\Box A$.

Sada se lako pokazuje tvrđenje:

Teorema 7.3.7 Konačna konjunkcija označenih atomskih formula važi pri indukovanoj valuaciji I_v akko skup odgovarajućih označenih prefiksiranih atomskih formula važi pri interpretaciji I.

odakle direktno sledi teorema potpunosti:

Teorema 7.3.8 Modalna formula A je T-valjana akko se iz indukovanog skupa dualnih klauza njenog dualnog tabloa primenom pravila dualne rezolucije izvodi prazna klauza.

Primer 7.3.9 U dualnom tablou iz primera 7.3.1, indukovani skup dualnih klauza je oblika: $\{\{F\ A_1\}, \{F\ A_{1,1}\}, \{T\ A_{1,1,1}\}\}$. Pošto se iz ovog skupa ne može izvesti prazna klauza, formula $\Box A \to \Box\Box A$ nije T-valjana.

Na slici 7.6 prikazan je dialni $S4$-tablo za istu formulu. Razlika u relaciji dostižnosti omogućava i pojavu čvora (8), pa je sada je indukovani skup $\{\{F\ A_1\}, \{F\ A_{1,1}\}, \{F\ A_{1,1,1}\}, \{T\ A_{1,1,1}\}\}$. Očigledno, jednom primenom pravila rezolucije na poslednje dve klauze iz skupa izvodi se prazna klauza, te je posmatrana formula $S4$-valjana. □

7.4 Zamene za klasičnu logiku

Istraživanja u oblasti veštačke inteligencije su pokazala da su postupci zaključivanja zasnovani na klasičnoj logici često nedovoljno fleksibilni da odgovore na zahteve koji se javljaju u stvarnosti kada znanje ne mora biti potpuno ili konzistentno, pa čak ne mora imati ni jednostavnu tačno-netačno formu istinitosti. U narednim odeljcima ćemo prikazati pristupe koji su konkurentski klasičnoj logici, a čije primene u manjoj ili većoj meri rešavaju neke od tih problema. U nekim od tih logika izvode se formule koje se ne mogu dobiti u klasičnom pristupu, dok se opet u nekim drugim logikama ne mogu izvesti sve formule koje su klasično valjane.

7.4.1 Nemonotono zaključivanje

Od inteligentnih sistema se zahteva da se u situacijama u kojima poseduju nepotpune ili protivrečne informacije ne blokiraju, već da za dato stanje pruže neki prihvatljiv, mada ne i nužan, zaključak. U slučaju pristizanja novih podataka, prethodni zaključci se mogu odbaciti ili zadržati, u čemu se ogleda nemonotonost zaključivanja[20].

[20] U monotonom zaključivanju pristizanje novih informacija ne utiče na veće formirane zaključke. Klasična logike je primer monotone logike.

Primer 7.4.1 Neka se u bazi znanja nalaze informacije da: 'ptice *po pravilu*[21] lete', 'pingvini ne lete', 'pingvini su ptice' i da je neka konkretna životinja koju posmatramo ptica. Sa stanovišta klasične logike ova baza znanja nije konzistentna (ako zanemarimo kvalifikator *po pravilu*), jer za bilo kog konkretnog pingvina zaključujemo i da leti (jer su pingvini ptice, a ptice lete) i da ne leti (jer pingvini ne lete), pa je time i neupotrebljiva. Međutim, iz ugla nemonontonog rezonovanja koje pokušava da oponaša svakodnevno zaključivanje, bilo bi korisno zaključiti da naša životinja, pošto je ptica - leti. Ako bi se naknadno pojavio podatak da je reč o pingvinu, taj zaključak bi trebalo preispitati, odbaciti i izvesti novi - da ne leti.

Slično situacija se javlja i kada neka osoba ima motiv, recimo finansijski, za zločin i nema alibi, pa možemo reći da je tada po pravilu i osumnjičena. Ako se alibi ipak pojavi, osoba se oslobađa sumnje. Ξ

Primer 137 ilustruje elemente nemonotonog zaključivanja. Najpre je, iz (klasično) nekonzistentne baze znanja koja sadrži skup opštih pravila ('ptice po pravilu lete') koja imaju izuzetke ('pingvini su ptice' i 'pingvini ne lete'), izveden prihvatljiv zaključak, da bi zatim, nakon pristizanja novih informacija, on bio revidiran.

Formalizacija ovakvog načina zaključivanja dovela je do razvoja *zaključivanja po pravilu*[22]. Osnovni sistem za zaključivanje po pravilu označava se sa P, a centralni pojam koji se u njemu razmatra je pojam *difolt pravila*[23]. Difolt pravilo možemo shvatiti kao iskaz 'ako važi α, onda po pravilu važi β' i zapisati u obliku[24] $\alpha \mapsto \beta$. *Difolt bazu* čini skup difolt pravila i klasičnih formula koje opisuju naše znanje.

Zaključivanje po pravilu u sistemu P je opisano pomoću odgovarajuće relacije posledice $\mid\!\sim$. Preciznije, zadatak je odrediti skup prihvatljivih posledica difolt baze *Delta*. Tada, ako α predstavlja opis znanja i važi da je $\Delta \mid\!\sim \alpha \mapsto \beta$, po pravilu zaključujemo da važi β (u oznaci $\alpha \mid\!\sim_\Delta \beta$). Minimalan skup osobina koje treba da ispunjava relacija $\mid\!\sim$ je sintaksno opisan sistemom P (ovde $\models$ označava klasičnu valjanost):

- $\alpha \mapsto \alpha$ (refleksivnost)
- iz $\models \alpha \leftrightarrow \alpha'$ i $\alpha \mapsto \beta$, izvesti $\alpha' \mapsto \beta$ (leva logička ekvivalencija)
- iz $\models \beta \rightarrow \beta'$ i $\alpha \mapsto \beta$, izvesti $\alpha \mapsto \beta'$ (desno slabljenje)
- iz $\alpha \mapsto \beta$ i $\alpha \mapsto \gamma$, izvesti $\alpha \mapsto \beta \wedge \gamma$ (I)
- iz $\alpha \mapsto \gamma$ i $\beta \mapsto \gamma$, izvesti $\alpha \vee \beta \mapsto \gamma$ (ILI)
- iz $\alpha \mapsto \beta$ i $\alpha \mapsto \gamma$, izvesti $\alpha \wedge \beta \mapsto \gamma$ (ograničena monotonost[25]).

Difolt pravilo $\alpha \mapsto \beta$ je (sintaksna) posledica difolt baze Δ u sistemu P (u oznaci $\Delta \vdash_P \alpha \mapsto \beta$), ako se $\alpha \mapsto \beta$ može izvesti iz Δ primenom aksiome i pravila sistema P.

[21] Obratite pažnju na istaknutu reč *po pravilu*. Zbog nje ovaj iskaz ne mora biti tačan baš u svim situacijama.

[22] Default reasoning.

[23] Default rule. Kao i u slučaju fazi logike, ni ovde nismo pronašli pogodan prevod na naš jezik.

[24] Različiti autori koriste drugačije simbole, recimo $\rightarrow$ ili $\mid\!\sim$ za zapisivanje difolt pravila. U našem slučaju te oznake bi mogle uneti zabunu imajući u vidu njihovu upotrebu u drugim kontekstima.

[25] Cautious monotonicity.

Razmotrimo ukratko aksiomu i pravila sistema P. Aksioma refleksivnosti je prisutna faktički u svim sistemima koji opisuju bilo kakvu relaciju posledice. Leva logička ekvivalencija izraža zahtev da ekvivalentne formule imaju iste posledice, tj. da skup posledica zavisi od značenja, a ne od sintaksnog oblika formule. Pravilo desnog slabljenja znači da se kao posledica prihvata i sve što logički sledi iz onoga što smatramo prihvatljivim posledicama difolt baze. Pravilom I se zahteva da je konjunkcija dve posledice takođe prihvatljiva posledica difolt baze, dok pravilo ILI kaže da je svaka formula koja je posledica dveju različitih formula ponaosob, takođe posledica i njihove disjunkcije. Konačno, ograničenom monotonošću se zahteva da prihvatljive posledice neke difolt baze ne mogu dovesti do odbacivanja prethodnih zaključaka. Ovo pravilo ima suštinski značaj u sistemu P. Iako liči, ono je strogo slabije od pravila koje garantuje monotonost zaključivanja. Preciznije, pokazuje se da ni jedno od sledećih pravila nije izvodivo u sistemu P, a da njihovo dodavanje tom sistemu dovodi do relacije posledice koja odgovara klasičnoj (monotonoj) logici:

- iz $\models \alpha \rightarrow \alpha'$ i $\alpha' \mapsto \beta$, izvesti $\alpha \mapsto \beta$ (monotonost)
- iz $\alpha \mapsto \beta \rightarrow \gamma$ izvesti $\alpha \wedge \beta \mapsto \gamma$ (jednostavni smer dedukcije)
- iz $\alpha \mapsto \beta$ i $\beta \mapsto \gamma$, izvesti $\alpha \mapsto \gamma$ (tranzitivnost)
- iz $\alpha \mapsto \beta$, izvesti $\neg\beta \mapsto \neg\alpha$ (kontrapozicija).

Relacija $\mid\!\sim$ se karakteriše i semantički pomoću relacije zadovoljenja $\models$ u nekoj klasi modela, tj. $\Delta \mid\!\sim \alpha \mapsto \beta$ ako i samo ako u svakom modelu M neke određene klase koji zadovoljava Δ, važi $M \models \alpha \mapsto \beta$. Jednu od spomenutih klasa modela čine takozvani *preferencijalni modeli*. Preferencijalni model M je struktura $\langle W, v, < \rangle$ koju sačinjavaju:

- skup svetova W,
- valuacija $v : W \times \Phi \rightarrow \{\top, \bot\}$ koja svakom svetu i svakom iskaznom slovu pridružuje jednu od istinitosnih vrednosti $\top$ i $\bot$ i
- $<$ je relacija striktnog parcijalnog uređenja (tranzitivna irrefleksivna relacija) na skupu svetova takva da svaki skup svetova definabilan formulom ($[\alpha] = \{s \in W : s \models \alpha\}$) zadovoljava uslov:
  - za svaki svet $t \in [\alpha]$ ili postoji minimalan svet $u \in [\alpha]$ takav da je $u < t$ ili je t minimalan u skupu $[\alpha]$.

U preferencijalnom modelu minimalni svetovi opisuju one situacije u kojima možemo zaključivati po pravilu, dok u svetovima koji su viši u hijerarhiji $<$ mogu važiti i neki izuzeci. Preciznije, difolt pravilo $\alpha \mapsto \beta$ važi u preferencijalnom modelu M ako za svaki svet s koji je minimalan u $[\alpha]$ važi $s \models \beta$. Da sistem P karakteriše relaciju $\mid\!\sim$ u klasi svih preferencijalnih modela tvrdi sledeća teorema:

Teorema 7.4.2 $\Delta \mid\!\sim \alpha \mapsto \beta$ u odnosu na klasu svih preferencijalnih modela ako i samo ako $\Delta \vdash_P \alpha \mapsto \beta$.

Primer 7.4.3 Difolt baza Δ koja bi odgovarala problemu sa letenjem ptica i neletenjem pingvina iz primera 137 može sadržati:

$$\{pt \mapsto l,\ pi \mapsto pt,\ pi \mapsto \neg l\}$$

gde pt, l i pi redom znače 'ptica', 'leti' i 'pingvin'. U sistemu P bismo mogli zaključiti:

- $pt|\!\sim_\Delta l$ (ptice lete),
- $pi|\!\sim_\Delta \neg l$ (pingvini ne lete),
- $pt \wedge pi|\!\sim_\Delta \neg l$ (ptica koja je pingvin ne leti),
- $l|\!\sim_\Delta \neg pi$ (ako nešto leti, nije pingvin),
- $pt \vee pi|\!\sim_\Delta l$ (ako je nešto ptica ili pingvin, onda leti - jer pingvini predstavljaju relativno mali deo ptica),
- $pt \vee pi|\!\sim_\Delta \neg pi$ (slično kao i malopre, pingvini predstavljaju relativno mali deo ptica) itd.,

dok se $pi|\!\sim_\Delta l$ (pingvin leti) ne bismo moglo dobiti. Ξ

Iz ovog primera se može naslutiti da sistem P korektno bira najspecifičnije situacije i da u prisustvu izuzetaka blokira primenu manje specifičnih difolt pravila. Na primer, ako znamo da je ptica o kojoj se radi pingvin, onda se koristi difolf pravilo $pi \mapsto \neg l$, a ne $pt \mapsto l$. Sledeći primer je u literaturi poznat pod nazivom Niksonov dijamant.

Primer 7.4.4 Američki predsednik Ričard Nikson je bio pripadnik kvekerske verske zajednice i Republikanske partije. Neka difolt baza Δ sadrži:

$$\{r \mapsto \neg p,\ q \mapsto p\}$$

gde r, p i q redom znače 'republikanac', 'pacifist' i 'kveker', što bi trebalo da znači da republikanci po pravili nisu pacifisti, dok kvekeri to jesu. U sistemu P ne bismo mogli zaključiti:

- $r \wedge q|\!\sim_\Delta r$, ni suprotno
- $r \wedge q|\!\sim_\Delta \neg r$,

tj., za Niksona ne bismo mogli reći ni da je pacifista, ni da to nije. Ξ

Zanimljivo je da se, slično kao sa različitim formalnim modelima izračunavanja, i ovde pokazalo da su mnogi predloženi sistemi ekvivalentni sa P, što je dovelo do toga da P predstavlja jednu vrstu gotovo plebiscitarno prihvaćenog minimalnog sistema nemonotonog zaključivanja, tj. polaznu osnovu čijim obogaćivanjem se dobijaju drugi formalni sistemi u ovoj oblasti. U sistemu P su formalizovani osnovni principi zaključivanja po pravilu:

- specifičniji podaci treba da dovedu do revizije zaključaka donetih na osnovu opštijih znanja, kao u primerima 137 i 7.4.3,
- kad postoji nezavisna argumentacija i za neki zaključak i za njegovu negaciju, ne bi trebalo zaključiti ni jedno ni drugo, kao u primeru 7.4.4
- zaključivanje ne bi trebalo da zavisi od forme zapisivanja baze znanje itd.

Za razliku od klasične logike koja je nastala kao jedno od oruđa za analizu dobro definisanih matematičkih problema, intuicija o tome šta je to prihvatljivo što bi trebalo da smo u stanju izvesti u sistemima sa nemonotonim zaključivanjem nije dovoljno razvijena. Posledica toga je da se kod predloženih proširenja sistema P pokazalo da svako od njih ima svoje prednosti, ali i neke mane jer bilo da ne uspeva da modelira sve interesantne situacije, bilo da ispunjavajući jednu grupu narušava drugu grupu zahteva. U primeru 7.4.5 su navedena dva specifična zahteva za nemonotono zaključivanje koje sistem P ne ispunjava.

Primer 7.4.5 Pretpostavimo da je β prihvatljiva posledica od α i da je γ irelevantno za β. Tada bi trebalo da bude i $\alpha \wedge \gamma \mapsto \beta$, tj. dodavanje irelevantnih podataka ne treba da utiče na zaključke. Ovde ćemo zanemariti šta znači da je γ irelevantno za β, ali se neka intuicija može steći ako pretpostavimo da α, β i γ redom označavaju: 'ptica', 'leteti' i 'biti crven'. Tada bi trebalo da informacija da je neka ptica crvena, ne utiče na zaključak da ta ptica leti.

Za izuzetke treba da važe druge karakteristike opštijih klasa u kojima mal-opređašnji izuzeci to više nisu. ptica u kojima pingvini nisu izuzeci. Recimo, pravilo 'ptice imaju noge' treba da bude primenljivo i na pingvine, iako su pingvini specifične ptice koje ne lete. Ξ

Konačno, recimo i da postoje i drugi pristupi nemonotonom zaključivanju: difolt logike, sistemi zasnovani na ograničenjima[26], sistemi zasnovani na modalnoj logici poput autoepisteničke logike itd.

7.4.2 Intuicionistička logika

Radikalni zahtevi koje postavljaju intuicionisti se ogledaju u tome da se značenje nekog logičkog operatora daje u odnosu na to šta treba dokazati da bi važila formula u kojoj je taj operator glavni. Drugim rečima, tačnost ima smisao dokazivosti. Ovo dovodi do toga da neke klasično valjane formule, poput:

- $\alpha \vee \neg\alpha$ ili
- $(\neg\alpha \rightarrow \neg\beta) \rightarrow (\beta \rightarrow \alpha)$

nisu valjane u intuicionističkoj logici. Na primer, intuicionisti određuju značenje operatora disjunkcije tako što se dokaz za $\alpha \vee \beta$ sastoji od dokaza za α ili dokaza za β. U klasičnom pristupu zakon isključenja trećeg $(\alpha \vee \neg\alpha)$ je trivijalna posledica definicije istinitosnih tablica. Kod intuicionista se, međutim, zahteva dokazivanje bilo formule α, bilo njene negacije, a kako to nekada nismo u stanju da uradimo,

[26] Circumscription.

$\alpha \vee \neg\alpha$ se ne prihvata kao intuicionistički valjana formula. Slično je i sa zakonom kontrapozicije. Takođe, u intuicionističkoj logici svi logički operatori su nezavisni.

Značenje formula u intuicionističkoj logici se daje upotrebom Kripkeovih modela oblika $\langle W, R, \models\rangle$, gde su W skup svetova, R refleksivna i tranzitivna relacija na W i $\models$ relacija zadovoljivosti koja ima neke posebne osobine. Svetovi modela se mogu shvatiti kao skupovi informacija, odnosno stanja znanja u određenom trenutku. Relacija R predstavlja moguća proširenja aktuelnog stanja znanja, tako da se wRu tumači kao 'ako sada znam w, onda ću u budućnosti možda znati u'. Relacija $\models$ ispunjava sledeće zahteve:

- za svako iskazno slovo p, ako $w \models p$, onda za svaki u za koji je wRu, važi $u \models p$,
- $w \models \alpha \wedge \beta$ ako i samo ako $w \models \alpha$ i $w \models \beta$,
- $w \models \alpha \vee \beta$ ako i samo ako $w \models \alpha$ ili $w \models \beta$,
- $w \models \neg\alpha$ ako i samo ako za svaki u za koji je wRu nije $u \models \alpha$ i
- $w \models \alpha \to \beta$ ako i samo ako za svaki u za koji je wRu, ako $u \models \alpha$, onda $u \models \beta$.

Sada $w \models \alpha$ možemo tumačiti kao 'ako znam w, onda mogu zaključiti α.' Uopštenje prvog uslova za relaciju $\models$ je *princip postojanosti*:

Teorema 7.4.6 U intuicionističkom modelu $\langle W, R, \models\rangle$ za svet $w \in W$ važi $w \models \alpha$ ako i samo ako za svaki svet u za koji je wRu važi $u \models \alpha$.

koji se može shvatiti kao 'znanje se ne gubi.' Primetimo da princip postojanosti dovodi do toga da se za neke formule, pa i za iskazna slova, ne zna da li važe u nekom svetu ili ne, tj. model ne daje potpuno određenje istinitosnih vrednosti kao što je to do sada bio slučaj.

Formula α je *valjana u modelu* ako za svaki svet modela važi $w \models \alpha$, a *intuicionistički valjana*, ako je valjana u svim modelima.

Primer 7.4.7 U intuicionističkom modelu $\langle W, R, \models\rangle$, gde je $W = \{w_1, w_2, w_3\}$ i $R = \{(w_1, w_1), (w_1, w_2), (w_1, w_3), (w_2, w_2), (w_3, w_3)\}$, i za iskazno slovo p važi $w_2 \models p$ i $w_3 \models \neg p$, u svetu w_1 ne važi formula $p \vee \neg p$ jer je:

- $w_1 \not\models p$, pošto je $w_1 R w_3$ i $w_3 \not\models p$ i
- $w_1 \not\models \neg p$, pošto je $w_1 R w_2$ i $w_2 \models p$. Ξ

Ovde prikazana iskazna intuicionistička logika je odlučiva i za nju je razvijena metoda tabloa za dokazivanje teorema.

Jedna od primena intuicionističke logike je u semantici programskih jezika. Kako se intuicionistički Kripkeovi modeli mogu shvatiti kao prikaz narastanja znanja, intuicionistička logika se koristi i za bazu logika za zaključivanje u prisustvu nekompletnih informacija.

7.4.3 Viševrednosne logike

U slučaju viševrednosnih logika pored istinitosnih vrednosti tačno i netačno pojavlju se i druge. Na primer, kod logika sa tri istinitosne vrednosti, ta treća vrednost se u raznim pristupima tumači kao: nepoznato, trenutno nema istinitosnu vrednost, vrednost je besmislena, ... Zavisno od tog tumačenja se definišu vrednosti logičkih operacija. Recimo, u logici u kojoj je treća istinitosna vrednost shvaćena kao nepoznato, tablica implikacije je kao u tabeli 7.6, gde se ta vrednost predstavlja slovom u.

$A \setminus B$	$\top$	$\bot$	u
$\top$	$\top$	$\bot$	u
$\bot$	$\top$	$\top$	$\top$
u	$\top$	u	u

Tabela 7.6. Tablica implikacije $A \to B$ u jednoj trovrednosnoj logici.

Tako je, kao i u klasičnoj logici, netačno $\top \to \bot$, ali ako istinitost premise nije poznata, $u \to \bot$ nije ni tačno, ni netačno.

Pored trovalentnih postoje i logike sa više istinitosnih vrednosti, uključujući i one sa beskonačno mnogo istinitosnih vrednosti.

Viševrednosne logike se koriste za formalizaciju situacija u kojima je znanje nekompletno, ali se tokom vremena ne odbacuju njegovi delovi. Povećanje interesovanja za ove logike posledica je i razvoja fazi logika u kojima se numerički karakterišu neprecizni kriterijumi pripadanja, poput $Visok(x)$.

7.5 Za dalje proučavanje

Pregled neklasičnih logika i njihovih primena u računarstvu dat je u [123]. Referentni tekstovi o modalnim logikama su [16, 35, 54, 55]. Matematički i filozofski problemi u modalnim sistemima prvog reda analizirani su podrobno u [37]. Metoda tabloa za modalne logike je uvedena u [34], dok su modalni dualni tabloi opisani u [89]. Dualni tablo (za klasičnu logiku) su predloženi u [62]. Pregled temporalnih logika može se naći u [14, 126], u [74] se ovim logikama pristupa koristeći tabloe, dok se razne primene diskutuju u [27]. [46] sadrži opise i primene modalnih logika znanja i/ili verovanja. Dinamičke logike su analizirane u [49]. Opisi raznih iskaznih i predikatskih verovatnosnih logika dati su u [92, 94]. Osnovne ideje metode provere modela prikazane su u [13, 27, 45]. Oblast nemononotonog zaključivanja u ovoj knjizi nije detaljno obrađena zbog velikog obima postojećeg materijala. Kao opširniji uvodni tekstovi sa puno daljih referenci mogu poslužiti [36, 40, 84]. Sistem P koji je prihvaćen kao nesumnjivo jezgro za nemonoitono zaključivanje uveden je i analiziran u [69, 72]. Prikaz difolt-logika se može naći u [111, 114]. Intuicionistička logika je analizirana u [16, 33]. Pregled viševrednosnih logika je dat u [124], dok se analiza fazi logika sa stanovišta matematičke logike i dugačka bibliografija nalazi u [43]. Na našem jeziku se viševrednosne logike u kojima se istinitost iskaza shvata kao verovatnoća proučavaju u [70], dok je jedan praktičan pristup fazi logici prikazan u [122].

8

Verovatnosne logike

U ovom poglavlju ćemo prikazati logike sa verovatnosnim operatorima koje predstavljaju jednu posebnu varijantu modalnih logika. Nove izražajne mogućnosti dozvoliće nam da rezonujemo o verovatnoćama događaja, na primer da formalno zapišemo iskaze poput: "Ako je verovatnoća događaja A jednaka s i verovatnoća da je događaj B posledica događaja A jednaka t, onda je verovatnoća događaja B jednaka r." U nastavku ćemo analizirati nekoliko vrsta verovatnosnih logika, njihove formalne jezike, aksiomatizacije i pitanje odlučivost.

8.1 Rezonovanje o verovatnoći

Rezonovanje o verovatnoći predstavlja svojevrsno preplitanje dve teorije - matematičke logike i verovatnoće. Nastanak teorije verovatnoće se povezuje sa Gerolamo Cardan-om (Kardan, 1501 – 1576), Galileo Galilei-em (1564 – 1642) i Blaise Pascal-om (Paskal, 1623 – 1662) koji su proučavali igre na sreću. Sve do kraja XIX veka, verovatnoća događaja je shvatana kao šansa, tj. količnik broja povoljnih mogućnosti i ukupnog broja slučajeva ili kao relativna učestanost uspešnih događaja u ukupnom broju pokušaja u nekom eksperimentu. Međutim, u rečenici "Verovatnoća da je Homer spevao Odiseju je 0.3" verovatnoću ne možemo tumačiti spomenutim količnicima pošto ne postoji ni ukupan broj slučajeva, ni niz eksperimenata u kojima sa sastavlja Odiseja. Verovatnoća ovde ima smisao stepena verovanja u iskaz, odnosno opisa stanja znanja. Nazovimo ova dva pristupa interpretaciji verovatnoće redom:

- statistički pristup (objektivistički, empirijski), odnosno
- stepen verovanja (subjektivistički, lični pristup).

Prethodni primeri ilustruju višesmislenost upotrebe reči verovatnoća, kao i da ni jedan od spomenutih koncepata nije ni beskoristan, ni dovoljan. Šta više, videćemo da ovi koncepti dovode do različitih logičkih sistema, bez obzira što su zakoni koji važe u oba slučaja isti.

Istraživanja u oblastima matematičke logike i teorije verovatnoće često su kroz istoriju imala dodirnih tačaka: pokušavano je zasnivanje teorije verovatnoće na

logičkim osnovama, proučavane su logike u kojima se istinitosna vrednost formula izračunava pomoću funkcija koje su verovatnosne mere ili liče na njih, proširivan je logički jezik konstrukcijama koje omogućavaju da se neposredno govori o verovatnoći itd. Od sredine 80-tih godina rad u oblasti veštačke inteligencije, preciznije problemi zaključivanja u situacijama kada je znanje nepotpuno ili dvosmisleno, ili kada nije jasno koja pravila zaključivanja su primenljiva, otvorio je nova pitanja i dodatno inspirisao istraživače.

Iako je još Lajbnic u okviru razvoja metoda poznatog kao *characteristica generalis* pokušavao da primeni logički zasnovan postupak u verifikaciji tvrđenja iz teorije verovatnoće, najveći doprinos oblasti među ranim istraživačim dao je DŽordž Bul (George Boole, 1815 – 1864). Veliki deo njegovog rada je upravo posvećen proučavanju odnosa logičkog računa i računa događaja koji se danas shvata kao standardni deo teorije verovatnoće. Iz ugla koji je zastupljen u daljem tekstu, međutim, interesantne su logike u čijim objekt jezicima se pojavljuju simboli (verovatnosni kvantifikatori i verovatnosni operatori) kojima se izražava verovatnoća. Prve takve logike proučavao je Jerome Keisler (Džerom Kisler) koji je uveo verovatnosne kvantifikatore oblika $Px \geq r$ i modele u kojima se verovatnoća definiše nad podskupovima domena. U logikama ovog tipa verovatnosne formule su oblika

$$(Px \geq r)\varphi(x)$$

a interpretiraju se sa: verovatnoća skupa $\{x : \varphi(x)\}$ je veća do jednaka od r. Pri tome istinitosna vrednost formule je bilo tačno, bilo netačno, kao i u klasičnoj logici. Formule u kojima, kao u prethodnom primeru, učestvuju verovatnosni kvantifikatori pogodne su za rad sa statistički interpretiranom verovatnoćom.

Nils Nilsson je u razmatrao osnove verovatnosnog rezonovanja koje se koristi u veštačkoj inteligenciji. Nilsson se nije bavio logičkim aspektom problema, tj. nije ni pokušao da da nekakav aksiomatski sistem, dokaže teoremu potpunosti i sl. Njegov cilj je bio da opiše metodu kojom se verovatnoća, odnosnosno interval u kome se nalazi verovatnoća, neke rečenice izračunava na osnovu verovatnoća drugih rečenica i da na taj način da kriterijum za opravdavanje (odnosno kritiku) postupaka koji se koriste u programima veštačke inteligencije. Ipak, Nilsson-ov rad je bitan u kontekstu problematike koja se ovde razmatra iz najmanje dva razloga. Prvo, taj rad je doveo do velikog porasta interesovanja za probleme formalnog rezonovanja o verovatnoći koje do tada nije prevazilazilo relativno uzak krug istraživača u matematičkoj logici i drugo, rad je direktno inspirisao proučavanje jedne klase logika sa verovatnosnim operatorima kakve su i logike prikazane u ovom odeljku. Ukratko, Nillson posmatra skup $S = \{\alpha_1, \alpha_2, \ldots, \alpha_l\}$ od l rečenica i skup $W = \{w_1, w_2, \ldots, w_k\}$ interpretacija koje se međusobno razlikuju u istinitosnoj vrednosti bar jedne od rečenica. Preciznije, svaki svet predstavlja jednu klasu interpretacija koje se poklapaju na rečenicama iz S. Verovatnoća je definisana nad skupom svetova, pri čemu p_i označava verovatnoću sveta w_i uz jasna ograničenja da je $p_i \geq 0$ za svako $i = 1, k$ i da je $\sum_{i=1}^{k} p_i = 1$. Sada je verovatnoća rečenice α_j zbir verovatnoća svetova u kojima α_j važi. Ovo se može zapisati i matričnom jednačinom

$$\Pi = V \cdot P$$

gde su:

- Π vektor, tako da je Π_i verovatnoća skupa svetova u kojima važi formula α_i,
- V matrica za koju je $v_{i,j} = 1$ ako formula α_i važi u svetu w_j i
- P vektor verovatnoća svetova.

Metod koji je Nilsson predložio se zasniva na sledećem. Ako je dat skup formula, njemu se dodaje još jedna formula i analizom matrične jednačine se određuju granice intervala verovatnoće nove formule. Recimo, neka je skupu formula $\{\alpha_1, \alpha_1 \to \alpha_2\}$ dodata formula α_2. Tada je $l = 3$, $S = \{\alpha_1, \alpha_1 \to \alpha_2, \alpha_2\}$, $k = 4$, $w_1 \models \{\alpha_1, \alpha_1 \to \alpha_2, \alpha_2\}$, $w_2 \models \{\alpha_1, \neg(\alpha_1 \to \alpha_2), \neg\alpha_2\}$, $w_3 \models \{\neg\alpha_1, \alpha_1 \to \alpha_2, \alpha_2\}$ i $w_4 \models \{\neg\alpha_1, \alpha_1 \to \alpha_2, \neg\alpha_2\}$. Neka je dalje $P(\alpha_1)$ verovatnoća skupa svetova $\{w_1, w_2\}$ u kojima važi α_1, $P(\alpha_2)$ verovatnoća skupa svetova $\{w_1, w_3\}$ u kojima važi α_2 i $P(\alpha_1 \to \alpha_2)$ verovatnoća skupa svetova $\{w_1, w_3, w_4\}$ u kojima važi $\alpha_1 \to \alpha_2$. Tada je $P(\alpha_2) \leq P(\alpha_1 \to \alpha_2)$. Analizom matrične jednačine se zaključuje da mora biti $P(\alpha_1) + P(\alpha_1 \to \alpha_2) - 1 \leq P(\alpha_2)$ odakle se dobija nejednakost

$$P(\alpha_1) + P(\alpha_1 \to \alpha_2) - 1 \leq P(\alpha_2) \leq P(\alpha_1 \to \alpha_2)$$

koja se shvata kao pravilo verovatnosnog modus ponensa za izračunavanje intervala u kome se nalazi verovatnoća formule α_2 ako su poznate verovatnoće preostale dve formule. Aksiomatizovanjem verovatnosnog rezonovanja se nezavisno bavilo više autora od kojih neki po svemu sudeći i nisu znali za Nilsson-ov rad. Hronološki, Ron Fagin, Joseph Halpern i Nimrod Megiddo su prvi dali aksiomatizaciju logika za eksplicitno rezonovanje o verovatnoći. Rezultati koji će biti u nastavku opisani su u izvesnom smislu nastavak njihovih radova.

Za kraj ovog pregleda pomenimo da su u vezi rezonovanja o verovatnoći interesantni i pristupi poput viševrednosnih logika, fazi logike i Bayes-ovih mreža.

8.2 Iskazna verovatnosna logika LPP_1

U ovom odeljku ćemo razmotriti iskaznu verovatnosnu logiku LPP_1, u kojoj su dozvoljene iteracije verovatnoća, mešanje iskaznih i verovatnosnih formula, a verovatnoće su realnovrednosne. U odeljku 8.2.8 ćemo navesti i neke varijante ove logike.

8.2.1 Formalni jezik

Neka je S skup svih racionalnih brojeva iz $[0, 1]$. Jezik $\mathcal{L}(LPP_1)$ logike LPP_1 je prebrojivo proširenje klasičnog iskaznog jezika koje se sastoji od prebrojivog skupa iskaznih slova $\phi = \{p_1, p_2, \ldots\}$, klasičnih operatora $\neg$ i $\wedge$ i liste verovatnosnih operatora oblika $P_{\geq s}$ za svaki $s \in S$.

8.2.2 Formule

Skup F$or(LPP_1)$ formula ovog jezika je najmanji skup koji sadrži iskazna slova i zatvoren je za sledeća pravila:

- Ako je α formula i $s \in S$, onda je $P_{\geq s}\alpha$ formula.

- Ako su α i β formule, formule su i $\neg\alpha$ i $\alpha \wedge \beta$.

Primeri formula su: $P_{\geq 0.5}p_1$ i $P_{\geq 0.3}(p_1 \wedge \neg P_{\geq 0.82}p_2) \wedge p_3$. Formule iz skupa $For(LPP_1)$ ćemo označavati malim slovima grčkog alfabeta: α, β, γ, ...

Za definisanje preostalih klasičnih iskaznih operatora ($\vee$, $\rightarrow$ i $\leftrightarrow$) se koristi standardni postupak, a drugi verovatnosni operatori se uvode definicijama:

- $P_{<s}\alpha =_{def} \neg P_{\geq s}\alpha$,
- $P_{\leq s}\alpha =_{def} P_{\geq 1-s}(\neg\alpha)$,
- $P_{>s}\alpha =_{def} \neg P_{\leq s}\alpha$ i
- $P_{=s}\alpha =_{def} P_{\geq s}(\alpha) \wedge \neg P_{>s}(\alpha)$.

Na primer, formula $(P_{>r}\alpha \wedge P_{=s}(\alpha \rightarrow \beta)) \rightarrow P_{\leq t}\beta$ se shvata kao 'ako je verovatnoća od α veća od r i β sledi iz α sa verovatnoćom s, verovatnoća od β nije veća od t'.

8.2.3 Klase modela

Za davanje značenje formulama koristićemo modele u kojima su verovatnoće definisane nad mogućim svetovima. Ovi modeli podsećaju na Kripkeove modalne modele, ali umesto modalne relacije dostižnosti svakom svetu je pridružen jedan prostor mere. Jednu komponentu tog prostora predstavlja skup svetova za koje možemo smatrati da su u nekom smislu dostižni iz posmatranog sveta, dok je verovatnoća za svaki svet definisana nad podskupovima tih dostižnih svetova. U našem pristupu formule, iako govore o verovatnoći, imaju standardne istinitosne vrednosti tačno ($\top$), odnosno netačno ($\bot$).

Definicija 8.2.1 *Verovatnosni model* je struktura $M = \langle W, v, Prob\rangle$, gde su:

- W – neprazan skup svetova,
- $v : W \times \phi \mapsto \{\top, \bot\}$ – iskazna interpretacija koja svakom svetu i svakom iskaznom slovu pridružuje $\top$ ili $\bot$ i
- $Prob$ – preslikavanje koje svakom svetu $w \in W$ pridružuje jedan konačno-aditivni verovatnosni prostor $\langle W(w), H(w), \mu(w)\rangle$ tako da:
  - $W(w)$ je podskup skupa svih svetova W,
  - $H(w)$ je algebra podskupova od $W(w)$, tj. $W(w) \in H(w)$, ako je $F_1 \in H(w)$, tada je komplement skupa F_1 u odnosu na $W(w)$, $F_1^c \in H(w)$ i ako su $F_1 \in H(w)$ i $F_2 \in H(w)$, tada je $F_1 \cup F_2 \in H(w)$ i
  - $\mu(w)$ je konačno-aditivna verovatnoća definisana na $H(w)$, tj. to je funkcija $\mu(w) : H(w) \rightarrow [0, 1]$ koja ispunjava da je $\mu(w)(W(w)) = 1$ i ako skupovi F_1 i F_2 pripadaju $H(w)$ i međusobno su disjunktni ($F_1 \cap F_2 = \emptyset$), tada je $\mu(F_1 \cup F_2) = \mu(F_1) + \mu(F_2)$.

Relacija zadovoljivosti je relacija između svetova verovatnosnih modela i formula. Za formulu koja je zadovoljena u nekom svetu kaže se i da je tačna ili da važi u tom svetu. Ako formula nije zadovoljena u nekom svetu kaže se i da je netačna ili da ne važi u tom svetu.

Definicija 8.2.2 Neka je M proizvoljan LPP_1-model i w proizvoljan svet tog modela. Formula α je *zadovoljena* u svetu w, u oznaci $w \models_M \alpha$, ako važi:

- ako je α iskazno slovo, $\alpha \in \phi$, $w \models_M \alpha$ ako i samo ako $v(w)(\alpha) = \top$,
- ako je α oblika $P_{\geq s}\beta$, $w \models_M \alpha$ ako i samo ako $\mu(w)(\{w' : w' \in W(w), w' \models_M \beta\}) \geq s$,
- ako je α oblika $\neg\beta$, $w \models_M \alpha$ ako i samo ako nije $w \models_M \beta$ i
- ako je α oblika $\beta \wedge \gamma$, $w \models_M \alpha$ ako i samo ako $w \models_M \beta$ i $w \models_M \gamma$.

Prokomentarišimo slučaj formule oblika $P_{\geq s}\beta$. Prema definiciji, ova formula važi u nekom svetu w modela M ako je mera $\mu(w)$ (iz sveta w) svih dostižnih svetova (koji su u skupu $W(w)$) u kojima važi formula β veća do jednaka sa s.

Indeks M koji označava razmatrani model ćemo izostavljati iz oznake $\models_M$ ako se model podrazumeva. Oznaku $\not\models$ koristićemo sa značenjem: nije $\models$.

Definicija 8.2.3 Formula α je *zadovoljiva* ako postoji svet nekog modela u kome je α zadovoljeno. Formula je *valjana* u modelu M ukoliko je zadovoljena u svakom svetu tog modela. Formula je *valjana* u klasi modela C ukoliko je valjana u svakom modelu te klase. Skup formula T je *zadovoljiv* ako postoji svet nekog modela u kome su zadovoljene sve formule iz skupa.

U daljem izlaganju bavićemo se samo merljivim modelima, odnosno modelima u kojima su svi skupovi svetova definabilni formulama merljivi. Preciznije:

Definicija 8.2.4 LPP_1-model M je *merljiv* ako je za svaki svet w tog modela i svaku formulu α ispunjeno:

$$\{w' \in W(w) : w' \models \alpha\} \in H(w).$$

Klasu merljivih modela označićemo sa: $LPP_{1,\mathrm{M}eas}$.

8.2.4 Aksiomatizacija

Aksiomatski sistem Ax_{LPP_1} za logiku LPP_1 sadrži sledeće shema aksiome:

1. Sve instance iskaznih tautologija.
2. $P_{\geq 0}\alpha$
3. $P_{\leq r}\alpha \rightarrow P_{<s}\alpha$, $s > r$
4. $P_{<s}\alpha \rightarrow P_{\leq s}\alpha$
5. $(P_{\geq r}\alpha \wedge P_{\geq s}\beta \wedge P_{\geq 1}(\neg\alpha \vee \neg\beta)) \rightarrow P_{\geq \min(1,r+s)}(\alpha \vee \beta)$
6. $(P_{\leq r}\alpha \wedge P_{<s}\beta) \rightarrow P_{<r+s}(\alpha \vee \beta)$, $r + s \leq 1$

i pravila izvođenja:

1. Iz α i $\alpha \rightarrow \beta$ izvesti β.

2. Iz α izvesti $P_{\geq 1}\alpha$.

3. Iz $\beta \rightarrow P_{\geq s-\frac{1}{k}}\alpha$, za svaki prirodni broj $k \geq \frac{1}{s}$, izvesti $\beta \rightarrow P_{\geq s}\alpha$.

Uključivanje aksiome 1 i pravila izvođenja 1 u sistem Ax_{LPP_1} obezbeđuje da je klasična iskazna logika podlogika ove logike. Umesto ovog pristupa je bilo moguće koristiti bilo koji potpuni aksiomatski sistem. Aksioma 2 osigurava da je svaka formula zadovoljena u skupu svetova mere bar 0. Zamenom α sa $\neg\alpha$ u aksiomi 2 dobija se formula (2') $P_{\leq 1}\alpha (= P_{\geq 0}\neg\alpha)$ koja znači da je svaka formula zadovoljiva u skupu svetova čija mera nije veća od 1. Aksiome 3 i 4 se mogu zapisati i u obliku: (3') $P_{\geq s}\alpha \rightarrow P_{>r}\alpha$, $s > r$, odnosno (4') $P_{>s}\alpha \rightarrow P_{\geq s}\alpha$. Recimo, za aksiomu 3, $P_{\leq r}\alpha \rightarrow P_{<s}\alpha$ je ekvivalentno sa $\neg P_{<s}\alpha \rightarrow \neg P_{\leq r}\alpha$, što se prema definiciji verovatnosih operatora zapisuje kao $P_{\geq s}\alpha \rightarrow P_{>r}\alpha$. Monotonost mere se iskazuje formulom $P_{\geq r}\alpha \rightarrow P_{\geq s}\alpha$, za $r \geq s$ i predstavlja posledicu aksioma 3 i 4. Aksiome 5 i 6 odgovaraju konačnoj aditivnosti verovatnoće. Na primer, aksiomom 5 se kaže da, ako su skupovi svetova u kojima važe α i β disjunktni, onda je verovatnoća skupa svetova u kojima važi $\alpha \vee \beta$ suma verovatnoća prethodna dva skupa. Pravilo izvođenja 1 je klasično pravilo modus ponensa. Pravilo izvođenja 2 liči na pravilo necesitacije u modalnim logikama. Pravilo izvođenja 3 je jedino beskonačno pravilo u aksiomatskom sistemu. Intuitivno, njime se kaže da, ako je verovatnoće formule α proizvoljno blizu nekom racionalnom broju s, onda je ona upravo jednaka sa s.

Definicija 8.2.5 Formula α je *teorema* aksiomatskog sistema Ax_{LPP_1} ($\vdash_{Ax_{LPP_1}} \alpha$) ako postoji najviše prebrojiv niz formula $\alpha_0, \alpha_1, \ldots, \alpha$, takav da je svaka α_i aksioma ili je pomoću nekog od pravila izvođenja izvedena iz prethodnih formula. Niz formula $\alpha_0, \alpha_1, \ldots, \alpha$, je *dokaz* (*izvođenje*) za α.

U dokazu teoreme potpunosti biće korišteni i pojmovi sintaksne posledice, konzistentnosti i maksimalnog skupa. Dok se poslednja dva pojma definišu kao i ranije u definiciji 3.6.6, prvi pojam, kao i pojam dokaza u definiciji 8.2.5, zahteva definiciju u kojoj se dozvoljavaju i beskonačni nizovi formula.

Definicija 8.2.6 Formula α je *sintaksna posledica skupa formula* T ($T \vdash_{Ax_{LPP_1}} \alpha$) ako postoji najviše prebrojiv niz formula $\alpha_0, \alpha_1, \ldots, \alpha$, takav da je svaka formula α_i aksioma ili pripada skupu T ili je pomoću nekog od pravila izvođenja izvedena iz prethodnih formula, uz ograničenje da se pravilo 2 može primeniti samo na teoreme. Niz formula je $\alpha_0, \alpha_1, \ldots, \alpha$ je *dokaz* (*izvođenje*) za α iz skupa formula T.

Indeks Ax_{LPP_1} u oznakama $\vdash_{Ax_{LPP_1}}$ i $T \vdash_{Ax_{LPP_1}}$ nećemo navoditi ukoliko to ne preti da izazove zabunu. Oznaku $\nvdash$ koristićemo sa značenjem: nije $\vdash$.

8.2.5 Korektnost i potpunost askiomatskog sistema

Teorema 8.2.7 (Korektnost) Aksiomatski sistem Ax_{LPP_1} je korektan u odnosu na klasu modela $LPP_{1,\text{Meas}}$.

Dokaz. Dokaz korektnosti se sprovodi tako što se pokaže da je svaki primerak aksioma valjana formula i da pravila izvođenja očuvaju valjanost. Neka je M proizvoljni model. Svaki svet modela se sa stanovišta klasičnih iskaznih formula

može shvatiti kao jedan klasičan iskazni model, tako da u svakom svetu važe sve klasične iskazne tautologije. Aksiome 2 – 6 se odnose na osobine verovatnoća i očigledno važe u proizvoljnom svetu. Pretpostavimo da su α i $\alpha \to \beta$ valjane formule. Ako formula β ne bi bila valjana postojao bi svet w nekog modela M u kome bi bilo $w \not\models \beta$ i $w \models \alpha \to \beta$, ali bi tada moralo biti i $w \not\models \alpha$, odakle ni α ne bi bilo valjano. Dakle, pravilo izvođenja 1 očuvava valjanost. U vezi pravila 2 pretpostavimo da je α valjana formula. Tada za svaki svet w svakog modela M važi $(\forall w' \in W(w))w' \models \alpha$. Kako je $\mu(w)(W(w)) = 1$, znači i da $w \models P_{\geq 1}\alpha$, odnosno da $P_{\geq 1}\alpha$ važi u svakom svetu svakog modela, tj. da se polazeći od valjane pretpostavke primenom pravila dobio valjan zaključak. Konačno, pravilo 3 očuvava valjanost zbog osobine skupa realnih brojeva. Ξ

U dokazivanju potpunosti se koristi postupak sličan dokazu potpunosti za klasičnu iskaznu logiku (teorema 3.6.14). Najpre ćemo iskazati jedno pomoćno tvrđenje:

Teorema 8.2.8 Neka su T skup formula i α i β formule.

1. (Teorema dedukcije) Ako je $T \cup \{\alpha\} \vdash \beta$, onda $T \vdash \alpha \to \beta$.
2. $\vdash P_{\geq s}\alpha \to P_{\geq s}(\alpha \vee \bot)$.
3. $\vdash P_{\geq s}(\alpha \vee \bot) \to P_{\geq s}\neg\neg\alpha$.
4. $\vdash P_{\geq s}\alpha \to P_{\geq s}\neg\neg\alpha$.
5. $\vdash P_{\geq 1}(\alpha \to \beta) \to (P_{\geq s}\alpha \to P_{\geq s}\beta)$.
6. Ako je $\vdash \alpha \leftrightarrow \beta$, onda je $\vdash P_{\geq s}\alpha \leftrightarrow P_{\geq s}\beta$.
7. $\vdash P_{\geq r}\alpha \to P_{\geq s}\alpha$, $r > s$.

Sledeći korak dokaza se odnosi na konstrukciju maksimalno konzistentnog proširenja konzistentnog skupa.

Teorema 8.2.9 Svaki konzistentan skup formula T se može proširiti do maksimalno konzistentog skupa.

Dokaz. U dokazu se koristi sledeća konstrukcija. Neka je $\alpha_0, \alpha_1, \ldots$ jedno nabrajanje svih formula. Definisaćemo niz skupova T_i, $i = 0, 1, 2, \ldots$ tako da je:

1. $T_0 = T$,
2. Za svaki $i \geq 0$, ako je $T_i \cup \{\alpha_i\}$ konzistentan, onda je $T_{i+1} = T_i \cup \{\alpha_i\}$.
3. Za svaki $i \geq 0$, ako $T_i \cup \{\alpha_i\}$ nije konzistentan, onda $T_{i+1} = T_i \cup \{\neg\alpha_i\}$.
4. Ako je skup T_{i+1} dobijen dodavanjem formule oblika $\neg(\beta \to P_{\geq s}\gamma)$, tada za neki $n \in \mathbb{N}$ skupu dodajemo i formulu $\beta \to \neg P_{\geq s-\frac{1}{n}}\gamma$, tako da T_{i+1} bude konzistentno[1].

[1]Formula $\beta \to \neg P_{\geq s-\frac{1}{n}}\gamma$ je svojevrsni svedok čije prisustvo u skupu T_{i+1} garantuje da skup $T_i \cup \{\beta \to P_{\geq s}\gamma\}$ nije konzistentan.

5. $\overline{T} = \cup_i T_i$.

Zatim se pokazuje da su skupovi dobijeni koracima 1, 2 i 3 konzistentni, a da isto važi i za korak 143 jer iz pretpostavke $T_i, \neg(\beta \rightarrow P_{\geq s}\gamma), \beta \rightarrow \neg P_{\geq s-\frac{1}{k}}\gamma \vdash \bot$, za svaki prirodan broj $k > \frac{1}{s}$, primenom pravila izvođenja 3, sledi $T_i \vdash \beta \rightarrow P_{\geq s}\gamma$ što je suprotno pretpostavci o konzistentnosti skupa T_i i nekonzistentnosti skupa $T_i \cup \{\beta \rightarrow P_{\geq s}\gamma\}$. Koraci 2 i 3 garantuju da je skup $\overline{T}$ maksimalan. Konačno, može se pokazati da je skup $\overline{T}$ deduktivno zatvoren (sadrži sve svoje posledice), a da ne sadrži sve formule, zbog čega je konzistentan. Ξ

Dalje ćemo, pomoću maksimalno konzistentnih skupova, definisati model proizvoljnog konzistentnog skupa formula. Neka je struktura $M = \langle W, v, Prob\rangle$ definisana na sledeći način:

- W je skup svih maksimalno konzistentnih skupova,
- v je preslikavanje koje svakom svetu $w \in W$ pridružuje interpretaciju $v(w) : \phi \mapsto \{\top, \bot\}$, tako da za svako iskazno slovo $p \in \phi$, $v(w)(p) = \top$ ako i samo ako $p \in w$,
- Za svako $w \in W$, $Prob(w) = \langle W(w), H(w), \mu(w)\rangle$ tako da:
  - $W(w) = W$,
  - $H(w)$ je klasa skupova oblika $[\alpha] = \{w \in W : \alpha \in w\}$, za svaku formulu α i
  - za svaki skup $[\alpha] \in H(w)$, $\mu(w)([\alpha]) = \sup\{r : P_{\geq r}\alpha \in w\}$.

Teorema 8.2.10 Neka je $M = \langle W, v, Prob\rangle$ upravo definisana struktura. Tada je za svaki $w \in W$, klasa $H(w) = \{[\alpha]\}$ algebra podskupova od $W(w)$.

Dokaz. Očigledno važi sledeće:

- $W(w) = [\alpha \vee \neg\alpha] \in H(w)$, za proizvoljnu formulu α,
- ako je $[\alpha] \in H(w)$, tada je $[\neg\alpha]$ komplement skupa $[\alpha]$ i taj komplement pripada $H(w)$ i
- ako su $[\alpha_1], [\alpha_2] \in H(w)$, tada je unija $[\alpha_1] \cup [\alpha_2] \in H(w)$ jer je $[\alpha_1] \cup [\alpha_2] = [\alpha_1 \vee \alpha_2]$.

Dakle, za svaki w, $H(w)$ je algebra podskupova od $W(w)$. Ξ

Naredna teorema, koja se direktno dokazuje primenom aksioma i pravila sistema Ax_{LPP_1}, pokazuje da je struktura M jedan $LPP_{1,\text{Meas}}$-model.

Teorema 8.2.11 Neka je $M = \langle W, v, Prob\rangle$ upravo definisana struktura. Tada za svaki $w \in W$ važi:

1. Ako je $[\alpha] = [\beta]$, onda je $\mu(w)([\alpha]) = \mu(w)([\beta])$.
2. $\mu(w)([\alpha]) \geq 0$.

3. $\mu(w)([\alpha]) = 1 - \mu(w)([\neg\alpha])$.

4. $\mu(w)([\alpha] \cup [\beta]) = \mu(w)([\alpha]) + \mu(w)([\beta])$, za sve formule α i β takve da je $[\alpha] \cap [\beta] = \emptyset$.

Konačno, možemo dokazati sledeću teoremu:

Teorema 8.2.12 (Proširena kompletnost) Skup formula T je konzistentan ako i samo ako je zadovoljiv.

Dokaz. Razmotrićemo samo smer ($\Rightarrow$). Neka je T konzistentan skup formula. U pethodnim teoremama je pokazano da je $M = \langle W, v, Prob\rangle$ jedan $LPP_{1,\text{Meas}}$-model. Indukcijom po složenosti formula dokazuje se da za svaku formulu α i svaki svet $w \in W$, $w \models \alpha$ ako i samo ako $\alpha \in w$. Odatle, skup T je zadovoljiv pošto postoji bar jedno njegovo maksimalno konzistentno proširenje koje je svet modela M.

Neka je α iskazno slovo. Tada tvrđenje važi prema definiciji modela M. Neka je $\alpha = P_{\geq s}\beta$. Ako $\alpha \in w$, $\sup\{r : P_{\geq r}(\beta) \in w\} = \mu(w)([\beta]) \geq s$ i $w \models P_{\geq s}\beta$. U suprotnom smeru, pretpostavimo da je $w \models P_{\geq s}\beta$, tj. $\sup\{r : P_{\geq r}(\beta) \in w\} \geq s$. Ako $\mu(w)([\beta]) > s$, onda zbog osobina supremuma i monotonosti verovatnoće $\mu(w)$, $P_{\geq s}\beta \in w$. Ako $\mu(w)([\beta]) = s$, prema pravilu 3, $w \vdash P_{\geq r}\alpha$ i pošto je w deduktivno zatvoren skup, $P_{\geq r}\alpha \in w$. Ostali slučajevi se rešavaju kao i u teoremi 3.6.14. Ξ

8.2.6 Komentar dokaza teoreme potpunosti

Verovatnosna logika LPP_1 je karakteristična po tome da za nju ne važi teorema kompaktnosti. Naime, razmotrimo skup formula

$$T = \{P_{<\frac{1}{n}}\alpha : n \in \mathbb{N}\} \cup \{\neg P_{=0}\alpha\}.$$

Da bi se pokazalo da je svaki konačan podskup T_0 ovog skupa zadovoljiv, dovoljno je posmatrati dvočlani model u kome je skup svetova $W = \{w_1, w_2\}$, pri čemu je $w_1 \not\models \alpha$ i $w_2 \models \alpha$, a u svetu w_1 definisati

$$\mu(w_1)(\{w' : w' \in W(w_1), w' \models_M \alpha\}) = \mu(w_1)(\{w_2\}) = \frac{1}{2}\min\{\frac{1}{n} : P_{<\frac{1}{n}}\alpha \in T_0\}$$

Međutim, ceo skup T nije zadovoljiv pošto za bilo koju nenultu meru skupa svetova u kojima važi α, $\mu(w)(\{w' : w' \in W(w), w' \models_M \alpha\}) = c$, postoji $n \in \mathbb{N}$ tako da je $\frac{1}{n} < c$.

Ovo ima za posledicu da ne postoji konačna aksiomatizacija za koju važi korektnost i proširena kompletnost (mada u ovom slučaju postoji konačna aksiomatizacija za koju važi obična kompletnost), što se pokazuje na sledeći način. Pretpostavimo da postoji konačna korektna aksiomatizacija za logiku LPP_1 za koju se može dokazati proširena kompletnost. Pretpostavimo da je svaki konačan podskup beskonačnog skupa T formula zadovoljiv, dok skup T to nije. Zbog potpunosti aksiomatskog sistema, skup T nije konzistentan, pa se iz ovog skupa formula izvodi kontradikcija. Pošto je aksiomatski sistem konačan, to izvođenje je konačno. Zato

postoji konačan podskup T_0 skupa T sa istom osobinom, tj. iz T_0 se izvodi kontradikcija. Odatle skup T_0 nije konzistentan. Zbog korektnosti aksiomatskog sistema T_0, suprotno pretpostavci, ne bi bio zadovoljiv.

U takvoj situaciji uvođenje beskonačnog pravila i beskonačnih dokaza je cena koju moramo platiti da bismo izbegli logički neprijatnu situaciji da postoje konzistetni skupovi koji nisu zadovoljivi. Druga mogućnost je da se ograniče domeni verovatnoća, tako da budu konačni, u kom slučaju bi važila teorema kompaktnosti. Primeri takvih logika biće spomenuti u odeljku 8.2.8.

8.2.7 Odlučivost

U dokazu odlučivosti problema zadovoljivosti i valjanosti za razmatranu klasu modela koristićemo se postupcima filtracije kojim se razmatranje proizvoljnih modela u ispitivanju zadovoljivosti svodi na ispitivanje konačnih modela i transformisanjem verovatnosnih formula u sisteme linearnih jednačina i nejednačina.

Teorema 8.2.13 Formula α je $LPP_{1,\mathrm{Meas}}$-zadovoljiva ako i samo ako je zadovoljiva u nekom konačnom $LPP_{1,\mathrm{Meas}}$-modelu.

Dokaz. Neka je $M = \langle W, v, Prob\rangle$ jedan $LPP_{1,\mathrm{Meas}}$-model u čijem svetu w važi formula α. Postupkom filtracije transformisaćemo model M u konačan model u čijem jednom svetu će α takođe važiti.

Neka $\mathrm{S}ubfor(\alpha)$ označava skup svih potformula od α. Definisaćemo jednu relaciju ekvivalencije $\approx$ na skupu svetova W takvu da $w \approx u$ ako i samo ako za svaku formulu $\beta \in \mathrm{S}ubfor(\alpha)$, $w \models \beta$ ako i samo ako $u \models \beta$. Pošto je skup $\mathrm{S}ubfor(\alpha)$ konačan, skup svetova W je relacijom $\approx$ podeljen u konačan broj klasa ekvivalencije. Sa C_u označimo klasu ekvivalencije kojoj pripada svet u. Filtracija modela M skupom $\mathrm{S}ubfor(\alpha)$ je model $M^* = \langle W^*, v^*, Prob^*\rangle$ za koji je:

- W^* sadrži tačno po jedan svet iz svake od klasa ekvivalencije količničkog skupa $W_{/\approx}$,
- za svaki $w \in W^*$, $v^*(w)(p) = v(w)(p)$,
- za svaki $w \in W^*$, $W^*(w) = \{u : (\exists v \in C_u) v \in W(w)\}$,
- za svaki $w \in W^*$, $H^*(w) = 2^{W^*(w)}$,
- za svaki $w \in W^*$, $\mu^*(w)(u) = \mu(w)(C_u \cap W(w))$, dok je za svaki $D \in H^*(w)$, $\mu^*(w)(D) = \sum_{u\in D} \mu(w)(C_u \cap W(w))$.

Pošto je

$$\mu^*(w)(W^*(w)) = \sum_{u\in W^*(w)} \mu(w)(C_u \cap W(w)) = \sum_{C_u \in W_{/\approx}} \mu(w)(C_u \cap W(w)) = 1$$

$\mu^*(w)$ jeste verovatnoća. Prema načinu definisanja modela M^*, ovaj model jeste $LPP_{1,\mathrm{Meas}}$-model. Indukcijom po složenosti formula iz skupa $\mathrm{S}ubfor(\alpha)$ pokazuje se da za svaki svet $w \in W^*$ i svaku formulu $\beta \in \mathrm{S}ubfor(\alpha)$, $w \models_M \beta$ ako i samo ako $w \models_{M^*} \beta$, čime se dokazuje polazno tvrđenje. Ξ

Primetimo da postupak filtracije nije dovoljan za dokazivanje odlučivosti, pošto zbog verovatnosne komponente u $LPP_{1,\text{Meas}}$-modelima, broj modela sa konačnim brojem svetova (≥ 2) nije konačan. Zato se u poslednjem koraku dokaza koristi prevođenje formula u sistem linearnih jednačina i nejednačina.

Teorema 8.2.14 Problemi zadovoljivosti i valjanosti za $LPP_{1,\text{Meas}}$ su odlučivi.

Dokaz. Neka je $M = \langle W, v, Prob\rangle$ jedan $LPP_{1,\text{Meas}}$-model u čijem svetu w važi formula α i neka $Subfor(\alpha)$ označava skup svih potformula od α. U svakom svetu w važi tačno jedna formula oblika

$$\beta_1 \wedge \ldots \wedge \beta_n \wedge \neg\gamma_1 \wedge \ldots \wedge \neg\gamma_m$$

gde je $Subfor(\alpha) = \{\beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_m\}$. Nazovimo tu formulu karakteristična formula sveta w. Neka je $k = n+m$. Postupkom filtracije obezbeđeno je da postoji $LPP_{1,\text{Meas}}$-model sa najviše 2^k svetova, tako da formula α važi u bar jednom svetu tog modela. Za svaki prirodan broj $l \leq 2^k$ razmotrićemo modele sa l svetova. U svakom od tih svetova važiće tačno jedna karakteristična formula. Dakle, za svaki l razmotrićemo sve skupove od l karakterističnih formula koje su iskazno nekontradiktorne i od kojih bar jedna formula sadrži polaznu formulu α. Zapravo, ovde je reč o multiskupovima pošto se ista karakteristična formula može pojaviti više puta. Ti skupovi u potpunosti opisuju model. Za svaki takav izbor i za svaki svet w_i (tj. odgovarajuću karakterističnu formulu α_i) posmatraćemo sledeći skup linearnih jednačina i nejednačina ($\beta \in \alpha_j$ označava da je β konjunkt u α_j):

$\sum_{j=1}^{l} \mu(w_i)(w_j) = 1$

$\mu(w_i)(w_j) \geq 0$, za sve $j = 1, l$

$\sum_{w_j:\beta\in\alpha_j} \mu(w_i)(w_j) \geq r$, ako $P_{\geq r}\beta \in \alpha_i$

$\sum_{w_j:\beta\in\alpha_j} \mu(w_i)(w_j) < r$, ako $\neg P_{\geq r}\beta \in \alpha_i$

Prva jednačina znači da je za svaki svet w_i verovatnoća skupa svih svetova jednaka 1, a nejednačine iz drugog reda da su verovatnoće svetova nenegativne. Trećom, odnosno četvrtom, grupom nejednačina tvrdi se da su sve verovatnosne formule oblika $P_{\geq r}\beta$, odnosno $\neg P_{\geq r}\beta$, koje se javljaju kao konjunkcije u karakterističnoj formuli α_i zadovoljene. Unija ovih linearnih jednačina i nejednačina predstavlja jedan sistem linearnih jednačina i nejednačina čije rešavanje je odlučiv problem. Ako je sistem za fiksirane l i izbor karakterističnih formula zadovoljiv, u svakom od l svetova se mogu definisati verovatnosni prostori i u bar jednom svetu važi polazna formula α, odnosno α je $LPP_{1,\text{Meas}}$-zadovoljiva. Ako za neke fiksirane l i izbor karakterističnih formula odgovarajući sistem nije rešiv, tada ne postoji model sa najviše l svetova takav da u i-tom svetu važi karakteristična formula α_i. Ako ni za koje fiksirane l i izbor karakterističnih formula odgovarajući sistem nije rešiv, prema teoremi 8.2.13 formula α nije $LPP_{1,\text{Meas}}$-zadovoljiva.

Kako se opisanim postupkom razmatra samo konačno mnogo sistema linearnih jednačina i nejednačina, problem zadovoljivosti je odlučiv, a pošto je neka formula α $LPP_{1,\text{Meas}}$-valjana ako i samo ako $\neg\alpha$ nije $LPP_{1,\text{Meas}}$-zadovoljiva, odlučiv je i problem valjanosti. Ξ

Pokazuje se da je problem zadovoljivosti u logici LPP_1 kompletan problem u klasi složenosti $PSPACE$.

8.2.8 Varijante logike LPP_1

Postoji više varijanti verovatnosne logike LPP_1 koje se razmatraju zbog određenih pogodnosti. Na primer, logika $LPP_1^{\mathrm{FR}(n)}$, predstavlja restrikciju logike LPP_1 u kojoj su dozvoljene samo verovatnoće sa konačnim unapred fiksiranim rangom, zbog čega, kako je ranije rečeno, ima konačnu aksiomatizaciju za koju važi proširena kompaktnost. U logici LPP_2, nisu dozvoljene iteracije verovatnoća i mešanje iskaznih i verovatnosnih formula. Zbog toga se u ovoj logici može jedino govoriti o verovatnoći događaja, a ne i o verovatnoći verovatnoće. Međutim, ovo smanjenje izražajnosti omogućava da se složenost problema zadovoljivosti svede na nivo NP-kompletnosti. Konačno, logika $LPP_2^{\mathrm{FR}(n)}$, je restrikcija logike LPP_2 u kojoj su dozvoljene samo verovatnoće sa konačnim unapred fiksiranim rangom.

U logici koju su uveli Fagin, Halpern i Megiddo u formulama su dozvoljene linearne kombinacije verovatnoća, na primer jedna formula je oblika $a_1 w(\alpha_1) + \ldots + a_k w(\alpha_k) \geq c$, gde su a_i i c racionalni brojevi, dok se $w(\alpha_i)$ interpretira kao verovatnoća od α_i. Za ovu logiku je data konačna aksiomatizacija, no kako je ona proširenje logike LPP_1, pri takvoj akiomatizaciji se takođe javljaju problemi sa kompaktnošću o kojima je bilo reči u odeljku 8.2.6.

Verovatnosnim logikama sa realnovrednosnim merama mogu se dodati i unarni verovatnosni operatori oblika Q_F sa značenjem 'verovatnoća pripada skupu F', gde je F rekurzivan racionalni podskup od [0,1]. Ovakvi operatori, u opštem slučaju, nisu uzajamno definabilni sa operatorima $P_{\geq s}$, što znači da se njihovim uvođenjem povećava izražajnost logike. Međutim, pojedini izbori familija skupova koji se javljaju u operatorima mogu dovesti do toga da čak i iskazna verovatnosna logika koja ih dopušta postane neodlučiva. Drugo moguće proširenje verovatnosne logike predstavljalo bi uvođenje binarnog operatora $\preceq$ takvog da formula $\alpha \preceq \beta$ ima smisao 'β je verovatnije od α'. Ovim bi se omogućilo kvalitativno rezonovanje o verovatnoći, nasuprot dosadašnjem striktno kvantitativnom rezonovanju na osnovu numeričkih vrednosti verovatnoća.

U literaturi koja se odnosi na verovatnosne logike je konstatovano da formula $\alpha \rightarrow \beta$ nekada ne modelira verno smisao pravila oblika 'ako je α, onda je β' koja se javljaju u bazama znanja. Objašnjenje za to je sledeće: dok je formula $\alpha \rightarrow \beta$ tačna i kada je α netačno, pravilo se odnosi na one situacije u kojima se podrazumeva da α važi. Zbog toga, verovatnoća formule $\alpha \rightarrow \beta$ može biti jako visoka, iako je verovatnoća formule $\alpha \wedge \beta$ niska. Sa druge strane, uslovna verovatnoća ima upravo željeni smisao, pa je u ovakvim situacijama pogodno proširiti razmatranu verovatnosnu logiku tako da dozvoli i rezonovanje o uslovnim verovatnoćama. To se postiže dodavanjem odgovarajućih binarnih operatora uslovne verovatnoće, tako da formule oblika $CP_{\geq s}(\beta|\alpha)$ i $CP_{\leq s}(\beta|\alpha)$ znače 'uslovna verovatnoća od β pod uslovom α je veća (odnosno, manja) do jednaka od s'. I za ovakvu logiku se mogu dokazati teoreme potpunosti i odlučivosti analogne odgovarajućim tvrđenjima za logiku LPP_1. Uvođenje uslovnih verovatnoća dopušta nam još jednu interesantnu primenu verovatnosnih logika. Naime, ako pored realnih vrednosti iz $[0, 1]$, verovatnoće mogu biti budu i pozitive beskonačno male vrednosti, takozvane *infinitezimale* (pozitivna infinitezimala je broj veći od 0, a ipak manji od svakog pozitivnog racionalnog broja), onda se u okviru verovatnosnih logika može modelirati i nemonotono zaključivanje. Preciznije, formula oblika $CP_{\approx 1}(\beta|\alpha)$ bi imala smisao 'uslovna vero-

vatnoća od β pod uslovom α je približno 1' i modelirala bi iskaz tipa 'ako važi α, po pravilu važi i β' o kojem je bilo reči u odeljku 7.4.1.

U variranju kodomena verovatnoće se može ići i dalje. Na primer, kodomeni mere kojom merimo neizvesnost mogu biti parcijalno uređeni skupovi, poput skupova n-torki ili nizova racionalnih brojeva, skup prirodnih brojeva koji uključuje i beskonačnost itd. Za neke od tih slučajeva rešeni su problemi kompletne aksiomatizacije i odlučivosti.

Naredna vrsta proširenja verovatnosne logike LPP_1 se postiže njenim kombinovanjem sa različitim modalnim logikama. Na primer, u kombinaciji logike LPP_1 sa diskretnom, linearnom temporalnom logikom, moguće je modelirati iskaze koji govore o promeni verovatnoće sa protokom vremena. Na primer, formula

$$\neg\alpha \rightarrow (P_{\geq s}\alpha \rightarrow \bigcirc P_{\geq s}\alpha)$$

bi karakterisala situacije u kojima verovatnoća od α ne opada, ako se α nije dogodilo.

U prethodnim varijantama verovatnosne logike LPP_1 podrazumevalo se da je bazna logika sa kojom se radi klasična logika. Naime, instance klasičnih iskaznih tautologija su bile deo aksiomatskog sistema, dok je posmatrano sa semantičke strane relacija zadovoljivosti u verovatnosnim modelima bila proširenje istoimene relacije za klasičnu logiku. Međutim, pošto je - kao što je rečeno i u poglavlju 7 - klasična logika ponekad neadekvatna za modeliranje realnih situacija, razmatraju se i verovatnosne logike u kojima se kao bazna koristi neka od neklasičnih logika, recimo intuistička. Imajući ovo u vidu razmotrimo sledeći primer.

Primer 8.2.15 Neka iskazna slova p i q redom označavaju iskaze 'pada kiša' i 'uređaj za zalivanje je uključen'. Smisleno je zahtevati da uređaj za zalivanje bude isključen kada pada kiša, odnosno da verovatnoća formule $p \rightarrow q$ bude mala. Sa druge strane, pošto je $(p \rightarrow q) \vee (q \rightarrow p)$ klasična iskazna tautologija, njena verovatnoća mora biti jednaka 1. To dovodi do paradoksalne situacije da verovatnoća formule $q \rightarrow p$ koju shvatamo kao 'ako je uređaj za zalivanje uključen, onda pada kiša' mora biti jako visoka (jer je verovatnoća od $(p \rightarrow q) \vee (q \rightarrow p)$ manja do jednaka od zbira verovatnoća formula $p \rightarrow q$ i $q \rightarrow p$). Ξ

Ovde možemo primetiti da formula $(p \rightarrow q) \vee (q \rightarrow p)$ nije intuicionistički valjana i da njena verovatnoća ne mora biti jednaka 1, odakle se prethodni paradoks eliminiše, i na taj način bolje modelira realna situacija, ako kao baznu koristimo intuicionističku logiku.

8.3 Predikatska verovatnosna logika prvog reda

Slično iskaznim logikama, moguće je posmatrati i predikatske verovatnosne logike prvog reda koje ćemo, po analogiji, označavati sa $LFOP_1$, $LFOP_1^{\mathrm{F}R(n)}$, $LFOP_2$ i $LFOP_2^{\mathrm{F}R(n)}$, uz iste vrste ograničenja kao i u iskaznom slučaju, smatrajući logiku $LFOP_1$ ponovo kao osnovnu. Međutim, situacija u slučaju predikatskih verovatnosnih logika prvog reda je nešto drugačija nego u iskaznom slučaju. Naime, za logiku $LFOP_1$ (kao i za $LFOP_2$) ne samo da ne važi odlučivost (pošto se radi

o logici prvog reda) i nepostojanje konačne aksiomatizacije za koju se dokazuje proširena kompletnost, već se pokazuje da za ovu logiku ne postoji ni rekurzivna, korektna i kompletna aksiomatizacija. U ovom slučaju beskonačna aksiomatizacijia nije alternativa, već nužnost.

U nastavku ćemo ukratko opisati jezik, modele i aksiomatizaciju logike $LFOP_1$.

Jezik logike $LFOP_1$ je predikatski jezik prvog reda proširen skupom verovatnosnih operatora $P_{\geq s}$, za svaki racionalni broj $s \in [0,1]$, dok se termi i formule definišu u skladu sa uobičajenim pravilima, kao u odeljku 5.2.

Definicija 8.3.1 *Verovatnosni model* je struktura $M = \langle W, D, I, Prob\rangle$ gde:

- W je neprazan skup svetova,
- D je preslikavanje koje svakom svetu $w \in W$ pridružuje neprazan domen $D(w)$,
- I je preslikavanje koje svakom svetu $w \in W$ pridružuje klasičnu interpretaciju $I(w)$ prvog reda tako da:
  - za svako i i svaki simbol konstante F_i^0, $I(w)(F_i^0)$ je element skupa $D(w)$,
  - za svako i i k, $I(w)(F_i^k)$ je funkcija iz $D(w)^k$ u $D(w)$,
  - za svako i i k, $I(w)(P_i^k)$ je k-arna relacija nad $D(w)$, tj. podskup od $D(w)^k$,
- $Prob$ je preslikavanje koje svakom svetu $w \in W$ pridružuje jedan konačno-aditivni verovatnosni prostor $\langle W(w), H(w), \mu(w)\rangle$ tako da:
  - $W(w)$ je podskup skupa svih svetova W,
  - $H(w)$ je algebra podskupova od $W(w)$ i
  - $\mu(w)$ je konačno-aditivna verovatnoća definisana na $H(w)$.

Model je sa *fiksiranim domenom* ako je za sve svetove w i u, $D(w) = D(u)$. Model je sa *rigidnim termima* ako je za sve svetove w i u, i svaki funkcijski simbol F, $I(w)(F) = I(u)(F)$.

Definicija 8.3.2 Neka je $M = \langle W, D, I, Prob\rangle$ jedan $LFOP_1$-model. *Valuacija* v pridružuje svakom paru koji se sastoji od sveta $w \in W$ i promenljive x, element domena $D(w)$, tj. $v(w)(x) \in D(w)$. Ako je $w \in W$, $d \in D(w)$ i v valuacija, sa $v_w[d/x]$ označićemo valuaciju koja se sa v poklapa svuda, sem možda za par w, x, pri čemu je $v_w[d/x](w)(x) = d$.

Definicija 8.3.3 Za dati $LFOP_1$-model $M = \langle W, D, I, Prob\rangle$ i valuaciju v, *vrednost terma* t u svetu w ($I(w)(t)_v$) je:

- ako je t promenljiva x, onda je $I(w)(x)_v = v(w)(x)$,
- ako je t simbol konstante, onda je $I(w)(t)_v = I(w)(t)$ i
- ako je $t = F_i^m(t_1, \ldots, t_m)$, $(m > 0)$, onda je $I(w)(t)_v = I(w)(F_i^m)(I(w)(t_1)_v, \ldots, I(w)(t_m)_v)$

Definicija 8.3.4 Za dati $LFOP_1$-model $M = \langle W, D, I, Prob\rangle$ i valuaciju v, *vrednost formule* α u svetu w $(I(w)(\alpha)_v)$ je:

- ako je $\alpha = P_i^m(t_1, \ldots, t_m)$, onda je $I(w)(\alpha)_v = \top$ ako je $\langle I(w)(t_1)_v, \ldots, I(w)(t_m)_v\rangle \in I(w)(P_i^m)$, inače je $I(w)(\alpha)_v = \bot$,
- ako je $\alpha = \neg\beta$, onda je $I(w)(\alpha)_v = \top$ ako je $I(w)(\beta)_v = \bot$, inače je $I(w)(\alpha)_v = \bot$,
- ako je $\alpha = P_{\geq s}\beta$, onda je $I(w)(\alpha)_v = \top$ ako je $\mu(w)\{u \in W(w) : I(u)(\beta)_v = \top\} \geq s$, inače je $I(w)(\alpha)_v = \bot$,
- ako je $\alpha = \beta \wedge \gamma$, onda je $I(w)(\alpha)_v = \top$ ako je $I(w)(\beta)_v = \top$ i $I(w)(\gamma)_v = \top$, inače je $I(w)(\alpha)_v = \bot$ i
- ako je $\alpha = (\forall x)\beta$, onda je $I(w)(\alpha)_v = \top$ ako za svaki $d \in D(w)$,$I(w)(\beta)_{v_w[d/x]} = \top$, inače je $I(w)(\alpha)_v = \bot$.

U sledećem primeru ilustrovaćemo prethodne definicije.

Primer 8.3.5 Razmotrimo formulu $P_{\geq s}P_1^1(x)$ i pretpostavimo da je za neki model $M = \langle W, D, I, Prob\rangle$ i svet $w \in W$ formula tačna u w: $w \models P_{\geq s}P_1^1(x)$. Prema definiciji 8.3.4 ovo važi ako i samo ako za svaku valuaciju v, $I(w)(P_{\geq s}P_1^1(x))_v = \top$, ako i samo ako za svaku valuaciju v, $\mu(w)\{u \in W(w) : I(u)(P_1^1(x))_v = \top\} \geq s$, odnosno, ako i samo ako je mera svetova $w' \in W(w)$ takvih da kakva god da je vrednost $d = v(w)(x)$, važi $w' \models P_1^1(d)$, veća do jednaka sa s. Očigledno je da je prema definicijama formula $P_{\geq s}P_1^1(x)$ tačna u nekom svetu ako i samo ako je tačna i formula $(\forall x)P_{\geq s}P_1^1(x)$. Sa druge strane, tačnost formule $P_{\geq s}P_1^1(x)$ ne implicira tačnost formule $P_{\geq s}(\forall x)P_1^1(x)$. Recimo, za $s = 1$, razmotrimo model M sa fiksiranim domenom:

- W je porebrojiv skup $\{w_0, w_1, w_2, \ldots\}$
- D je prebrojiv skup $\{d_0, d_1, d_2, \ldots\}$,
- za svaki $i \geq 0$, $w_i \models P_1^1(d_j)$ ako i samo ako $i \neq j$,
- algebra $H(w_0)$ sadrži sve konačne podskupove od W i sve ko-konačne podskupove od W,
- $\mu(w_0)$ je konačno aditivna verovatnoća za koju je $\mu(w_0)(A) = 0$ za svaki konačan skup $A \in H(w_0)$, $\mu(w_0)(A) = 1$ za svaki ko-konačan skup $A \in H(w_0)$.

Očigledno, za svaki $w_i \in W$, $w_i \not\models (\forall x)P_1^1(x)$ i $\mu(w_0)(\{w : w \models (\forall x)P_1^1(x)\}) = 0$. Sa druge strane, za svaki $d \in D$ je skup $\{w : w \models P_1^1(d)\}$ ko-konačan, pa je $\mu(w_0)(\{w : w \models P_1^1(d)\}) = 1$. Sledi da je $w_0 \models (\forall x)P_{\geq 1}P_1^1(x)$ i $w_0 \not\models P_{\geq 1}(\forall x)P_1^1(x)$. Odatle, formula ne važi u w_0. Ξ

Kao i malopre u slučaju iskazne verovatnosne logike, i ovde će nas interesovati merljivi modeli, sa dodatkom da su modeli rigidni i fiksiranih domena. Aksiomatizacija te klase modela se ostvaruje jednostavnim kombinovanjem aksiomatskih sistema za klasičnu logiku prvog reda i sistema Ax_{LPP_1} iz odeljka 8.2.4. Naredni primeri ilustruju zašto je bilo potrebno uvesti zahteve za rigidnost i fiksiranost domena.

Primer 8.3.6 Neka je $LFOP_1$-model $M = \langle W, D, I, Prob\rangle$ takav da ne važi rigidnost terma i neka je $w \in W$. Razmotrimo istinitosnu vrednost formule $\alpha(t/x)$ oblika $P_{\geq s}\beta(t/x)$. Prema definiciji istinitosne vrednosti formule 8.3.4, $I(w)(P_{\geq s}\beta(t/x))_v = \top$ ako je $\mu(w)\{u \in W(w) : I(u)(\beta(t/x))_v = \top\} \geq s$. Dakle, vrednost terma t se računa u svetu u, pa se term odnosi na objekte iz svetova koji pripadaju $W(w)$. Ovo za posledicu ima da neke instance aksiome $(\forall x)\alpha(x) \to \alpha(t/x)$ nisu valjane. Da bi se to utvrdilo dovoljno je da model M bude sledećeg oblika:

- $W = \{w_1, w_2\}$
- $D(w_1) = D(w_2) = \{d_1, d_2\}$,
- $(M, w_1) \models P^1(d_1)$, $(M, w_1) \not\models P^1(d_2)$, $(M, w_2) \not\models P^1(d_1)$, $(M, w_2) \models P^1(d_2)$,
- $\mu(w_1)(\{w_1\}) = \frac{1}{2}$, $\mu(w_1)(\{w_2\}) = \frac{1}{2}$.

i da se simbol konstante c interpretira tako da je $I(w_1)(c) = d_2$ i $I(w_2)(c) = d_1$. U ovom modelu važi:

$$\mu(w_1)(\{w : w \models P^1(d_1)\}) = \mu(w_1)(\{w_1\}) = \tfrac{1}{2},$$

$$\mu(w_1)(\{w : w \models P^1(d_2)\}) = \mu(w_1)(\{w_2\}) = \tfrac{1}{2},$$

$$w_1 \not\models P^1(c) \text{ i}$$

$$w_2 \not\models P^1(c).$$

pa je

$$w_1 \models (\forall x)P_{\geq \frac{1}{2}}\alpha(x) \text{ i}$$

$$w_1 \not\models (\forall x)P_{\geq \frac{1}{2}}P(x) \to P_{\geq \frac{1}{2}}P(c). \qquad \Xi$$

Primer 8.3.7 Slična situacija se javlja i ako domeni nisu fiksirani. Istinitosna vrednost formule $(\forall x)P_{\geq s}\alpha(x)$ u nekom svetu w $LFOP_1$-modela $M = \langle W, D, I, Prob\rangle$, prema definiciji 8.3.4, zavisi od vrednosti $I(u)(\alpha(x))_{v[d/x]}$ u svetovima $u \in W(w)$. Sada se postavlja pitanje šta se dešava ukoliko d ne postoji u domenu $D(u)$ i kako aksiomatizovati klasu valjanih formula bez obzira na vrstu odgovora na prethidno pitanje. Ξ

8.4 Verovatnosne i modalne logike

Na početku ovog poglavlja je rečeno da verovatnosne logike predstavljaju jednu posebnu varijantu modalnih logika. Zaista, na osnovu dosada izloženog mogu se uočiti sličnosti ova dva tipa logika:

- jednu komponentu verovatnosnih modela predstavlja skup mogućih svetova,
- relacija zadovoljivosti je definisana tako da se u slučaju verovatnosnih operatora razmatraju istinitosne vrednosti formula u drugim svetovima, tako da su verovatnosni operatori, poput modalnih operatora, svojevrsni kvantifikatori nad mogućim svetovima,
- formula $P_{\geq 1}\alpha$ važi u nekom svetu verovatnosnog modela ako je mera svetova u kojima važi α jedan, dok formula $\Box\alpha$ važi u nekom svetu modalnog modela ako α važi u svim dostižnim svetovima,
- teorema 8.2.8.5 o distribuiranju verovatnosnog operatora $P_{\geq 1}$ podseća na aksiomu K kod modalnih logika itd.

8.5 Za dalje proučavanje

Istorijat teorije verovatnoće i prvih istraživanja u oblasti verovatnosnih logika opisan je, recimo, u [10, 28, 42, 50, 103]. Keislerovi radovi o verovatnosnim kvantifikatorima su [63, 64]. Nilsson je u [85, 86] razvio osnove verovatnosnog rezonovanja. Prve aksiomatizacije logika sa verovatnosnim operatorima mogu se naći u [31, 32]. Logika $LPP_2^{\mathrm{FR}(n)}$ je razmatrana u [106]. Logika LPP_1, kao i neke njene varijante analizirane su u [80, 90, 92, 93, 94, 106, 107] gde se nogu naći i detaljni dokazi odgovarajućih teorema potpunosti i odlučivosti. Nepostajanje rekurzivne aksiomatizacije za verovatnosnu logiku prvog reda dokazana je u [1]. Beskonačna aksiomatizacija za verovatnosnu logiku prvog reda data je u [94]. Verovatnosna logika sa operatorima oblika Q_F opisuje se u [92]. Rad [102] analizira operator kojim se modelira kvalitativno rezonovanje o verovatnoći. Logike sa uslovnim verovatnoćama opisane su u [56, 99]. Primena takvih logika u modeliranju nemonotonog zaključivanja opisuje se u [108, 109] U [25] je aksiomatizovana verovatnosna logika sa proizvoljnim, ali konačnim kodomenima. Druge varijacije kodomena analizirane su u [57]. Verovatnosna logika koje proširuje intuicionističku logiku analizira se u [78, 79]. U radu [101] verovatnosna logika je kombinovana sa diskretnom linearnom temporalnom logikom. Primena heurističkih procedura, poput genetskih algoritama i metode promenljivih okolina, u ispitivanju zadovoljivosti iskaznih verovatnosnih formula predložena je u [61, 96, 98, 100].

9

Teorija formalnih jezika

U ovom poglavlju će biti prikazana teorija formalnih jezika u kojoj se proučavaju osobine skupova nizova simbola i mogućnosti za njihovo predstavljanje. Osnovna sredstva koja se pri tome koriste su gramatike i automati. Pored tvrđenja o klasama formalnih jezika i vezama sa odgovarajućim tipovima gramatika i automata, navešćemo i više primena, pre svega u oblasti leksičke i sintaksne analize programskih jezika. Savremeni prevodioci i interpretatori programskih jezika zasnivaju se upravo na rezultatima teorije formalnih jezika. Donekle neočekivana veza najopštijih formalnih jezika i parcijalno izračunljivih funkcija o kojoj će biti reči u odeljku 9.6 pruža mogućnost da se obe oblasti međusobno prožmu i na taj način vide u jednom sasvim novom svetlu. Još jedno preplitanje, ovoga puta temporalnih logika i automata, opisano u odeljku 9.8.1, koristi se u postupku verifikacije.

9.1 Opis formalnih jezika

Alfabet[1] je konačan skup znakova. Recimo, binarni alfabet je skup $\{0, 1\}$. *Reč*[2] na nekom alfabetu je niz znakova tog alfabeta. Reč je *konačna*, odnosno *beskonačna*, ako je taj niz konačan, odnosno beskonačan. *Prazna reč*, u oznaci ε, predstavlja prazan niz znakova. *Dužina reči* je ukupan broj znakova niza koji čini reč. Dužina reči w se označava sa $|w|$. U nastavku, sem kada to bude posebno naglašeno koristićemo konačne reči. Ako su u i v dve reči na nekom alfabet njihovim *spajanjem*[3] se dobija reč uv. Pri tome važi $|uv| = |u| + |w|$. Primetimo da je slaganje reči asocijativna operacija za koju je neutral prazna reč. Ako je w reč, onda ćemo koristiti oznake $w^0 = \varepsilon$ i $w^{n+1} = ww^n$. Reč u je *podreč* reči v ako postoje reči x i y tako da je $v = xuy$. Ako je x prazna reč i $v = uy$, reč u je *prefiks* reči v. Slično, ako je y prazna reč i $v = xu$, reč u je *sufiks* reči v. Sa w^{-1} ćemo označavati reč u kojoj je redosled znakova obrnut u odnosu na reč w.

Ako je A alfabet, skup svih (konačnih) reči, uključujući i praznu reč, se označava sa A^*, a sa $A^+ = A^* \setminus \{\varepsilon\}$. Ako je alfabet A konačan skup, skup svih reči A^* je beskonačan prebrojiv skup. *Jezik* je bilo koji podskup skupa svih reči. U većini

[1] *Azbuka, vokabular.*
[2] *String, word.*
[3] *Konkatenacija, nadovezivanje.*

slučajeva se pretpostavlja da je jezik beskonačan skup. Ako je L jezik na alfabetu A, njegov *komplement* $C(L)$ je skup $A^* \setminus L$, tj. skup svih reči alfabeta koji nisu u L. Nad jezicima nad istim alfabetom se primenjuju:

- unarna operacija *zatvorenja*[4] definisana sa $L^* = \{w : w = w_1 w_2 \dots w_k$, za neko $k \geq 0, w_i \in L, i = 1, k\}$, gde je L neki jezik,
- binarna operacija *spajanja* definisana sa $L_1 \cdot L_2 = \{w : w = xy, x \in L_1, y \in L_2\}$, gde su L_1 i L_2 neki jezici,
- unarna operacija komplementiranja jezika, binarne operacije unije, preseka itd.

Primer 9.1.1 Neka je alfabet $A = \{a, b, c\}$. Tada su abc i aa reči za koje je $|abc| = 3$, $|aa| = 2$, dok je za praznu reč ε, $|\varepsilon| = 0$. Jedan konačan jezik na ovom alfabetu $\{abc, aba, aa, \varepsilon\}$, dok jedan beskonačan jezik sadrži sve reči u kojima se javlja samo znak a: $\{a, aa, aaa, \dots\}$. Ako je $L = \{01, 1, 100\}$ jezik na binarnom alfabetu, onda je reč 011 dobijena spajanjem prve i druge reči pripada jeziku L^*. Ako je jezik $L_1 = \emptyset$ prazan, onda je $L_1^* = \{\varepsilon\}$. Ako su jezici L_1 i L_2 na binarnom alfabetu takvi da je $L_1 = \{w : w$ sadrži samo neparan broj jedinica$\}$ i $L_2 = \{w : w$ počinje znakom 1 iza koga je konačan broj nula$\}$, onda je $L_1 \cdot L_2 = \{w : w$ počinje parnim brojem znakova 1 iza kojih ima nula ili više znakova 0$\}$. Ξ

9.1.1 Predstavljanje jezika

U vezi sa formalnim jezicima se postavljaju sledeća pitanja: kako predstaviti neki jezik, da li je neki beskonačan jezik moguće predstaviti konačnim sredstvima[5] i kako opisati klase jezika koji su predstavljeni konačnim sredstvima. Primetimo da se svaki konačan jezik može konačno predstaviti navođenjem svih reči.

Predstavljanje jezika se vrši:

- Zadavanjem postupka koji generiše sve reči jezika u nekom redosledu, tako da se svaka reč pojavi kao izlaz nakon nekog konačnog broja koraka rada.
- Zadavanjem postupka za prepoznavanje koji za svaku reč jezika koju dobija kao ulazni podatak odgovara sa 'da', dok za reči koje ne pripadaju jeziku odgovara sa 'ne' ili se, eventulano, ne zaustavlja, zavisno da li je skup reči jezika odlučiv ili ne.

Sistemi za generisanje jezika su: gramatike, regularni izrazi itd., dok sistemi za prepoznavanje jezika obuhvataju apstraktne mašine, takozvane automate, koji se, kao što je u odeljku 2.3.1 rečeno, dobijaju raznim restrikcijama modela Tjuringove mašine. Jasno je da su jezici za koje postoje postupci generisanja i/ili prepoznavanja barem parcijalno odlučivi.

[4] *Klinijeva zvezdica.*

[5] Pošto podskupova beskonačnog skupa ima neprebrojivo mnogo, a konačnim sredstvima se opisuje samo prebrojivo mnogo jezika, jasno je da većinu jezika nije moguće opisati konačnim sredstvima. Ipak, za neke interesantne klase jezika postoje konačni opisi.

9.2 Gramatike

9.2.1 Osnovne definicije

Definicija 9.2.1 *Gramatika* je uređena četvorka $G = \langle V_N, V_T, P, S\rangle$, gde su:

- V_N, V_T, P i S redom *skup neterminalnih znakova*, *skup terminalnih znakova*, *skup pravila izvođenja*[6] i *polazni znak*,
- V_N, V_T, i P neprazni, konačni skupovi,
- $V_N \cap V_T = \emptyset$,
- $V = V_N \cup V_T$,
- P je skup izraza oblika $\alpha \to \beta$, gde su $\alpha \in V^* \cdot V_N \cdot V^*$ i $\beta \in V^*$ i
- $S \in V_N$.

Ako su γ i δ reči na alfabetu V, odnosno $\gamma, \delta \in V^*$, i ako je $\alpha \to \beta \in P$, onda se reč $\gamma\beta\delta$ *direktno izvodi u gramatici* G iz reči $\gamma\alpha\delta$, u oznaci $\gamma\alpha\delta \to_G \gamma\beta\delta$. Ako su α_1, α_2, ..., α_n reči na alfabetu V za koje je $\alpha_1 \to_G \alpha_2$, ..., $\alpha_{n-1} \to_G \alpha_n$, onda se kaže da se reč α_n *izvodi* iz reči α_1, u oznaci $\alpha_1 \to_G^* \alpha_n$[7]. *Dužina izvođenja* $\alpha_1 \to_G^* \alpha_n$ je broj primena pravila izvođenja pomoću kojih se α_n izvodi iz α_1.

Jezik generisan nekom gramatikom je skup reči koje se sastoje od terminalnih znakova i koje se izvode iz reči S, preciznije

Definicija 9.2.2 Jezik $L(G)$ *generisan* gramatikom G je skup $\{w : S \to_G^* w, w \in V_T^*\}$. Dve gramatike G_1 i G_2 su *ekvivalentne* ako je $L(G_1) = L(G_2)$.

Primer 9.2.3 Razmotrimo gramatiku $G = \langle\{S\}, \{0,1\}, \{S \to 0S, S \to 1\}, S\rangle$ koja se sastoji od jednočlanog skupa neterminala koji sadrži samo polazni znak S, dvočlanog skupa terminala $\{0,1\}$ i dvočlanog skupa pravila izvođenja $\{S \to 0S, S \to 1\}$. Lako se pokazuje da se reč 001 izvodi iz reči S. Naime $S \to 0S \to 00S \to 001$ je izvođenje za posmatranu reč. U prvom i drugom koraku koristi se pravilo $S \to 0S$, a u poslednjem pravilo $S \to 1$. Ovde se može dokazati da je jezik $L(G)$ generisan gramatikom G upravo skup reči oblika 0^n1, za $n \geq 0$, gde 0^n označava reč koja se sastoji od n znakova 0. Najpre se pokaže da se svaka reč ovog oblika izvodi iz S, pošto se izvođenje sastoji iz n primena prvog pravila, nakon čega se jednom primeni drugo pravilo izvođenja. Zatim se indukcijom po dužini izvođenja pokazuje i da su sve reči koje se izvode iz S u k koraka oblika 0^kS ili $0^{k-1}1$. Za $k = 1$, odnosno za izvođenje u kojem je primenjeno samo jedno pravilo, jasno je da su moguće reči oblika $0S$ i 1 koje su željenog je oblika, pa za $k = 1$ tvrđenje važi. Neka su za $k \geq 1$ sve reči dobijene u k koraka željenog oblika 0^kS, odnosno $0^{k-1}1$. Druga reč je sastavljena samo od terminala, pa je dalje izvođenje primenljivo samo na prvu reč. Tu je moguće primeniti oba pravila, tako da se dobijaju reči oblika $0^{k+1}S$, odnosno 0^k1, čime je tvrđenje dokazano. Dakle, sve reči izvedene iz S i sastavljene od terminala su oblika 0^n1. Ξ

[6] Skup produkcija.

[7] U oba slučaja znak G se može izostaviti iz indeksa ukoliko to ne unosi zabunu.

9.2.2 Tipovi gramatika

Gramatike uvedene definicijom 150 su gramatike bez ograničenja, tj. gramatike *tipa* 0. Preostali tipovi se dobijaju postavljanjem nekih ograničenja na pravila izvođenja. Podela gramatika koju ćemo navesti se naziva *hijerarhija Čomskog*.

Definicija 9.2.4 Ako za svako pravilo izvođenja $\alpha \to \beta$ gramatike G važi da je $|\beta| \geq |\alpha|$, gramatika je *tipa* 1. Gramatike tipa 1 se nazivaju i *kontekstno osetljive*[8].

Ako za svako pravilo izvođenja $\alpha \to \beta$ gramatike G važi da je α neterminalni znak i da je β neprazna reč, gramatika je *tipa* 2 ili *kontekstno slobodna*[9].

Ako su sva pravila izvođenja oblika $A \to aB$, odnosno $A \to a$, gde su $A, B \in V_N$ i $a \in V_T$, gramatika je *tipa* 3. Za ovakve gramatike se kaže i da su *regularne*.

Primer 9.2.5 Gramatika $G = \langle \{A, B, S\}, \{0, 1, \ldots, 9\}, \{S \to 0, S \to 1, \ldots, S \to 9, S \to AB, A \to 1, A \to 2, \ldots, A \to 9, B \to 0, B \to 1, \ldots, B \to 9, B \to BB\}, S\rangle$ je kontekstno slobodna gramatika koja generiše jezik koji se sastoji od prirodnih brojeva. Primetimo da su pravila tako izabrana da se cifra 0 ne može pojaviti na poziciji najteže cifre višecifrenog broja. Gramatika $G = \langle \{A, S\}, \{0, 1, \ldots, 9\}, \{S \to 0, S \to 1, \ldots, S \to 9, S \to 1A, S \to 2A, \ldots, S \to 9A, A \to 0, A \to 1, \ldots, A \to 9, A \to 0A, A \to 1A, \ldots, A \to 9A\}, S\rangle$ je regularna gramatika koja generiše isti jezik. ⊐

Na osnovu ograničenja koja se postavljaju na pravila jasno je da su gramatike tipa 3 istovremeno i tipa 2, da su gramatike tipa 2 istovremeno i tipa 1, te da su gramatike tipa 1 istovremeno i tipa 0. Jezici koji im odgovaraju se nazivaju: jezici tipa 0 ili *parcijalno odlučivi jezici*, tipa 1 ili *kontekstno osetljivi jezici*, tipa 2 ili *kontekstno slobodni jezici* i tipa 3 ili *regularani jezici*.

Primetimo ovde da, prema prethodnim definicijama, prazna reč ε ne pripada ni jednom od jezika tipa 1, 2 ili 3. Kako je u raznim situacijama pogodno da jezik sadrži i praznu reč, definicija se slabi tako da se dozvoli pravilo oblika $S \to \varepsilon$, gde je S polazni znak i S se ne nalazi sa desne strane ni jednog pravila izvođenja. Jasno je da se pravilo $S \to \varepsilon$ može upotrebiti samo u prvom koraku izvođenja. Imajući u vidu ovu izmenu, važi sledeće tvrđenje:

Teorema 9.2.6 Ako je L jezik tipa 1, 2 ili 3, tada su istog tipa i jezici $L \cup \{\varepsilon\}$ i $L \setminus \{\varepsilon\}$.

Dokaz. Neka je $G = \langle V_N, V_T, P, S\rangle$ gramatika tipa 1, 2 ili 3 u odnosu na definiciju 152 (pa u njoj nema pravila oblika $A \to \varepsilon$) i $S_1 \notin V$. Posmatrajmo gramatiku $G_1 = \langle V_N \cup \{S_1\}, V_T, P_1, S_1\rangle$ kod koje skup P_1 sadrži skup P i još $\{S_1 \to \alpha : S \to \alpha \in P\}$. Jasno je da se dobija gramatika istog tipa u kojoj se polazni znak ne nalazi sa desne strane ni jednog pravila. Takođe važi i da $L(G) = L(G_1)$. Ako $S \to \alpha \to^*_G w$, onda se u G_1 izvodi $S_1 \to \alpha \to^*_{G_1} w$. Slično, ako je $S_1 \to \alpha \to^*_{G_1} w$ reč α ne sadrži znak S_1, pa se u G izvodi $S \to \alpha \to^*_G w$.

[8] Ovo ime dolazi otuda što se ograničenje na pravila izvođenja ovih gramatika može ekvivalentno formulisati tako da sva pravila budu oblika $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$, gde su $\alpha_1, \alpha_2 \in V^*$, $A \in V_N$ i $\beta \in V^+$. Pošto reč β nije prazna, očigledno je da $|\alpha_1 A \alpha_2| \leq |\alpha_1 \beta \alpha_2|$. U slučaju ovakvih gramatika neterminalni znak A se zamenjuje rečju β samo kad se A nalazi u kontekstu α_1, α_2.

[9] U ovom slučaju se neterminalni znak A zamenjuje rečju β bez obzira na kontekst u kome se A nalazi.

Na osnovu prethodnog možemo pretpostaviti da za gramatiku G za koju je $L = L(G)$ važi da se polazni znak ne nalazi sa desne strane ni jednog pravila. Dodavanjem pravila $S \to \varepsilon$, lako se postiže da se jezik proširuje sa $\{\varepsilon\}$. Slično se izbacivanjem pravila $S \to \varepsilon$ obezbeđuje da se ε ne nalazi u jeziku. Pošto sva ostala pravila ostaju ista, nova gramatika zadržava tip polazne. Ξ

zbog čega ćemo u nastavku po potrebi uključivati ili isključivati praznu reč iz jezika koje budemo analizirali.

9.2.3 (Ne)odlučivi problemi kod gramatika

Gramatika je *odlučiva* ako je odlučiv njome generisani jezik shvaćen kao skup, tj. odlučiv je problem da li neka reč pripada jeziku. Slično je i sa parcijalnom odlučivošću. Prema teoremi 9.5.4, odlučive su sve gramatike tipa 1 ili višeg, pa za svaki odgovarajući jezik postoji postupak koji za svaku reč koja pripada jeziku odgovara sa 'da', a za sve ostale reči sa 'ne'. Međutim, sledeći problemi nisu odlučivi u opštem slučaju, tj. za gramatike tipa 0 u kojima nema restrikcija:

- da li proizvoljna reč pripada jeziku generisanom gramatikom, tj. gramatike tipa 0 nisu odlučive,
- da li je jezik generisan jednom gramatikom podskup jezika generisanog drugom gramatikom,
- da li su dve gramatike ekvivalentne,
- da li je generisani jezik prazan,
- da li je generisani jezik beskonačan.

Takođe, nisu odlučivi ni *problem reči za Tue sisteme* (Axel Thue, 1863 – 1922.) i *Postov problem korespondecije*. U prvom problemu se posmatra Tue sistem, odnosno konačan skup J parova reči za koje se kaže da $x \sim y$ ako (x, y) ili (y, x) pripadaju skupu J. Postavlja se pitanje da li su dve reči w i u ekvivalentne, odnosno da li se proizvoljnim brojem zamena podreči iz w odgovarajućim rečima sa kojima su u relaciji $\sim$ može dobiti reč u. Zapravo, postavlja se pitanje da li $w \to^* u$ u gramatici čiji skup pravila sadrži $x \to y$ i $y \to x$, za $(x, y) \in J$. Sličan je i drugi problem je u kome se za zadati konačan alfabet A posmatraju dve liste od po k reči $l_1 = w_1, w_2, \ldots, w_k$ i $l_2 = u_1, u_2, \ldots, u_k$, a postavlja se pitanje da li postoji niz brojeva $i_1, i_2, \ldots, i_m$, $m \geq 1$, tako da je $w_{i_1} w_{i_2} \ldots w_{i_m} = u_{i_1} u_{i_2} \ldots u_{i_m}$.

9.3 Regularni jezici i konačni automati

U ovom odeljku ćemo detaljnije analizirati regularne jezike i odnos gramatika koje ih generišu i apstraktnih mašina koje ih prepoznaju.

9.3.1 Konačni automati

Konačni automati se mogu shvatiti kao Tjuringove mašine koje ne poseduju memoriju, odnosno trake u koje bi se mogli smestiti pomoćni i izlazni podaci. Konačni automati mogu jedino menjati svoja stanja u zavisnosti od ulaznih podataka, tako da samo zaustavljanjem u nekom od unapred izdvojenih stanja signaliziraju da li te podatke prihvataju ili ne. Konačni automati se uprkos ovakvim ograničenjima koriste u praksi, recimo u modulima za leksičku analizu prevodilaca programskih jezika gde vrše izdvajanje leksičkih celina, poput brojeva i imena iz izvornog zapisa programa.

Definicija 9.3.1 *Konačni automat* je uređena petorka $M = \langle S, s, F, A, \delta \rangle$, gde su:

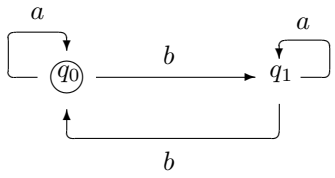
- S konačan skup stanja,
- $s \in S$ početno stanje,
- $F \subset S$ skup završnih stanja,
- A alfabet i
- $\delta : S \times A \to S$ funkcija prelaza koja svakom paru koji se sastoji od stanja $q \in S$ i znaka $a \in A$ pridružuje neko stanje $q' \in S$.

Rad konačnog automata intuitivno se shvata na sledeći način. Na početku rada automat se nalazi u stanju s. Zatim se učita znak sa ulaza i prelazi u stanje definisano funkcijom δ. Postupak se ponavlja tako što se čita znak po znak i menjaju stanja, sve dok se ne stigne do kraja ulaza. Pošto se neki znak učita i dovede do eventualne promene stanja, konačni automat se na njega više ne vraća. Zato je za opis situacije u kojoj se nalazi konačni automat dovoljno opisati tekuće stanje i preostali, još nepročitani deo ulazne reči. Kao i u slučaju Tjuringovih mašina, ovaj opis se naziva *konfiguracija*. Konfiguracija je uređeni par (q, w), gde je $q \in S$ stanje i $w \in A^*$ reč na alfabetu A. *Računski korak* je svaki par konfiguracija (q, w), (q', w') gde je $w = aw'$, $a \in A$ i $\delta(q, a) = q'$. *Izračunavanje* je niz konfiguracija sa osobinom da svake dve uzastopne konfiguracije čine računski korak. Sa $(q, w) \vdash_M (p, u)$ ćemo označiti da postoji izračunavanje automata M čiji je prvi element konfiguracija (q, w), a poslednji (p, u).

Konačni automat $M = \langle S, s, F, A, \delta \rangle$ *prihvata* reč $w \in A^*$, ako za neko završno stanje $q \in F$ važi $(s, w) \vdash_M (q, \varepsilon)$. $L(M)$ će označavati jezik, tj. skup svih reči koje prihvata konačni automat M.

Primer 9.3.2 Neka je konačni automat $M = \langle \{q_0, q_1\}, q_0, \{q_0\}, \{a, b\}, \delta \rangle$, gde je funkcija δ definisana sa $\delta(q_0, a) = q_0$, $\delta(q_0, b) = q_1$, $\delta(q_1, a) = q_1$, $\delta(q_1, b) = q_0$. Jezik $L(M)$ je skup svih reči koje sadrže paran broj pojava znaka b. Na primer, izračunavanje koje odgovara $(q_0, aabba) \vdash_M (q_0, \varepsilon)$ je niz konfiguracije $(q_0, aabba)$, $(q_0, abba)$, (q_0, bba), (q_1, ba), (q_0, a), (q_0, ε), pa M prihvata reč $aabba$.

Konačni automati se prikazuju i pomoću dijagrama, direktnih grafova čiji su čvorovi označeni stanjima, a ivice znacima alfabeta. Postojanje ivice označene sa b koja ide od čvora označanog stanjem q_0 ka čvoru označenom sa q_1 znači da u odgovarajućem konačnom automatu $\delta(q_0, b) = q_1$. Na slici 9.1 je dat dijagram koji prikazuje prethodno opisani konačni automat. Ponekad se, kao na slici 9.1, završna stanja ističu time što su zaokružena. Ξ



Slika 9.1. Dijagram za konačni automat iz primera 9.3.2.

Slično nedeterminističkim Tjuringovim mašinama postoje i *nedeterministički konačni automati* koji se od do sada razmatranih, *determinističkih*, razlikuju po tome što funkcija $\delta : S \times A \to \mathbb{P}(S)$ svakom paru $(q, a) \in S \times A$ pridružuje neki podskup skupa svih stanja. Ideja ovog pristupa je da se iz tekućeg stanja q kada se učita znak a prelazi u jedno od stanja sadržanih u $\delta(q, a)$. Pri tome se ne precizira koje je to naredno stanje. I nedeterministički konačni automati se mogu opisati dijagramom, ali sada iz jednog čvora može izlaziti više ivica označenih istim znakom koje vode u različite čvorove.

U slučaju determinističkih konačnih automata ulazna reč je jedinstveno određivala stanje do koga se dolazi polazeći od početnog stanja. Kod nedeterminističkih konačnih automata to nije slučaj, ali se prihvatanje reči definiše analogno. Naime, nedeterministički konačni automat *prihvata* reč $w \in A^*$, ako za neko završno stanje $q \in F$ postoji bar jedan niz konfiguracija koje predstavljaju izračunavanje za koje je poslednji član oblika (q, ε).

Iako je nazigled pojam nedeterminističkog konačnog automata opštiji od determinističkog, pokazuje se da to nije tako.

Definicija 9.3.3 Dva konačna automata M_1 i M_2 su *ekvivalentna* ako je $L(M_1) = L(M_2)$.

Teorema 9.3.4 Jezik L je prihvaćen nekim nedeterminističkim konačnim automatom ako i samo ako je prihvaćen nekim determinističkim konačnim automatom.

Dokaz. Primetimo najpre da je ($\Leftarrow$) smer trivijalan jer je svaki deterministički konačni automat istovremeno i nedeterministički. Prema tome razmotrimo samo smer ($\Rightarrow$). Neka je $M = \langle S, s, F, A, \delta\rangle$ nedeterministički konačni automat. Konstruisaćemo njemu ekvivalentan konačni automat $M' = \langle S', s', F', A, \delta'\rangle$, gde su $S' = \mathbb{P}(S)$ partitivni skup skupa S, $s' = \{s\}$, $F' = \{q' \in S' : q' \cap F \neq \emptyset\}$ i $\delta'(\{q_1, \ldots, q_n\}, a) = \cup_{i=1}^{n} \delta(q_i, a)$. Jasno je da je M' deterministički konačni automat. Sa $\{q_1, \ldots, q_n\}_{s,w,M}$ označimo da su q_1, $\ldots$, q_n sva stanja do kojih se radom nedeterminističkog automata M može doći za ulaznu reč w. Indukcijom po dužini reči w se lako pokazuje da je $\{q_1, \ldots, q_n\}_{s,w,M}$ ako i samo ako $(\{s\}, w) \vdash_{M'} (\{q_1, \ldots, q_n\}, \varepsilon)$. Za praznu reč tvrđenje trivijalno važi. Neka je dalje $w = ua$, $a \in A$. Prema indukcijskoj pretpostavci $\{p_1, \ldots, p_j\}_{s,u,M}$ ako i samo ako $(\{s\}, u) \vdash_{M'} (\{p_1, \ldots, p_j\}, \varepsilon)$. Dalje je $\{q_1, \ldots, q_n\}_{s,w,M} = \cup_{i=1}^{j} \delta(p_i, a)$, što je upravo definicija za $\delta'(\{p_1, \ldots, p_j\}, a)$, pa je tvrđenje dokazano. Sada neposredno sledi da za svaku reč w, $w \in L(M)$ ako i samo ako $w \in L(M')$, jer za reč w postoji

izračunavanje automata M koje dovodi do nekog završnog stanja q ako i samo ako izvršavanje automata M' za w dovodi do stanja koje sadrži q, a koje je završno stanje za M. Ξ

Ova jednakost se koristi kad god je u nekoj situaciji pogodno raditi sa jednom ili drugom vrstom automata, a da dokazano važi za sve konačne automate. Primetimo da, iako su deterministički i nedeterministički konačni automati jednako izražajni, prilikom konstrukcije determinističkog konačnog automata u teoremi 9.3.4 dolazi do eksponencijalnog povećanja veličine automata jer, ako nedetreministički automat M ima n stanja, detreministički automat M' ima 2^n stanja. Pokazano je da je takav porast broja stanja neizbežan.

9.3.2 Odnos regularnih jezika i konačnih automata

Sledeća teorema dovodi u vezu gramatike tipa 3 i konačne automate i zapravo znači da je jezik L regularan ako i samo ako postoji konačni automat M takav da je $L = L(M)$.

Teorema 9.3.5 Za svaku gramatiku G tipa 3 postoji konačni automat M takav da je $L(G) = L(M)$ i obrnuto, za svaki konačni automat M postoji gramatika G tipa 3 takva da je $L(M) = L(G)$.

Dokaz. ($\Rightarrow$) Neka je gramatika $G = \langle V_N, V_T, P, S_0\rangle$. Definisaćemo nedeterministički konačni automat $M = \langle S, S_0, F, V_T, \delta\rangle$ gde su:

- $S = V_N \cup \{Q\}$, gde $Q \notin V_N \cup V_T$,
- $F = \{Q\}$, ako $\varepsilon \notin L(G)$, a ako $\varepsilon \in L(G)$, $F = \{Q, S_0\}$ i
- $Q \in \delta(B, a)$ ako je $B \to a \in P$,
- $C \in \delta(B, a)$ ako je $B \to aC \in P$ i
- $\delta(Q, a) = \emptyset$.

Sada se lako vidi da u gramatici G

$$S_0 \to a_1A_1 \to \ldots \to a_1 \ldots a_{n-1}A_{n-1} \to a_1 \ldots a_{n-1}a_n$$

ako i samo ako postoji izračunavanje tako da je $(S_0, a_1 \ldots a_{n-1}a_n) \vdash_M (Q, \varepsilon)$. Dakle, $L(G) = L(M)$.
($\Leftarrow$) Neka je sada $M = \langle S, s, F, A, \delta\rangle$ deterministički konačni automat. Definisaćemo gramatiku $G = \langle S, A, P, s\rangle$ tako da je:

- $B \to aC \in P$ ako je $\delta(B, a) = C \notin F$ i
- $B \to a \in P$ ako je $\delta(B, a) \in F$

odakle se lako pokazuje da $(s, w) \vdash_M (q, \varepsilon)$ za neko $q \in F$ ako i samo ako $s \to_G^* w$. Dakle, $L(M) = L(G)$. Ξ

Primetimo da se u dokazu teoreme 9.3.5 pri konstrukciji konačnog automata za datu regularnu gramatiku neterminalni znaci postaju stanja, dok pravilu oblika $B \to aC$ odgovara ivica označena znakom a koja od stanja B vodi ka stanju C.

9.3.3 Svojstva regularnih jezika

U narednoj teoremi ćemo pokazati da kada uobičajene skupovne operacije primenimo na regularne jezike, rezultat ostaje jezik istog tipa.

Teorema 9.3.6 Klasa regularnih jezika je zatvorena za:

1. uniju,
2. nadovezivanje,
3. zatvorenje (Klinijevu zvezdicu),
4. komplementiranje i
5. presek.

Dokaz. (1) Neka su $L(M_1)$ i $L(M_2)$ dva regularna jezika i $M_1 = \langle S_1, s_1, F_1, A, \delta_1\rangle$ i $M_2 = \langle S_2, s_2, F_2, A, \delta_2\rangle$ odgovarajući nedeterministički konačni automati. Pretpostavimo da je $S_1 \cap S_2 = \emptyset$. Posmatrajmo konačni automat $M = \langle S_1 \cup S_2 \cup \{s\}, s, F_1 \cup F_2, A, \delta\rangle$ tako da važi:

- $s \notin S_1 \cup S_2$ i
- $\delta = \delta_1 \cup \delta_2 \cup \{(s, \varepsilon, s_1), (s, \varepsilon, s_2)\}$.

Dakle, automat M najpre nedeterministički bira (na osnovu pročitane prazne reči, tj. ne čitajući ni jedan znak) da li da pređe u stanje s_1 ili s_2, a zatim oponaša rad M_1 ili M_2. Jasno je da $w \in L(M_1) \cup L(M_2)$ ako i samo ako $w \in L(M)$.
(2) Kao i u prošlom koraku, može se definisti novi konačni automat na osnovu automata M_1 i M_2. Ovde će se iz završnih stanja automata M_1 veštački (čitanjem prazne reči) preći u početno stanje automata M_2 i potom nastaviti rad.
(3) Zatvorenost za operaciju Klinijeve zvezdice se pokazuje slično koraku (2). Neka je $L = L(M_1)$, za $M_1 = \langle S_1, s_1, F_1, A, \delta_1\rangle$. Novi automat $M = \langle S_1 \cup \{s\}, s, F_1 \cup \{s\}, A, \delta\rangle$ ima sva stanja kao i polazni automat M_1 i novo stanje s koje je početno. Funkcija prelaza se dodefiniše tako da se iz s čitanjem prazne reči prelazi u stanje s_1, kao i da se iz završnih stanja vraća praznom rečju u stanje s_1. Stanje s je istovremeno i završno, čime se omogućava da se prihvata i prazna reč koju po deficiji sadrži jezik $L(M_1)^*$.
(4) Neka je $L = L(M_1)$ za deterministički konačni automat $M_1 = \langle S_1, s_1, F_1, A, \delta_1\rangle$. Komplement $A^* \setminus L$ jezika L se prihvata automatom $M = \langle S_1, s_1, S_1 \setminus F_1, A, \delta_1\rangle$ u kome se jedino menjaju završna stanja u odnosu na M_1.
(5) Zatvorenost za presek je posledica zatvorenosti za uniju i komplement, tj. $L_1 \cap L_2 = A^* \setminus ((A^* \setminus L_1) \cup (A^* \setminus L_2))$. Ξ

Posledica teoreme 9.3.5 je da su odlučivi problemi:

- da li reč pripada regularnom jeziku, jer se proveri da li odgovarajući konačni automat prihvata reč,
- da li je regularan jezik prazan, jer se proveri da li u dijagramu za odgovarajući konačni automat postoji put od početnog do nekog od završnih čvorova,

- da li je regularan jezik univerzalan, odnosno jednak skupu svih reči alfabeta, jer se proveri da li je jezik prihvaćen komplementom odgovarajućeg konačnog automata prazan,
- da li je regularni jezik L_1 podskup regularnog jezika L_2, jer se proveri da li je presek komplementa jezika L_2 i jezika L_1 prazan i
- da li su regularni jezici L_1 i L_2 jednaki, što neposredno sledi iz prethodnog problema.

Primetimo da su svi konačni jezici regularni, jer se mogu predstaviti kao konačna unija reči, a za svaku pojedinačnu reč trivijalno postoji konačni automat koji je prihvata. Uopštenje ove konstatacije je:

Teorema 9.3.7 Svaki regularan jezik se može dobiti konačnim brojem primena operacija unije, nadovezivanja i Klinijeve zvezdice na konačne jezike.

Dokaz. Neka je L regularan jezik i $M = \langle\{q_1, \ldots, q_n\}, q_1, F, A, \delta\rangle$ konačni automat tako da je $L = L(M)$. Uvodimo skupove $R^k_{i,j}$ svih reči $w \in A^*$, tako da se iz stanja q_i učitavanjem reči w prelazi u stanje q_j, a da se pri tom ne nađe u stanju q_l za $l > k$. Preciznije, $R^0_{i,j} = \{a \in A : \delta(q_i, a) = q_j\}$ i

$$R^{k+1}_{i,j} = R^k_{i,j} \cup (R^k_{i,k+1} \cdot (R^k_{k+1,k+1})^* \cdot R^k_{k+1,j}).$$

Sada je jasno da se svaki skup $R^k_{i,j}$ dobija konačnim brojem primena operacija unije, nadovezivanja i Klinijeve zvezdice na konačne jezike. Kako važi $L(M) = \cup_{q_j \in F} R^n_{1,j}$, direktno sledi traženo tvrđenja. $\square$

Dalje, jezik koji prihvata konačni automat sa n stanja je neprazan ako i samo ako automat prihvata bar jednu reč kraću od n znakova jer se iz svakog izračunavanja mogu izbaciti i u svako izračunavanje ubaciti petlje u kojima se jedno isto stanje javlja na početku i kraju petlje. Slično, regularan jezik je beskonačan ako i samo ako odgovarajući automat prihvata bar jednu reč dužine između n i $2n - 1$.

9.3.4 Jezici koji nisu regularni

Pošto regularnih jezika ima prebrojivo mnogo, a svih jezika neprebrojivo, jasno je da postoje jezici koji nisu regularni. Sledeća teorema[10] pruža jedan kriterijum za utvrđivanje da jezik nije regularan.

Teorema 9.3.8 Neka je L beskonačan regularan jezik. Tada postoje reči x, y i z tako da važi $y \neq \varepsilon$ i $xy^kz \in L$ za svaki $k \geq 0$.

Dokaz. Neka je $M = \langle\{q_1, \ldots, q_n\}, q_1, F, A, \delta\rangle$ deterministički konačni automat sa n stanja za koji je $L = L(M)$. Pošto je L beskonačan, postoji reč $w = a_1 \ldots a_{|w|} \in L$ takva da $|w| \geq n$. Najduži put u konačnom automatu u kome se svaki čvor javlja najviše jednom sadrži najviše $n-1$ ivicu, pa se prilikom prihvatanja reči w automat

[10] Njeno ime, lema naduvavanja, (pumping lemma), dolazi od načina konstruisanja reči koje se koriste u dokazu.

mora naći bar dva puta u istom stanju. Neka je to stanje q. Znači da postoji podreč y reči w koja, kada počne da se čita u stanju q dovodi do istog tog stanja. Odatle je reč w oblika xyz za neke x, y i z tako da je $y \neq \varepsilon$, $(q_1, w) \vdash_M (q, yz)$, $(q, yz) \vdash_M (q, z)$, $(q, z) \vdash_M (q', \varepsilon)$ i $q' \in F$. Takođe je očigledno da isti automat prihvata i reči $xy^k z$ ne prolazeći petlju koja počinje i završava u stanju q, pri čemu je $k = 0$, odnosno prolazeći petlju proizvoljan konačan broj puta za $k > 0$. Ξ

Prema teoremi 9.3.8 regularnim izrazima, odnosno gramatikama tipa 3, se mogu opisati, a konačnim automatima prepoznati, samo reči u kojima se nalazi bilo neki fiksan, bilo proizvoljan broj ponavljanja neke konstrukcije. Nije moguće izvršiti poređenje dva broja ponavljanja nekih konstrukcija kako bi se videlo da li su jednaki, kao što je slučaj u primeru 9.3.9 sa jezikom $\{0^n 1^n\}$.

Primer 9.3.9 Jezik $\{0^n 1^n : n \geq 0\}$ je beskonačan. Kada bi jezik bio i regularan postojali bi stringovi x, $y \neq \varepsilon$ i z takvi da $xy^k z \in L$ za $k \geq 0$. Ako je $y = 0^q$, onda mora biti $x = 0^p$, $z = 0^r 1^s$. Jasno je da bismo naduvavanjem ove reči dobili reč u kojoj ima više znakova 0 nego 1, koja ne pripada jeziku. Slična je i situacija u kojoj je $y = 1^q$. Konačno, ako je $y = 0^p 1^q$ naduvavanjem se dobija reč u kojoj se neki 1 nalaze levo od nekih 0 i koja očigledno ne pripada jeziku L. Ξ

9.3.5 Regularni izrazi

Regularni izrazi nad alfabetom A se definišu na sledeći način:

- prazna reč ε, prazan skup $\emptyset$ i svi znaci alfabeta A su regularni izrazi,
- ako su α i β regularni izrazi, regularni izrazi su i $(\alpha \cdot \beta)$, $(\alpha \cup \beta)$ i α^* i
- regularni izrazi se dobijaju samo konačnom primenom prethodna dva koraka

i predstavljaju još jedno sredstvo za opis regularnih jezika. Regularni izrazi se mogu shvatiti kao neka vrsta generatora jezika. Na primer, izraz $0 \cdot ((0^* \cup 1^*) \cdot 1)$ se može shvatiti kao generator reči koje počinju znakom 0 iza koga sledi ili k znakova 0, ili l znakova 1, gde su $k, l \geq 0$, i koja završava znakom 1. Preciznije, svakom regularnom izrazu α se pridružuje jezik $L(\alpha)$ na sledeći način:

- $L(\varepsilon) = \{\varepsilon\}$, $L(\emptyset) = \emptyset$ i za $a \in A$, $L(a) = \{a\}$ i
- $L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$, $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$ i $L(\alpha^*) = (L(\alpha))^*$.

Primer 9.3.10 Na osnovu rečenog važi $L(0 \cdot ((0^* \cup 1^*) \cdot 1)) = L(0) \cdot L((0^* \cup 1^*) \cdot 1) = L(0) \cdot (L(0^* \cup 1^*) \cdot L(1)) = L(0) \cdot ((L(0^*) \cup L(1^*)) \cdot L(1)) = L(0) \cdot ((L(0)^* \cup L(1)^*) \cdot L(1)) = \{00^k 1, 01^l 1 : k, l \geq 0\}$. Takođe je $L(a^* \cdot b^*) = L(a^*) \cdot L(b^*) = (\{a\})^* \cdot (\{b\})^* = \{a^k b^l : k, l \geq 0\}$ i slično $L(\varepsilon \cup (a \cdot b)^*) = L(\varepsilon) \cup L((a \cdot b)^*) = \{\varepsilon\} \cup (L(a \cdot b))^* = \{\varepsilon\} \cup (\{ab\})^* = \{(ab)^n : n \geq 0\}$. Ξ

Veza regularnih jezika i regularnih izraza opisana je sledećim tvrđenjima.

Teorema 9.3.11 Za svaki konačan jezik L postoji regularan izraz α tako da je $L = L(\alpha)$.

Dokaz. Prazan jezik, jezik koji sadrži praznu reč i jednočlani jezici čije reči su dužine jedan su predstavljeni sa $L(\emptyset)$, $L(\varepsilon)$ i $L(a)$, za a iz alfabeta. Jednočlani jezici čije su reči nizovi znakova alfabeta se dobijaju nadovezivanjem jezika predstavljenih tim znacima. Konačni jezici sa više nego jednim elementom se dobijaju unijom jednočlanih jezika. Ξ

Teorema 9.3.12 Jezik L je regularan ako i samo ako postoji regularan izraz α tako da važi $L = L(\alpha)$.

Dokaz. ($\Rightarrow$) Pošto se, prema teoremi 9.3.7, svaki regularan jezik dobija iz konačnih jezika, a konačnim jezicima, prema teoremi 9.3.11, odgovaraju regularni izrazi, onda i svakom regularnom jeziku odgovara regularni izraz.
($\Leftarrow$) Sa druge strane, regularnim izrazima $\emptyset$, ε i a, za svaki znak alfabeta, odgovaraju konačni jezici koji jesu regularni. Kako su regularni jezici zatvoreni za operacije $\cup$, $\cdot$ i *, svakom regularnom izrazu odgovara regularni jezik. Ξ

Sredstvima korištenim u teoremi 9.3.6 moguće je svakom regularnom izrazu α pridružiti konačni automat M_α tako da bude $L(\alpha) = L(M_\alpha)$. U narednom primeru prikazan je ovaj postupak.

Primer 9.3.13 Regularni izraz $(ab \cup aab)^*$ predstavlja jezik $L((ab \cup aab)^*) = (L(ab) \cup L(aab))^*$. Izrazu se konačni automat pridružuje na sledeći način. Najpre se znacima a i b pridruže konačni automati:

$$s_a \xrightarrow{a} f_a$$

$$s_b \xrightarrow{b} f_b$$

Zatim se prave automati za izraze ab i aab:

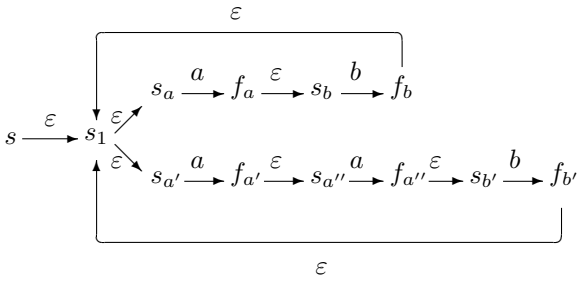
$$s_a \xrightarrow{a} f_a \xrightarrow{\varepsilon} s_b \xrightarrow{b} f_b$$

$$s_{a'} \xrightarrow{a} f_{a'} \xrightarrow{\varepsilon} s_{a''} \xrightarrow{a} f_{a''} \xrightarrow{\varepsilon} s_{b'} \xrightarrow{b} f_{b'}$$

i $ab \cup aab$:

$$\begin{array}{lll} & s_a \xrightarrow{a} f_a \xrightarrow{\varepsilon} s_b \xrightarrow{b} f_b & \\ s_1 \overset{\varepsilon}{\nearrow} & & \\ s_1 \underset{\varepsilon}{\searrow} & s_{a'} \xrightarrow{a} f_{a'} \xrightarrow{\varepsilon} s_{a''} \xrightarrow{a} f_{a''} \xrightarrow{\varepsilon} s_{b'} \xrightarrow{b} f_{b'} & \end{array}$$

i konačno



za $(ab \cup aab)^*$ u kome je početno stanje s, dok su završna stanja s, f_b i $f_{b'}$. Ξ

Takođe je zbog dokazanih ekvivalentnosti moguće izvesti i obrnuto pridruživanje, tj. za svaki konačan automat postoji ekvivalentan regularni izraz.

9.3.6 Leksička analiza

Leksički analizator vrši prvu fazu u prevođenju programa pisanih na nekom programskom jeziku u kojoj se učitava niz znakova iz izvornog programa i izdvajaju celine kao što su identifikatori ili brojevi koje se nazivaju tokeni. Leksički analizator se obično realizuje kao procedura koju po potrebi poziva sintaksni analizator kada mu se pojavi potreba za sledećim tokenom. Pored ovoga, leksički analizator može obavljati i neke dodatne poslove poput eliminisanja komentara, sažimanja većeg broja razmaka, takozvanih belih znakova u koje spadaju znaci za novi red, blanko i tab, razvoj makro-naredbi i sl.

Leksička analiza se u suštini svodi na obradu regularnih jezika. Zapravo, tokeni su reči u nekom regularnom jeziku koga po pravilu opisujemo regularnim izrazima ili gramatikama tipa 3 i koji se iz teksta izdvajaju u nekom vidu implementacije konačnog automata koji se dobija na osnovu opisa jezika. Konstrukcija konačnog automata i njegova implementacija u formi procedure na nekom progamskom jeziku se uglavnom radi automatski pomoću posebnih programskih oruđa. Najpoznatiji sistem za ovu namenu je Lex koji je javni program i deo standardne distibucije operativnog sistema Unix. Lexu se zadaje opis jezika u obliku niza definicija koje su zapravo regularni izrazi, a za koji se proizvodi funkcija na programskom jeziku C koja postavlja vrednosti određenih promenljivih preko kojih faza sintaksne analize u prevođenju dobija potrebne informacije.

Na slici 9.2 je dat primer jednog opisa koji dobija Lex. On se sastoji od tri dela. U prvom se navode konstantne vrednosti koje će koristi program i definicije regularnih izraza koji opisuju jezik. Na primer *bz*, beli znak, je izraz u kome se jedan ili više puta ponavlja bilo koji od znakova blanko, tab i novi red. Slično, slovo je bilo koji od znakova A do Z, odnosno a do z, dok je broj neprazan niz cifra iza kojih eventualno stoji tačka i novi niz cifara. U drugom delu opisa se navode akcije koje se preduzimaju ako se izdvoji određeni token. Recimo, ako se izdvoji beli znak, ne preduzima se nikakva akcija što znači da se nastavlja sa izdvajanjem sledećeg tokena. Ako se izdvoji broj, promenljivoj *yylval* se pridruži pokazivač na mesto u simboličkoj tabeli u koju je smešten taj broj. Konačno, u trećem delu opisa se definišu pomoćne funkcije koje se koriste tokom izdvajanja. Ovde su navedena samo imena funkcija koje izdvojene tokene smeštaju u simboličku tabelu.

```
%{
# define BROJ 0
# define ID   1
}%
delim [ \t\n]
bz {delim}+
cifra [0-9]
slovo [A-Za-z]
id {slovo}({slovo} | {cifra})*
broj {cifra}+(\. {cifra}+)?
%%
{bz} { }
{id} {yyval = postavi_id(); return(ID);}
{broj} {yyval = postavi_broj(); return(BROJ);}
%%
int postavi_broj() { /* kod funkcije */ }
int postavi_id() { /* kod funkcije */ }
```

Slika 9.2. Deo opisa koji dobija Lex.

U postupku konstrukcije simulatora rada konačnog automata najpre se od definisanih regularnih izraza konstruiše nedeterministički konačni automat kao što je opisano u odeljku 9.3.5, a zatim se kao u teoremi 9.3.4 vrši prelazak na deterministički konačni automat. Na kraju se deterministički automat optimizuje, odnosno minimizira se broj njegovih stanja. Može se pokazati da za svaki konačni automat postoji konačni automat koji prihvata isti jezik, a koji ima najmanji mogući broj stanja. Jedan postupak minimizacije je prikazan u nastavku teksta. Osnovni deo simulacije rada konačnog automata se vrši u petlji u kojoj se u zavisnosti od tekućeg stanja i učitanog znaka prelazi u novo stanje.

procedure min_ka(polazni automat M, izlazni automat M')
begin
(1) Napraviti polaznu podelu Π skupa stanja na dva skupa: skup završnih stanja i skup preostalih stanja.
(2) Pomoću
for svaka grupa G iz Π **do**
podeliti G u podgrupe tako da su u istoj podrupi stanja s i t ako i samo ako se za svaki terminalni znak a iz stanja s i t prelazi u stanja koja su u istoj grupi u Π.
napraviti podelu Π_n.
(3) Ako je $\Pi = \Pi_n$, postaviti $\Pi_f := \Pi$ i preći na (4), inače postaviti $\Pi := \Pi_n$ i preći na (2).
(4) Iz svake grupe u Π_f izabrati po jednog predstavnika. On će biti stanje u M'. Ivica označena terminalnim znakom a će ići od predstavnika jedne do predstavnika druge grupa ako i samo ako ide između bilo koja dva člana tih grupa. Početno stanje u M' će biti predstavnik grupe u kojoj je početno stanje automata M, a slično i završna stanja u M' će biti predstavnici grupa koje sadrže završna stanja automata M.

(5) Odbaciti sva stanja koja nisu dostižna iz početnog stanja.

9.4 Kontekstno slobodni jezici i potisni automati

Podsetimo se da su kontekstno slobodni jezici generisani gramatikama za čija pravila važi:

ako $\alpha \to \beta \in P$, onda $\alpha \in V_N$, $\beta \in V^+$,

te da se ova definicija može proširiti tako da dozvoljava i izvođenje prazne reči. Kontekstno slobodni jezici se koriste za specifikaciju programskih jezika i prevođenje programa pisanih na njima.

Primer 9.4.1 Gramatika $G = \langle\{E\}, \{id, +, *, (,)\}, P, E\rangle$ čiji skup P sadrži:

- $E \to E + E$,
- $E \to E * E$,
- $E \to (E)$,
- $E \to id$

opisuje deo jezika aritmetičkih izraza koji sadrži većina programskih jezika. Intuitivno, terminalni simbol id znači identifikator, tj. ime promenljive. Ξ

U zadatku 11.93 je opisana kontekstno slobodna gramatika kojoj odgovara jezik $\{0^n 1^n : n \geq 0\}$. Kako je u odeljku 9.3.4 pokazano da taj jezik nije regularan, postoje jezici koji nisu regularni, a jesu kontekstno slobodni. Sa druge strane, na osnovu uvida u ograničenja za pravila koja su dozvoljena u gramatikama, jasno je da su svi regularani jezici i kontekstno slobodni.

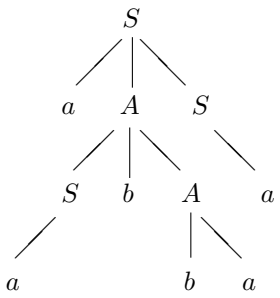
9.4.1 Drveta izvođenja i vrste izvođenja

Jedan od načina predstavljanja izvođenja u kontekstno slobodnim gramatikama je u formi *drveta izvođenja*[11] čiji su čvorovi označeni elementima skupa V. Koren tog drveta je polazni znak. Ako znaci koji označavaju s leva na desno sve naslednike nekog čvora označenog sa A ($A \in V_N$) čine reč α, onda je $A \to \alpha$ pravilo izvođenja gramatike. Listovi drveta su terminali koji čine reč koja se izvodi iz polaznog znaka. Pokazuje se da reč pripada nekom kontekstno slobodnom jeziku ako i samo ako za nju postoji drvo izvođenja.

Primer 9.4.2 Neka je $G = \langle\{S, A\}, \{a, b\}, \{S \to a, S \to aAS, A \to SbA, A \to SS, A \to ba\}, S\rangle$ kontektno slobodna gramatika. Izvođenje $S \to aAS \to aSbAS \to aabAS \to aabbaS \to aabbaa$ se prikazuje drvetom na slici 9.3. Ξ

Neka je p najveći broj znakova sa desne strane bilo kog pravila izvođenja u G. Tada u drvetu izvođenja čija najduža grana sadrži m čvorova ima najviše p^{m-1}

[11] *Parsee tree.*



Slika 9.3. Drvo za izvođenje iz primera 9.4.2.

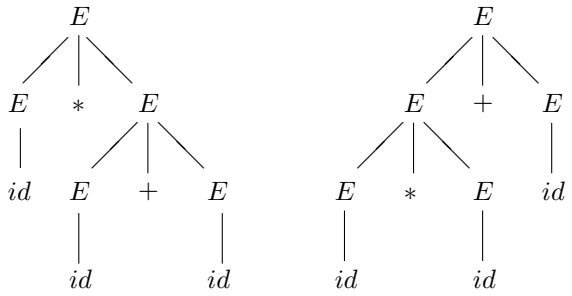
listova. Odatle, ako je reč iz jezika duža od p^{m-1}, onda u odgovarajućem drvetu izvođenja postoji grana sa bar $m+1$ čvorom.

Sva izvođenja u kontekstno slobodnim gramatikama se mogu organizovati tako da je neterminalni znak na koji se neko pravilo primenjuje uvek najlevlji neterminalni znak u reči. Takvo izvođenje nazovimo *najlevlje izvođenje*. Slično je i sa *desnim izvođenjem* u kome se sva pravila primenjuju na krajnji desni neterminalni znak. Ako je negde potrebno naglasiti, sa $\rightarrow_G^{*l}$, odnosno $\rightarrow_G^{*d}$, ćemo označavati najlevlje, odnosno desno, izvođenje. Slično, sa $\rightarrow^l$, odnosno $\rightarrow^d$, ćemo označavati jedan korak najlevljeg, odnosno desnog, izvođenja. Intuitivno, postojanje najlevljeg (odnosno desnog) izvođenja se da naslutiti na osnovu analize drveta izvođenja i činjenice da kontekst ne utiče na izbor pravila izvođenja, a precizno se dokazuje u teoremi 9.4.3.

Teorema 9.4.3 Za svaku reč w nekog kontekstnog slobodnog jezika generisanog gramatikom $G = \langle V_N, V_T, P, S\rangle$ postoji najlevlje izvođenje $S \rightarrow_G^{*l} w$.

Dokaz. Dokaz ćemo sprovesti indukcijom po dužini izvođenja reči w iz proizvoljnog $A \in V_N$. Ako se w direktno izvodi iz A, odnosno $A \rightarrow_G w$, to je istovremeno i najlevlje izvođenje. Neka je tvrđenje tačno za sva izvođenja dužine najviše $k \geq 1$ i neka izvođenje $A \rightarrow \alpha_1 \rightarrow_G^* w$ ima $k+1$ korak. Neka je $\alpha_1 = B_1B_2 \ldots B_m$, gde su svi B_i iz V. Prvi korak prethodnog izvođenja je $A \rightarrow B_1B_2 \ldots B_m$. Reč w možemo zapisati u obliku $w = u_1u_2 \ldots u_m$, pri čemu važi $B_i \rightarrow_G^* u_i$. Dužina svakog od ovih izvođenja je najviše k, pa po induktivnoj hipotezi postoji najlevlje izvođenje $B_i \rightarrow_G^{*l} u_i$, za $1 \leq i \leq m$. Posmatrajmo sada izvođenje koje počinje korakom $A \rightarrow B_1B_2 \ldots B_m$ kojeg sledi $B_1B_2 \ldots B_m \rightarrow_G^{*l} u_1B_2 \ldots B_m \rightarrow_G^{*l} u_1u_2B_3 \ldots B_m \rightarrow_G^{*l} \cdots \rightarrow_G^{*l} u_1u_2 \ldots u_m$. Pošto su svi $u_i \in V_T^*$, ovo je najlevlje izvođenje w is A. Ξ

Ako je negde potrebno naglasiti, sa $\rightarrow_G^{*l}$, odnosno $\rightarrow_G^{*d}$, ćemo označavati najlevlje, odnosno desno, izvođenje. Slično, sa $\rightarrow^l$, odnosno $\rightarrow^d$, ćemo označavati jedan korak najlevljeg, odnosno desnog, izvođenja.



Slika 9.4. Drveta za izvođenja iz primera 9.4.4.

.

9.4.2 Višeznačne gramatike

Kontekstno slobodna gramatika G je *višeznačna* ako postoje bar dva međusobno različita najlevlja izvođenja bar jedne reči $w \in L(G)$. Ako G nije višeznačna, ona je *jednoznačna*.

Primer 9.4.4 Gramatika iz primera 9.4.1 je višeznačna jer postoje dva najlevlja izvođenja za reč $id * id + id$:

$$E \to E * E \to id * E \to id * E + E \to id * id + E \to id * id + id \text{ i}$$

$$E \to E + E \to E * E + E \to id * E + E \to id * id + E \to id * id + id$$

kojima odgovaraju drveta izvođenja na slici 9.4. Ξ

Postoje kontekstno slobodni jezici čije odgovarajuće gramatike moraju biti višeznačne.

Primer 9.4.5 Za jezik $L = \{a^i b^j c^k : i = j \text{ ili } j = k\}$ se može pokazati da je kontekstno slobodan. Svaka gramatika G za koju je $L = L(G)$ mora imati jednu vrstu najlevljeg izvođenja za reči u kojima je $i = j$, a drugu za reči u kojima je $j = k$, pa postoje dve vrste najlevljih izvođenja za reči u kojima je $i = j = k$. Ξ

9.4.3 Normalne forme

Tvrđenja 9.4.7 i 9.4.8 govore da se za proizvoljnu kontekstno slobodnu gramatiku može podrazumevati da sadrži samo pravila izvođenja posebnog, jednostavnijeg, oblika. U prvom slučaju se kaže da je gramatika u *normalnoj formi Čomskog*, a u drugom da je u *Grajbahovoj normalnoj formi*. Obe ove forme se koriste u raznim dokazima kada je pogodno izvođenje predstaviti u nekom posebnom obliku. Međutim, najpre je potrebno dokazati jedno pomoćno tvrđenje.

Teorema 9.4.6 Za svaku kontekstno slobodnu gramatiku G_1 postoji njoj ekvivalentna gramatika G_2 u kojoj nema pravila izvođenja oblika $A \to B$, gde su A i B neterminali.

Dokaz. Neka je $G_1 = \langle V_{N_1}, V_{T_1}, P_1, S_1 \rangle$. Skup pravila P_2 gramatike G_2 će sadržati sva pravila koja nisu oblika $A \to B$ i sva pravila $A \to \alpha$ za koja je $A \to^*_{G_1} B$, $B \to \alpha \in P_1$ i $\alpha \notin V_{N_1}$. Da bi se ispitalo da li $A \to^*_{G_1} B$ dovoljno je proveriti izvođenja koja nisu duža od $|V_{N_1}|$. Sa G_2 označimo gramatiku $\langle V_{N_1}, V_{T_1}, P_2, S_1 \rangle$. Očigledno je da ako $S_1 \to^*_{G_2} w$ da je $S_1 \to^*_{G_1} w$ jer se na svim mestima gde se primene nova pravila $A \to \alpha$, najpre primeni izvođenje $A \to^*_{G_1} B$, a zatim pravilo $B \to \alpha$. Sa druge strane, neka je

$$S_1 \to \alpha_1 \to \ldots \to \alpha_{i-1} \to \alpha_i \to \alpha_{i+1} \to \ldots \to \alpha_{j-1} \to \alpha_j \to \alpha_{j+1} \to \ldots \to w$$

jedno najlevlje izvođenje u gramatici G_1 takvo da:

- $\alpha_i \to \alpha_{i+1}$ i $\alpha_j \to \alpha_{j+1}$ nisu izvođenja tipa $A \to B$,
- izvođenja $\alpha_{i+1} \to \alpha_{i+2}$, $\ldots$, $\alpha_{j-1} \to \alpha_j$ to jesu.

Reči α_{i+1}, α_{i+2}, $\ldots$, α_j su sve iste dužine, a pošto je izvođenje najlevlje, znači da se promene vrše uvek na istom mestu u njima. Međutim, tada se primenom nekog od pravila iz $P_2 \setminus P_1$ može direktno dobiti $\alpha_{i+1} \to \alpha_{j+1}$, pa je $L(G_1) = L(G_2)$. Ξ

Teorema 9.4.7 (Normalna forma Čomskog) Svaki kontekstno slobodni jezik se može generisati gramatikom u kojoj su sva pravila oblika $A \to a$ ili $A \to BC$ gde su A, B i C neterminali i a terminal.

Dokaz. Na osnovu teoreme 9.4.6 možemo pretpostaviti da razmatrana gramatika zadovoljava:

- ako je pravilo izvođenje oblika $A \to \alpha$ i $|\alpha| = 1$, onda je $\alpha \in V_T$, odnosno ovakva pravila zadovoljavaju traženi oblik iz tvrđenja,
- jedino pravilo $A \to \varepsilon$ je za $A = S$ i
- polazni znak S se ne javlja na desnoj strani ni jednog pravila izvođenja.

Za svako pravilo $A \to B_1 B_2 \ldots B_m$ uvodimo novo pravilo oblika $A \to B'_1 B'_2 \ldots B'_m$, gde je $B'_i = B_i$, ako je $B_i \in V_N$, inače je B'_i novi neterminal, kada dodajemo i pravilo $B'_i \to B_i$. Trivijalno je da su ove gramatike ekvivalentne. Konačno, za svako pravilo $A \to B_1 B_2 \ldots B_m$ u kome su svi $B_i \in V_N$ ponovo uvodimo nove neterminalne simbole D_1, $\ldots$, D_{m-2} i pravila $A \to B_1 D_1$, $D_1 \to B_2 D_2$, $\ldots D_{m-2} \to B_{m-1} B_m$. Ponovo se lako pokazuje da su dve gramatike ekvivalentne. Ξ

Sličnim postupkom se dokazuje i sledeće tvrđenje.

Teorema 9.4.8 (Grajbahova normalna forma) Svaki kontekstno slobodni jezik se može generisati gramatikom u kojoj su sva pravila oblika $A \to a\alpha$, gde je A neterminal, a terminal i α reč koja se sastoji samo od neterminala, $|\alpha| \geq 0$.

Na osnovu teoreme 9.4.7 o normalnoj formi Čomskog, pokazuje se da se za svaku konkretnu reč dužina izvođenja može ograničiti.

Teorema 9.4.9 Neka je G kontekstno slobodna gramatika u normalnoj formi Čomskog i reč $w \in L(G)$ takva da je $|w| \geq 1$. Svako izvođenje $S \to^*_G w$ sadrži tačno $2|w| - 1$ primena pravila izvođenja.

Dokaz. Zbog oblika pravila u gramatici G, svaki od $|w|$ terminala u reči w dobijen je od tačno jednog neterminalnog simbola zašto je potrebno $|w|$ primena pravila izvođenja. Takođe, za generisanje $|w|$ neterminala je potrebno $|w| - 1$ primena pravila izvođenja, pošto svako pravilo povećava dužinu reči za jedan znak. Ξ

9.4.4 Potisni automati

U teoremi 9.3.5 u kojoj je pokazana veza regularnih jezika i konačnih automata neterminali su predstavljali stanja, dok su terminalima označavane ivice koje ih povezuju. U slučaju kontekstno slobodnih jezika pravila su oblika $A \to xBy$ gde su $A \in V_N$, $B \in V_N \cup \{\varepsilon\}$, $x \in V_T^*$ i $y \in V^*$. Prema analogiji sa teoremom 9.3.5 u apstraktnoj mašini koja će odgovarati kontekstno slobodnom jeziku niz terminala x će usloviti prelazak iz stanja A u stanje B, dok će reč y biti sačuvana za naknadnu analizu, nakon što se obrade sva pravila u kojima je neterminalni znak B sa leve strane. Pošto su i ova pravila spomenutog oblika jedan deo njihove desne strane se takođe sklanja i čuva za kasnije korištenje, do koga svakako mora doći pre nego se razmatra reč y. Dakle, organizacija memorije u kojoj se čuvaju delovi desnih strana pravila je po principu 'poslednji stavljen, prvi uzet', odnosno po principu steka.

Definicija 9.4.10 *(Nedeterministički) potisni automat*[12] je uređena šestorka $M = \langle S, s, F, A, \Gamma, \delta \rangle$, gde su:

- S konačan skup stanja,
- $s \in S$ početno stanje,
- $F \subset S$ skup završnih stanja,
- A alfabet ulaznih znakova,
- Γ alfabet znakova na steku,
- $\delta : (S \times (A \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\})) \to \mathbb{P}_{fin}(S \times \Gamma^*)$ funkcija prelaza koja uređenoj trojci koja se sastoji od stanja, ulaznog znaka i znaka na vrhu steka pridružuje neki konačan podskup skupa svih uređenih parova koji sadrže stanje i novi opis vrha steka.

Rad potisnog automata se shvata na sledeći način. Na početku rada automat se nalazi i stanju s, a stek je prazan. Čitajući ulaznu reč, automat prelazi u novo stanje i menja sadržaj steka u skladu sa funkcijom prelaza. U nekoj opštoj situaciji, promena stanja se sprovodi i u zavisnosti od sadržaja na vrhu steka.

Primer 9.4.11 Ako je $(q, y) \in \delta(p, a, x)$, onda se iz stanja p, ako je na vrhu steka x, a čita se a, prelazi u stanje q, dok se dotadašni vrh steka zamenjuje sa y. Ako je $(q, y) \in \delta(p, a, \varepsilon)$, onda se iz stanja p, ako se čita a prelazi u stanje q i dodaje y na vrh steka. Ako je $(q, \varepsilon) \in \delta(p, a, x)$, onda se iz stanja p, ako je na vrhu steka x i čita se a, prelazi u stanje q, dok se x skida sa vrha steka. Ξ

[12] *Pushdown automata. Automat sa stekom.*

Primetimo da se zbog oblika funkcije prelaza δ definicijom uvodi nedeterministički potisni automat. Kod determinističkih potisnih automata funkcija prelaza δ se ograničava tako da

- za svako $q \in S$, $a \in A \cup \{\varepsilon\}$ i $\gamma \in \Gamma \cup \{\varepsilon\}$, $\delta(q, a, \gamma)$ je najviše jednočlan skup i
- ako je $\delta(q, \varepsilon, \gamma)$ neprazan skup, onda je $\delta(q, a, \gamma)$ prazan skup za svako $a \neq \varepsilon$,

odnosno za svako stanje, ulazni znak i vrh steka postoji najviše jedna akcija kojom se menja stanje i vrh steka, a ako se prazna reč ε koristi kao pravi ulazni simbol, onda ne postoji konflikt u izboru akcije u odnosu na neki znak alfabeta. Za razliku od konačnih automata, klase jezika koji odgovaraju nedeterminističkim i determinističkim potisnim automatima nisu jednake. Zato ćemo u nastavku, kada se govori o potisnim automatima, podrazumevati da se radi o nedeterminističkim automatima, dok ćemo posebno naglasiti kada bude reč o determinističkim potisnim automatima.

Konfiguracija je uređena trojka (q, w, y), gde je $q \in S$ stanje i $w \in A^*$ ulazna reč na alfabetu A i $y \in \Gamma^*$ opis steka tako da je prvi znak reči y na vrhu steka, drugi znak reči y je na steku odmah ispod vrha itd. *Računski korak* je svaki par konfiguracija $(q, ux, \alpha y)$, $(q', x, \beta y)$ za koje postoji $(q', \beta) \in \delta(q, u, \alpha)$. *Izračunavanje* je niz konfiguracija sa osobinom da svake dve uzastopne konfiguracije čine računski korak. Sa $(q, w, x) \vdash_M (p, u, y)$ ćemo označiti da postoji izračunavanje automata M čiji je prvi element konfiguracija (q, w, x), a poslednji (p, u, y).

Potisni automat M *prihvata* reč $w \in A^*$, ako za neko završno stanje $q \in F$ važi $(s, w, \varepsilon) \vdash_M (q, \varepsilon, \varepsilon)$. $L(M)$ će označavati jezik, tj. skup svih reči koje prihvata potisni automat M. Ako u bilo kom koraku ne postoji definisan prelaz koji odgovara tekućem stanju, reči koja se čita i sadržaju vrha steka ili se učitavanje završi pre nego se isprazni stek, ulazna reč se ne prihvata.

Primer 9.4.12 Neka je $M = \langle S, s, F, A, \Gamma, \delta \rangle$, potisni automat tako da $S = \{s, f\}$, $F = \{f\}$, $A = \{a, b, c\}$, $\Gamma = \{a, b\}$ i $\delta = \{((s, a, \varepsilon), (s, a)), ((s, b, \varepsilon), (s, b)), ((s, c, \varepsilon), (f, \varepsilon)), ((f, a, a), (f, \varepsilon)), ((f, b, b), (f, \varepsilon))\}$. Pretpostavimo da se na ulazu nalazi reč *abbcbba*. Sledeća tabela ilustruje izračunavanje potisnog automata M koji prihvata ovu reč:

stanje	ulazna reč	stek	primenjen korak
s	*abbcbba*	ε	1
s	*bbcbba*	a	2
s	*bcbba*	*ba*	2
s	*cbba*	*bba*	3
f	*bba*	*bba*	5
f	*ba*	*ba*	5
f	a	a	4
f	ε	ε	

Zapravo, može se dokazati da je $L(M) = \{wcw^{-1} : w \in \{a, b\}^*\}$. Ξ

Teorema 9.4.13 Klasa jezika koje prihvataju potisni automati jednaka je klasi kontekstno slobodnih jezika.

Dokaz. ($\Leftarrow$) Neka je $G = \langle V_N, V_T, P, S\rangle$ kontekstno slobodna gramatika u Grajbahovoj formi, tako da su sva izvođenja oblika $A \to a\alpha$, $A \in V_N$, $a \in V_T$, $\alpha \in V_N^*$. Neka je $\varepsilon \in L(G)$. Definišimo potisni automat $M = \langle \{p, q\}, p, \{q\}, V_T, V_N, \delta\rangle$ tako da:

- $\delta(p, \varepsilon, \varepsilon) = (q, S)$ i
- $\delta(q, a, A) = \{(q, \alpha) : A \to a\alpha \in P\}$.

Posmatrajmo reč $w \in L(G)$ i neko njeno najlevlje izvođenje. Izračunavanje automata M započinje postavljanjem znaka S na vrh praznog steka u skladu sa računskim korakom oblika $((p, \varepsilon w, \varepsilon), (q, w, S))$, gde je $w = vau$. Za svaki korak izvođenja oblika $xA\beta \vdash_G xa\alpha\beta$ u kome je primenjeno pravilo izvođenja $A \to a\alpha$ i gde su $x \in V_N^*$, $A \in V_N$, $a \in V_T$ i $\alpha, \beta \in V_T^*$ potisni automat M će napraviti računski korak oblika $((q, au, A\beta), (q, u, \alpha\beta))$.

Da bismo pokazali da je $L(G) = L(M)$ dovoljno je primetiti da važi $xA\beta \to_G^* xy\alpha\beta$ ako i samo ako za M važi $(q, y, A\beta) \vdash_M (q, \varepsilon, \alpha\beta)$. Poslednje tvrđenje se pokazuje indukcijom po dužini izvođenja. Sada neposredno sledi $S \to_G^* w$ ako i samo ako $(p, \varepsilon w, \varepsilon) \vdash (q, w, S) \vdash_M (q, \varepsilon, \varepsilon)$, odnosno $L(G) = L(M)$.
($\Rightarrow$) Neka je $M = \langle S, s, F, A, \Gamma, \delta\rangle$ potisni automat i $L = L(M)$ i neka je S_0 novi znak. Definišimo kontekstno slobodnu gramatiku $G = \langle V_N, A, P, S_0\rangle$ gde su

- $V_N = \{(q, B, p) : q, p \in S, B \in \Gamma \cup \{\varepsilon\}\} \cup \{S_0\}$ i
- $P = \{S_0 \to (s, \varepsilon, q) : q \in F\} \cup \{(q, B, p) \to a(q_1, B_1, q_2)(q_2, B_2, q_3)\dots(q_m, B_m, q_{m+1}) : q_{m+1} = p, (\forall q_1, \dots q_{m+1} \in S), a \in A \cup \{\varepsilon\}, B, B_1, \dots, B_m \in \Gamma, (q_1, B_1 \dots B_m) \in \delta(q, a, B)\}$.

Primetimo da, ako je $m = 0$ u definiciji pravila izvođenja, onda je $q_1 = p$, $(p, \varepsilon) \in \delta(q, a, B)$ i $(q, B, p) \to a \in P$. Treba uočiti da su neterminalni simboli i pravila izvođenja definisani tako da je najlevlje izvođenje u G reči w simulacija rada potisnog automata M kome se na ulazu nalazi reč w. Recimo, pravilo $(q, B, p) \to a(q_1, B_1, q_2)(q_2, B_2, q_3)\dots(q_m, B_m, p)$ odgovara koraku rada potisnog automata određenog sa $(q_1, B_1 \dots B_m) \in \delta(q, a, B)$, tako što prvi znak na desnoj strani pravila znači generisanje sledećeg terminalnog znaka, a preostali deo desne strane pravila simulira uklanjanje sadržaja sa steka jer neterminalni simboli koji se pojavljuju u najlevljem izvođenju odgovaraju simbolima steka u trenutku do kog je potisni automat pročitao onaj deo ulazne reči koji je istovremeno proizvelo najlevlje izvođenje.

Indukcijom po dužini izvođenja se pokazuje da $(q, B, p) \to_G^* w$ ako i samo ako $(q, w, B) \vdash_M (p, \varepsilon, \varepsilon)$. Neka izračunavanje $(q, w, B) \vdash_M (p, \varepsilon, \varepsilon)$ ima k koraka. Ako je $k = 1$, važi da je $w \in V_T$ ili $w = \varepsilon$, pa $\delta(q, B, p)$ mora sadržati (p, ε) i P sadrži pravilo $(q, B, p) \to w$, zbog čega je $(q, B, p) \to_G^* w$. Dalje, neka je indukcijska hipoteza tačna za sva izračunavanja dužine najviše $k - 1$. Prvi korak izračunavanja mora biti oblika $(q, a, B) \vdash_M (q_1, \varepsilon, B_1B_2 \dots B_l)$, gde je a prvi znak u w ili $a = \varepsilon$. Reč w mora biti oblika $aw_1w_2 \dots w_l$, tako da za svaki i $(1 \le i \le l)$ važi $(q_i, w_i, B_i) \vdash_M (q_{i+1}, \varepsilon, B_1B_2 \dots B_l)$ u manje od k koraka, gde su $q_1, q_2, \dots, q_l \in S$ i $q_{l+1} = p$. Na osnovu induktivne hipoteze imamo da je $(q_i, B_i, q_{i+1}) \to_G^* w_i$. Kako je $(q, B, p) \to a(q_1, B_1, q_2)(q_2, B_2, q_3)\dots(q_l, B_l, q_{l+1}) \in P$, imamo da $(q, B, p) \to_G^* aw_1w_2 \dots w_l = w$. Indukcija u drugom smeru se sprovodi na analogni način.

Sada primetimo da, ako je $w \in L(G)$, onda $S_0 \to (s, \varepsilon, q) \to_G^* w$ za neko stanje q. Tada, $(s, w, \varepsilon) \vdash_M (q, \varepsilon, \varepsilon)$ i $w \in L(M)$. Slično, $w \in L(M)$ implicira da $(s, w, \varepsilon) \vdash_M (q, \varepsilon, \varepsilon)$, odakle je $S_0 \to (s, \varepsilon, q) \to_G^* w$, pa neposredno sledi da je $L(M) = L(G)$. Ξ

9.4.5 Svojstva kontekstno slobodnih jezika

Sledeće tvrđenje navodi neke osobine kontekstno slobodnih jezika koje su imali i regularni jezici.

Teorema 9.4.14 Klasa kontekstno slobodnih jezika je zatvorena za:

1. uniju,
2. nadovezivanje i
3. zatvorenje (Klinijevu zvezdicu).

Dokaz. Neka su G_1 i G_2 kontekstno slobodne gramatike za koje je $L_1 = L(G_1)$ i $L_2 = L(G_2)$. Pretpostavimo da su skupovi neterminalnih simbola ovih gramatika disjunktni.
(1) Uniji jezika $L(G_1)$ i $L(G_2)$ odgovara gramatika koja ima dodatni polazni znak S i pravila $S \to S_1$, $S \to S_2$ kojima se dobijaju početni simboli gramatika G_1 i G_2, a skupovi neterminala, terminala i pravila su unije odgovarajućih skupova polaznih gramatika.
(2) Razlika u odnosu na prethodni slučaj je u tome što je jedino novo pravilo oblika $S \to S_1 S_2$.
(3) Ako je G_1 polazna gramatika, onda jeziku $L(G_1)^*$ odgovara gramatika G koja se od polazne razlikuje samo utoliko što ima novi polazni znak S koji se ne javlja u G_1 i dodatna pravila izvođenja: $S \to \varepsilon$, $S \to S_1$ i $S_1 \to S_1 S_1$. Ξ

Međutim, klasa kontekstno slobodnih jezika nije zatvorena za komplement i presek, što ćemo ilustrovati u teoremi 9.4.17.

Sledeći problemi su odlučivi za kontekstno slobodne jezike, bez obzira da li su opisani gramatikom tipa 2 ili potisnim automatom:

- da li data reč pripada nekom određenom kontekstno slobodnom jeziku, jer je u teoremi 9.4.7 pokazano da za svaku kontekstno slobodnu gramatiku postoji njoj ekvivalentna kontekstno slobodna gramatika u normalnoj formi Čomskog, pa, prema teoremi 9.4.9, bilo koje izvođenje reči w za koju je $|w| \geq 1$ ima dužinu tačno $2|w| - 1$, odnosno dovoljno je posmatrati samo konačno mnogo izvođenja konačne dužine i time neposredno proveriti da li je w u jeziku,
- da li je jezik prazan skup, jer je jezik neprazan skup ako su svi listovi terminalni simboli u bar jednom drvetu izvođenja na čijoj najdužoj grani ima najviše $|V_N| + 1$ čvorova u kojima nema ponavljanja neterminalnih simbola i

- da li je jezik beskonača skup, jer jezik to jeste samo ako sadrži reč dužine između $p^{|V_N|}+1$ i $p^{|V_N|+1}$, gde je p najveći broj znakova sa desne strane bilo kog pravila izvođenja u gramatici, što se dokazuje na sličan način kao i teorema 9.4.15.

Sa druge strane nisu odlučivi problemi:

- da li je presek dva kontekstno slobodna jezika prazan,
- da li su jednaka dva kontekstno slobodna jezika i
- da li je data kontekstno slobodna gramatika višeznačna.

9.4.6 Jezici koji nisu kontekstno slobodni

Postupkom naduvavanja se može pokazati i da neki jezik nije kontekstno slobodan.

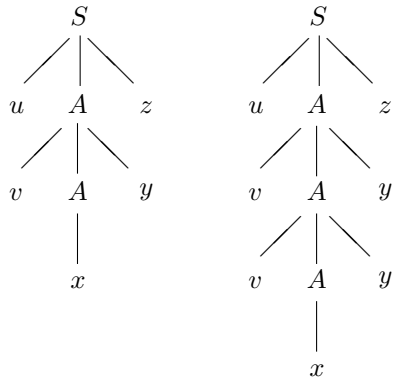
Teorema 9.4.15 Neka je $G = \langle V_N, V_T, P, S\rangle$ kontekstno slobodna gramatika. Tada postoji broj k koji zavisi od G takav da se svaka reč w iz $L(G)$ za koju je $|w| > k$ može napisati u obliku $w = uvxyz$ tako da su v ili y neprazne reči i za svako $n \geq 0$, $uv^nxy^nz \in L(G)$.

Dokaz. Dovoljno je pokazati da postoji $k = k(G)$ tako da za svaku reč dužu od k postoji izvođenje oblika $S \to_G^* uAz \to_G^* uvAyz \to_G^* uvxyz$, za $u, v, x, y, z \in V_T^*$, $A \in V_N$, pri čemu su v ili y neprazne reči, tj. $vy \neq \varepsilon$. Tada se izvođenje $A \to_G^* vAy$ može proizvoljno ponavljati čime se generiše reč oblika uv^nxy^nz. Neka su $m = |V_N| + 1$, p najveći broj znakova sa desne strane bilo kog pravila izvođenja u G, i $k = p^{m-1}$. U odeljku 9.4.1 je objašnjeno da, ako je reč iz jezika duža od p^{m-1}, onda u odgovarajućem drvetu izvođenja postoji grana sa bar $m+1$ čvorom, pa se na njoj mora bar dva puta pojaviti isti neterminalni znak. Neka je to znak A. Listovi podrveta čiji je koren A daju reč oblika vAy. Ako bi bilo $v = y = \varepsilon$, ovo podrvo se može ukloniti tako što se odseče deo između dve pojave znaka A. Ako bi se to ponovilo za sve grane na kojima se neki čvor javlja bar dva puta, dobili bismo drvo čija visina ne prelazi m što je nemoguće, jer je $|w| > p^{m-1}$. Zato je bar jedna od reči v i y neprazna. Sada je jasno da se uočeno poddrvo može ponavljati proizvoljan broj puta, kao na slici 9.5, čime se dobija traženo tvrđenje. Ξ

Primer 9.4.16 Primenjujući teoremu 9.4.15 pokazujemo da jezik $\{a^nb^nc^n : n \geq 0\}$ nije kontekstno slobodan. Neka je n izabran tako da je reč $w = a^nb^nc^n$ duža od broja k preciziranog u teoremi 9.4.15. Tada je $w = uvxyz$ i $v \neq \varepsilon$ ili $y \neq \varepsilon$. Lako se vidi da ma kakav izbor reči v i y bio, napumpavanjem dobijamo reč u kojoj redosled simbola nije takav da su svi znaci a pre svih znakova b koji su opet pre svih znakova c ili nije jednak broju pojava znakova a, b i c. Ξ

Teorema 9.4.17 Klasa kontekstno slobodnih jezika nije zatvorena za komplement i presek.

Dokaz. Dovoljno je uočiti da je presek kontekstno slobodnih jezika $\{a^nb^nc^m : n, m \geq 0\}$ i $\{a^mb^nc^n : n, m \geq 0\}$ jezik $\{a^nb^nc^n : n \geq 0\}$ koji nije kontekstno slobodan. Zbog toga klasa nije zatvorena ni za komplementiranje, jer da jeste, zbog zatvorenosti za uniju, klasa bi bila zatvorena i za presek. Ξ



Slika 9.5. Primer naduvavanja reči u kontekstno slobodnom jeziku.

9.4.7 Determinizam, $LR(k)$-gramatike i sintaksna analiza

Prevodilac je program koji kao ulaz dobija izvorni program pisan u nekom jeziku visokog nivoa, poput C-a ili PASCAL-a, i prevodi ga u ekvivalentan program na nekom od jezika niskog nivoa, poput mašinskog ili simboličkog jezika. U radu svakog prevodioca postoji više faza. U leksičkoj analizi, o kojoj je bilo reči u odeljku 9.3.6, izdvajaju se celine koje odgovaraju identifikatorima, konstantama, ključnim rečima itd. Sintaksna analiza se odnosi na utvrđivanje korektnosti konstrukcija kao što su naredbe, definicije potprograma i sl. na osnovu čega se kasnije pristupa generisanju koda i, konačno, optimizaciji. U sintaksnoj analizi se pre svega vrši simuliranje rada potisnog automata koji odgovara gramatici programskog jezika. Međutim, potisni automati koje smo do sada koristili su nedeterministički i kao takvi nisu praktični za simuliranje, za razliku od determinističkih konačnih automata koji se primenjuju u leksičkoj analizi.

Nazovimo jezik $L \subset V_T^*$ *determinističkim kontekstno slobodnim jezikom* ako je $L = L(M)$ za neki deterministički potisni automat M. Klasa determinističkih kontekstno slobodnih jezika je zatvorena za komplement, a pošto klasa kontekstno slobodnih jezika to nije, postoje kontekstno slobodni jezici koji nisu deterministički. Takođe, odlučiv je problem jednakosti dva deterministička kontekstno slobodna jezika.

Činjenica da klase jezika koje prepoznaju nedeterministički, odnosno deterministički potisni automati nisu jednake naizgled otežava prevođenje programa korištenjem metoda povezanih sa kontekstno slobodnim jezicima. Na sreću, za većinu programskih jezika se pokazalo da postoje deterministički potisni automati koji prihvataju sve sintaksno korektne programe. Razvijene su metode koje u mnogim slučajevima kontekstno slobodnu gramatiku transformišu u determinističku kontekstno slobodnu gramatiku.

Za determinističke gramatike postoje programi koji prihvataju opis programskog jezika preko gramatike i generišu sintaksni analizator i generator koda za programe pisane na tom jeziku. Jedan od poznatijih generatora prevodilaca je

Yacc[13]. U opisu jezika koji je ulaz u Yacc koristi se posebna vrsta jednoznačnih kontekstno slobodnih gramatika nazvanih $LR(k)$-gramatike koje generišu upravo determinističke kontekstno slobodne jezike.

Definicija 9.4.18 Neka su $G = \langle V_N, V_T, P, S\rangle$ kontekstno slobodna gramatika, znak $\# \notin V$, $A, B \in V_N$, $\alpha, \beta, \gamma \in V^*$ i $w, w_1, w_2, w_3 \in V_T^*$. G je *$LR(k)$-gramatika* ako za svaku reč $\alpha\beta w_1 w_2$ za koju je $|w_1| = k$ i $S\#^k \rightarrow_G^{*d} \alpha\beta w_1 w_2$ važi: ako je $S\#^k \rightarrow_G^{*d} \alpha A w_1 w_2 \rightarrow^d \alpha\beta w_1 w_2$ i za neku drugu reč $\alpha\beta w_1 w_3$ takvu da $S\#^k \rightarrow_G^{*d} \gamma B w \rightarrow^d \alpha\beta w_1 w_3$, onda je $\gamma = \alpha$, $A = B$ i $w = w_1 w_3$.

Drugim rečima, ako se u desnim izvođenjima dve reči poklapaju na k znakova iza mesta poslednje zamene, onda se i reči u pretposlednjem koraku izvođenja poklapaju od početka zaključno sa k simbola nakon mesta zamene. Odatle, ako imamo reč oblika $\alpha\beta\gamma$, gde su $\alpha, \beta, \gamma \in V^*$ i $|\gamma| = k$, onda se $A \in V_N$ i β mogu jednoznačno odrediti tako da je $S \rightarrow_G^{*d} \alpha A \gamma \rightarrow^d \alpha\beta\gamma$. Tih k znakova nazivamo *znaci koji se gledaju unapred*[14].

Ovakva osobina gramatika je pogodna za sintaksnu analizu. Pretpostavimo da je data reč $w \in V_N^*$ za koju želimo ispitati da li pripada nekom jeziku. Jedan način za to je da pokušamo konstruisati sve zamene podreči reči w neterminalima koje su opisane pravilima izvođenja gramatike. Zatim se postupak ponavlja, dok god se ne stigne do startnog simbola. Problem nastaje jer je u svakom koraku, u principu, moguće izvršiti više izmena što vodi nedeterminizmu. Uslovi koji se postavljaju za $LR(k)$-gramatike obezbeđuju da se ovaj postupak obavlja deterministički. Primetimo da se u definiciji $LR(k)$ gramatika zahteva da se sa desne strane reči nalazi k znakova # tako da se postupak rekonstrukcije izvođenja može započeti sa desnog kraja reči.

Neke osobine LR-gramatika su navedene u teoremi 9.4.19.

Teorema 9.4.19

1. Svaka $LR(k)$-gramatika je jednoznačna.
2. Za datu gramatiku G i broj k problem da li je G $LR(k)$-gramatika je odlučiv.
3. Jezici generisani $LR(k)$-gramatika su deterministički.
4. Svaki deterministički jezik je generisan nekom $LR(1)$-gramatikom.

Na slici 9.6 je dat primer jednog opisa koji se zadaje Yaccu. Reč je o implementaciji jednostavnog kalkulatora koji na ulazu dobija liniju u kojoj je upisan aritmetički izraz. Izraz se sastoji od prirodnih brojeva i znakova $+$, $*$, (i). Rezultat izračunavanja se štampa. Opis se sastoji iz tri dela. U prvom delu se nalaze makro komanda za uključivanje standardnog C-zaglavlja *ctype.h* i deklaracija tokena *BROJ*. Ovde pretpostavljamo da se tokeni izdvajaju nekim leksičkim analizatorom poput onih generisanih programom Lex. U drugom delu je dat opis kontekstno slobodne gramatike koja generiše jezik aritmetičkih izraza. Njena pravila izvođenja su:

- $E \rightarrow E + T$,

[13]Yet another compiler compiler. Program je javan i predstavlja deo distibucije operativnog sistema Unix.

[14]Lookahead symbols.

```
%{
# include <ctype.h>
%}
% token BROJ
%%
l : e '\n'  { printf("%d\n", $1); }
  ;
e : e '+' t { $$ = $1 + $3; }
  | t
  ;
t : t '*' f { $$ = $1 * $3; }
  | f
  ;
f : '(' e ')' { $$ = $2; }
  | BROJ
  ;
%%
/* pomocne funkcije */
```

Slika 9.6. Deo opisa koji dobija Yacc.

- $E \to T$,
- $T \to T * F$,
- $T \to F$,
- $F \to (E)$,
- $F \to BROJ$.

Primetimo da ova gramatika, za razliku od slične iz primera 9.4.1, nije višeznačna. U nastavku svakog pravila izvođenja se zadaju odgovarajuće akcije. Recimo, vrednost izraza u kojem je centralni operator znak + jednaka je zbiru operanada, kao što je zapisano u akciji za prvi red drugog pravila. U sistemu je predefinisano da \$\$ odgovara vrednosti tokena sa leve strane pravila, dok je $\$i$ vrednost i-tog tokena sa desne strane pravila. Prvo pravilo u opisu označava da je ulaz sistema izraz za kojim sledi znak za novi red i da je odgovarajuća akcija štampanje rezultata. Poslednji deo opisa, kao i u slučaju sistema Lex, čine definicije pomoćnih funkcija koje se koriste u akcijama.

Kontekstno slobodnim gramatikama se ipak ne mogu opisati sva svojstva programskih jezika, recimo zahtev da je identifikator definisan pre upotrebe. Jednu apstraktnu formulacija ovog problema predstavlja jezik $\{wcw : w \in (a \cup b)^*\}$ koji nije kontekstno slobodan i u kome prva pojava reči w predstavlja definiciju, a druga upotrebu identifikatora čije se ime sastoji od proizvoljnog broja znakova kako je uobičajeno u savremenim programskim jezicima. Slično, jezik $\{a^n b^n c^n\}$ koji takođe nije kontekstno slobodan predstavlja apstraktni zapis zahteva da broj argumenata

funkcije pri definiciji, deklaraciji i pozivu mora biti isti. Odatle, sintaksni analizator prihvata jedan nadskup skupa ispravnih konstrukcija koji se dodatno obrađuje u daljim fazama prevođenja.

9.5 Kontekstno osetljivi jezici i linearno ograničeni automati

Kontekstno osetljivi jezici su generisani gramatikama za čija pravila važi:

$$\alpha \to \beta,\ |\beta| \geq |\alpha|$$

uz dodatak da se uslov može oslabiti tako da dozvoljava i izvođenje prazne reči. Klasa kontekstno osetljivih jezika je zatvorena za konkatenaciju, Klinijevu zvezdicu, uniju i presek, dok nije poznato da li je zatvorena za komplement.

Primer 9.5.1 U zadatku 11.94 je opisana kontekstno osetljiva gramatika koja generiše jezik $\{a^n b^n c^n : n > 0\}$ koji, kao što je u primeru 9.4.16 pokazano, nije kontekstno slobodan. Ξ

Apstraktne mašine koje odgovaraju ovoj klasi gramatika su linearno ograničeni automati.

Definicija 9.5.2 *Linearno ograničeni automat* je nedeterministička Tjuringova mašina koja koristi samo onaj deo trake u kome su smešteni ulazni podaci.

Zahtev da se glava na traci ne pomera preko kraja ulazne reči se ostvaruje uvođenjem dva nova specijalna znaka početka i kraja ulaza i ograničenjima da bez obzira na stanje, ako se čita znak početka, glava se ne sme pomerati levo, odnosno za znak kraja, glava se ne sme pomerati udesno.

Teorema 9.5.3 Jezik L je kontekstno osetljiv ako i samo ako postoji linearno ograničeni automat M tako da je $L = L(M)$.

Za sada se ne zna da li za svaki kontekstno osetljiv jezik postoji deterministički linearno ograničeni automat koji ga prihvata.

Teorema 9.5.4 Ako je gramatika $G = \langle V_N, V_T, P, S\rangle$ kontekstno osetljiva, ona je odlučiva.

Dokaz. Na osnovu teoreme 9.2.6, pretpostavimo da $L(G)$ ne sadrži praznu reč. Posmatrajmo neku reč w i skupove reči $T_0 = \{S\}$, $T_{i+1} = T_i \cup \{\alpha : |\alpha| \leq |w|, (\exists \beta \in T_i)\beta \to_G \alpha\}$ takvih da T_i sadrži sve reči dobijene u najviše i koraka izvođenja. Svaki od skupova T_i je konačan jer su reči u njima nizovi znakova ograničene dužine. Zbog ograničenja za oblik pravila kontekstno osetljivih gramatika u kojima se ne mogu dobijati kraće reči, u jednom trenutku će biti $T_i = T_{i+1} = T_{i+2} = \ldots$ Upoređivanjem reči w sa rečima iz skupa T_i se proverava da li $w \in L(G)$ ili $w \notin L(G)$. Ξ

Prema tome, klasa kontekstno osetljivih jezika je potklasa klase odlučivih jezika. Sledeća teorema pokazuje da je potklasa prava, odnosno da postoji odlučiv jezik koji nije kontekstno osetljiv.

Teorema 9.5.5 Postoji rekurzivan jezik na alfabetu $A = \{1\}$ koji nije kontekstno osetljiv.

Dokaz. Dokaz se sprovodi postupkom dijagonalizacije. Sve kontekstno osetljive gramatike sa skupom $V_T = \{1\}$ možemo kodirati rečima alfabeta $\{1, V, b, \rightarrow, /\}$. Znak 1 odgovara jedinom terminalnom simbolu. Svaki znak iz V_N se kodira jednoznačno u obliku Vb^j. Pravilo se kodira tako što se kodiraju leva i desna strana i razdvoje znakom $\rightarrow$, dok se / koristi za razdvajanje pravila. Dakle, svih kontekstno osetljivih gramatika sa skupom $V_T = \{1\}$ ima prebrojivo mnogo i možemo ih poređati u niz $G_1, G_2, \ldots$ Definišimo jezik $L = \{1^n : 1^n \notin L(G_n)\}$. Lako se vidi da iz pretpostavke da je $L = L(G_i)$ za neko i sledi da $1^i \in L$ ako i samo ako $1^i \notin L$. Dakle, ovaj jezik nije kontekstno osetljiv. Sa druge strane, jezik jeste rekurzivan jer postoji postupak koji za dato n konstruiše G_n i $L(G_n)$, a prema teoremi 9.5.4 problem $1^n \in L_n$ jeste odlučiv. Ξ

Jedno neodlučivo pitanje za proizvoljnu kontekstno osetljivu gramatiku G je da li je jezik $L(G)$ prazan.

9.6 Gramatike tipa 0 i Tjuringove mašine

Gramatike tipa 0 su najopštije gramatike i njima, u smislu apstraktnih mašina koje prihvataju jezik, odgovaraju Tjuringove mašine. Preciznije, može se dokazati:

Teorema 9.6.1 Neka je L proizvoljan jezik. Ekvivalentna su tvrđenja:

1. Jezik L je parcijalno odlučiv.
2. Postoji Tjuringova mašina M tako da je $L = L(M)$.
3. Postoji gramatika G tako da je $L = L(G)$.

Dokaz. Osnovni deo dokaza se sastoji u tome da se pokaže da se sva izvođenja mogu nabrojati, pa će Tjuringova mašina radeći po širini pre ili posle pronaći odgovarajuće izvođenje. Alternativno, moguće je koristiti Tjuringovu mašinu koja će nedeterministički izabrati pravo izvođenje. U suprotnom smeru, izvođenja gramatika mogu simulirati rad Tjuringovih mašina tako što tekuća reč predstavlja konfiguraciju u nekom koraku rada. Ξ

Klasa jezika generisanih gramatikama tipa 0 je zatvorena za uniju, presek, nadovezivanje i zatvorenje, a nije zatvorena za komplement pošto komplement parcijalno odlučivog skupa ne mora biti parcijalno odlučiv.

Pitanje / tip	3	2d	2	1	0
$L(G)$ prazan/konačan/beskonačan	+	+	+	–	–
$L(G_1) = L(G_2)$	+	+	–	–	–
$L(G_1) \subset L(G_2)$	+	–	–	–	–
$L(G_1) \cap L(G_2)$ prazan/konačan/beskonačan	+	–	–	–	–

Tabela 9.1. (Ne)odlučivost problema za formalne jezike.

Operacija / tip	3	2d	2	1	0
unija	+	+	+	+	+
konkatenacija	+	+	+	+	+
zatvorenje	+	+	+	+	+
presek	+	+	–	+	+
komplement	+	+	–	?	–

Tabela 9.2. Zatvorenost klasa za operacije nad jezicima.

9.7 Hijerarhija Čomskog

Na osnovu prethodno iznetih tvrđenja i primera zaključuje se da je hijerarhija Čomskog prava:

- na osnovu definicija jasno je da je klasa gramatika tipa 3 potklasa klase gramatika tipa 2 koja je potklasa klase gramatika tipa 1 koja je potklasa klase gramatika tipa 0,
- pošto postoje jezici, kakav je recimo $\{a^n b^n : n \geq 1\}$, koji nisu regularni, a jesu kontekstno slobodan, klasa gramatika tipa 3 je prava potklasa klase gramatika tipa 2,
- pošto jezik $\{a^n b^n c^n : n \geq 1\}$ nije kontekstno slobodan, a jeste kontekstno osetljiv, klasa gramatika tipa 2 je prava potklasa klase gramatika tipa 1,
- pošto postoji jezik na alfabetu $\{1\}$ koji nije kontekstno osetljiv, a jeste odlučiv, klasa gramatika tipa 1 je prava potklasa klase gramatika tipa 0.

Konačno, postojanje jezika, kakav je recimo jezik koji sadrži kodove svih totalnih funkcija, koji nisu parcijalno odlučivi već pripadaju nekom stepenu aritmetičke hijerarhije, dozvoljava da se ova hijerarhija proširuje unedogled.

U tabelama 9.1 i 9.2 su sumarno prikazane osobine klasa jezika i odgovarajućih gramatika iz hijerarhije. Brojevi tipova klasa redom označavaju regularne, determinističke kontekstno slobodne, kontekstno slobodne, kontekstno zavisne i parcijalno odlučive jezike. U tabelama znakovi + i – redom označavaju pozitivan i negativan odgovor. U tabeli 9.1 to znači da su problemi odlučivi ili neodlučivi, a u tabeli 9.2 da su klase zatvorene za operacije, odnosno da nisu zatvorene. Znak ? u tabeli 9.2 znači da odgovor još nije poznat.

9.8 Još neke primene formalnih jezika

Pored prevođenja programa postoje i druge primene formalnih jezika u računarstvu. Regularni izrazi se često koriste u pretraživanju teksta ili u skraćivanju komandi školjci operativnog sistema. Na primer, komandom *dir m** se traži spisak svih datoteka koje počinju slovom m iza koga se nalazi nula ili više ASCII-znakova. Formalni jezici se koriste u prepoznavanju oblika. Poznate su, recimo, primene regularnih izrazi u otkrivanju grešaka na štampanim logičkim pločama, kontekstno slobodnih gramatika u prepoznavanju grafičke reprezentacije vrsta hromozoma, pri čemu pravila opisuju oblike delova hromozoma koji se mogu naslanjati jedni na druge itd. U nastavku je opisana jedna primena konačnih automata u metodi provere modela koja se koristi u verifikaciji.

9.8.1 Formalna verifikacija

Verifikacija nekog programa ili logičkog kola podrazumeva proveru da li je konkretna implementacija korektna, odnosno da li ispunjava neke unapred zadate zahteve. Poznati su slučajevi, recimo greška u jedinici za računanje u pokretnom zarezu procesora Pentium sredinom devedesetih godina, padovi satelita koji je NASA slala na Mars u jesen 1999. godine i evropske svemirske letelice Ariana zbog prekoračenja vrednosti nekih numeričkih izraza, u kojima je nekorektna implementacija dovela do teških i skupih udesa. Verifikacija se upravo primenjuje da bi se izbegle takve ili gore štete. Iskustvo je pokazalo da sigurnost verifikacije ne može biti apsolutna, u smislu da garantuje potpunu korektnost, jer se uvek mogu javiti neki spoljni faktori koji utiču na sistem, a koje nije moguće unapred predvideti. Međutim, verifikacija značajno utiče na povećanje pouzdanosti sistema i skraćivanje ciklusa razvoja, pa su i cene savremenih programskih sistema ove namene izuzetno visoke.

Klasične metode testiranja u kojima se analiziraju rezultati svih mogućih ulaznih vrednosti nisu u opštem slučaju primenljive zbog velike složenosti objekata, na primer telekomunikacionih sistema, koji se testiraju.

Primer 9.8.1 Posmatrajmo program:

```
#include <stdio.h>
main()
{
 unsigned x = 0;
 int input;
 while( (input = getchar()) != EOF ) {
   if( input == '0' ) x = x*2;
   else
     if( input == '1' ) x = x*2+1;
     else
       if( input == '\n' ) {
           printf( "%u\n", x );
           x = 0;
       }
 }
}
```

na programskom jeziku C u kome se binarni neoznačeni broj sa ulaza konvertuje u dekadnu vrednost. Na primer, ulazni podatak "1100" iza koga sledi znak za novi red rezultirao bi izlazom "12". Greška u ovom programu je posledica ograničene dužine registara u kojima se pamte celi brojevi, a koji obično sadrže 32 bita, tako da se sve aritmetičke operacije obavljaju zapravo po modulu 2^{32}. U klasičnim metodima provere ispravnosti ispitivali bi se redom svi binarni nizovi, što bi i u ovom jednostavnom slučaju dugo trajalo. U složenijim situacijama potrebno vreme za takvu proveru nije prihvatljivo sa stanovišta korisnika. Varijanta ovog pristupa podrazumeva da se ne koriste svi ulazni podaci, već da se na neki inteligentan način izaberu test primeri i da se tako skrati dužina rada. Međutim, pronalaženje pravilnosti u pojavama grešaka je često teško. Ξ

Formalnu verifikaciju logičkih kola i nekih tipova programa automatski obavljaju programi u kojima složenost izračunavanja ostaje dovoljno mala bez obzira na rast veličine problema. Ulazni podaci takvog programa su opisi objekta čija korektnost se proverava i odgovarajućeg zahteva.

Primer 9.8.2 Za program iz primera 9.8.1 zahtev bi jednostavno bila formula kojom se kaže da je izlaz uvek jednak dekadnoj vrednosti ulaznih bitova, dok bi automatski prevodilac programski kod transformisao u jednostavniju strukturu pogodnu za manipulaciju. Ξ

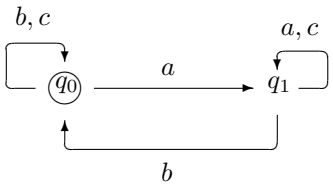
Pogodan oblik zapisa zahteva koje sistem treba da ispuni se vrši sredstvima linearne temporalne logike. Objekat je korektan ako sva njegova izračunavanja zadovoljavaju zahteve. Da bi se verifikacija izvršila, potrebno je dati i nekakav formalni opis objekta i to se često vrši pomoću mašina sa konačnim skupom stanja[15]. Treba reći da nisu svi programi, za razliku od logičkih kola, pogodni za opisivanje mašinama sa konačnim skupom stanja. Ovo se pogotovo odnosi na programe pisana na programskom jeziku C u kojima je naglasak na pokazivačima koji u principu mogu da pokazuju na bilo koji memorijski registar i time uvode nedefinisane programske promenljive. Primetimo da to nije slučaj sa programom iz primera 9.8.1. Formalni opis programa zavisi i od računara na kome se program izvršava.

Primer 9.8.3 Greška u radu programa iz primera 9.8.1 se javlja pri pojavi trideset i trećeg bita kod računara čiji registri su dužine 32 bita, ali ne i kod računara sa registrima dužine 64 bita. Ξ

Konačno, potrebno je na neki način povezati opis objekta i opis zahteva koji se proverava. Za to se, u pristupu koji ćemo predstaviti do kraja odeljka, koristi teorija apstraktnih mašina, odnosno automati nad beskonačnim rečima[16]. Primer osobine koje se verifikuje je osobina dogodivosti da će svaki zahtev za nekim resursom biti ispoštovan. Recimo, kod višekorisničkih operativnih sistema se javljaju nedeljivi resursi za čije korištenje je neophodna dozvola. Procesi koji traže resurs čekaju na takozvanom semaforu, pa je prirodno zahtevati da bude ispunjeno da ako proces dovoljno dugo čeka na crvenom svetlu, ono će eventualno, pre ili posle, postati zeleno, čime se procesu dozvoljava da koristi resurs. Druga vrsta osobine koja se može verifikovati iskazana je u primeru 9.8.1 i za nju se traži da uvek važi.

[15] Finite state machines.

[16] Automata on infinite words, property automata, ω-automata, Büchi-automata.



Slika 9.7. Buhijev automat.

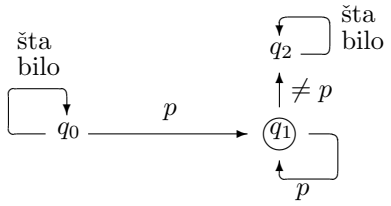
Automati nad beskonačnim rečima ili automati Buhija su zapravo konačni automati. Suštinsku novinu u odnosu na do sada izlaganu teoriju apstraktnih mašina predstavlja to što su reči koje se ispituju beskonačne zbog čega se može izraziti eventualnost, važenje zauvek ili stalno ponavljanje neke osobine. Pošto su reči beskonačne, njihovo prihvatanje se ne može definisati zaustavljanjem u nekom od završnih stanja. Kako je broj stanja automata konačan, a dužina reči nije, postoje stanja u kojima se čitajući ulaznu reč automat nalazi beskonačno mnogo puta. Kriterijum prihvatanja reči je da se među stanjima koja se beskonačno mnogo puta pojavljuju tokom čitanja reči nalazi bar jedno završno stanje. U nedeterminističkom slučaju je dovoljno da postoji bar jedno izvršavanje u kome se bar jedno završno stanje pojavljuje beskonačno mnogo puta. Buhijev automat ima slična svojstva koja su u teoremi 9.3.6 pripisana konačnim automatima. Do razlika, naravno, dolazi u dokazima i složenosti pojedinih algoritama. Ovde je od posebnog značaja da je, bez obzira na to što automat prihvata beskonačne reči, odlučiv problem da li je jezik prazan.

Primer 9.8.4 Na slici 9.7 je prikazan Buhijev automat sa dva stanja u kome je stanje q_0 i početno i završno. Automat prihvata reči u kojima se nakon svake pojave znaka a mora pojaviti i b. Ξ

O temporalnim logikama je bilo reči u odeljku 7.1.7. Ovde ćemo koristiti linearnu vremensku logiku, klasu beskonačnih, linearnih, diskretnih modela sa početnim momentnom i formule na jeziku $\{\neg, \bigcirc, \wedge, F, G, U\}$ za zapisivanje osobina objekata koji se testiraju.

Svakom momentu se pridružuje skup iskaznih slova koja su u njemu tačna. Kaže se da model zadovoljava formulu ako formula važi u početnom momentu. Pošto svaka formula α sadrži konačan podskup $Prop(\alpha)$ skupa svih iskaznih slova, u ispitivanju da li neki model zadovoljava formulu, ceo model se može shvatiti kao beskonačna reč na alfabetu $\mathbb{P}(Prop(\alpha))$. Dokazano je da se za proizvoljnu formulu α može konstruisati Buhijev automat A_α koji prihvata tačno one reči koje opisuju modele koji zadovoljavaju α.

Primer 9.8.5 Na slici 9.8 je prikazan nedeterministički Buhijev automat A_{FGp} za formulu FGp koja je zadovoljena ako i samo ako počev od nekog momenta stalno važi iskazno slovo p. Automat čitajući bilo šta ostaje u stanju q_0. Kada pročita p može ostati u stanju q_0, ali i preći u jedino završno stanje q_1. Kada se nađe u stanju q_1, automat zauvek ostaje u njemu ako stalno učitava p, inače, ako se pojavi



Slika 9.8. Buhijev automat za formulu FGp.

bilo šta što nije p, reč se ne prihvata. Prema tome, automat prihvata reč ako i samo ako se počev od nekog mesta pojavljuje samo p što odgovara modelu u kome počev od nekog stanja p važi zauvek. Ξ

Primer 9.8.6 U programu iz primera 9.8.1 zahtev za koji se testira korektnost bi mogao biti opisan automatom sa dva stanja. Stanje 0 je završno stanje. Prelazak iz njega u stanje 1 bi se ostvario ako se pojavi iskaz da je dekadna vrednost ulaznih bitova različita od vrednosti broja x. Ξ

Mašina sa konačnim skupom stanja je četvorka oblika $P = \langle W, w_0, R, v\rangle$, gde su W konačan skup stanja, $w_0 \in W$ početno stanje, R relacija dostižnosti između stanja i $v : W \to \mathbb{P}(\phi)$ interpretacija koja svakom stanju pridružuje skup iskaznih slova koja važe u njemu. Skup W opisuje stanja u kojima se objekat može naći tokom rada, a R način prelaska iz jednog u drugo stanje. Pretpostavlja se da, ako se neko izvršavanje objekta završava, onda on zauvek ostaje u stanju u kome se našao na kraju rada. Neka je beskonačni niz stanja $u = u_0, u_1, \ldots$ takav da važi $w_0 = u_0$ i $u_i R u_{i+1}$, za $i \geq 0$. Tada je niz $v(u_0)$, $v(u_1)$, $\ldots$ izračunavanje za P. Izračunavanje u zadovoljava formulu α ako $u_0 \models \alpha$, dok P zadovoljava formulu α ako sva izračunavanja za P zadovoljavaju α. P se može shvatiti i kao nedeterministički automat Buhija oblika $A_P = \langle W, w_0, W, \mathbb{P}(\phi), \delta\rangle$ gde je funkcija prelaza definisana tako da važi:

- $s' \in \delta(s, a)$ ako i samo ako $(s, s') \in R$, $a = v(s)$.

Primer 9.8.7 U jednoj implementaciji, programu iz primera 9.8.1 stanja bi predstavljale sve moguće (od 2^{32}, recimo) vrednosti promenljive x. Za svaku vrednost v funkcija prelaza vodi do stanja $2v$, $2v+1$, 0 i v zavisno da li je vrednost učitanog znaka *input* jednaka 0, 1, $\backslash n$ ili bilo kom drugom znaku. Ξ

Pošto je skup završnih stanja jednak skupu svih stanja, svakim beskonačnim nizom stanja prihvata se neka beskonačna reč alfabeta $\mathbb{P}(\phi)$, tj. neko izračunavanje od P. Zato je $L(A_P)$ skup svih izračunavanja za P, odnosno skup potencijalnih modela nekih formula.

Dakle, formalizam Buhijevih automata povezuje opis zahteva i opis objekta za koji se proverava da li zahtev važi. Preciznije, problem verifikacije se odnosi na ispitivanje da li sistem opisan mašinom sa konačnim skupom stanja $P = \langle W, w_0, R, v\rangle$ zadovoljava neku formulu α. Na osnovu iznetog, to se svodi na proveru da li

izračunavanja koja prihvata A_P zadovoljavaju formulu α, odnosno da li je $L(A_P) \subset L(A_\alpha)$. Ovo pitanje se može preformulisati tako da se proverava da li je

$$L(A_P) \cap L(\overline{A_\alpha}) = L(A_P) \cap L(A_{\neg\alpha}) = \emptyset$$

što je odlučiv problem.

9.9 Za dalje proučavanje

Početak teorije formalnih jezika vezuje se za radove Noama Čomskog [19]. Celovit pregled teorijskih rezultata iz ove oblasti može se naći u [51], dok pojedina poglavlja iz [23, 73] sažimaju tu materiju. Na našem jeziku postoji udžbenik [21] i zbirka zadataka [120] sa većim brojem urađenih primera. Dokaz odlučivosti problema utvrđivanja ekvivalencije determinističkih konačnih automata dat je relativno skoro u [117]. Autor je za taj rad osvojio Gödelovu nagradu. U oblasti prevodilaca programskih jezika osnovnu referencu predstavlja [5]. $LR(k)$-gramatike su uvedene u [66]. Verifikacija zasnovana na kombinaciji teorije automata i temporalne logike prikazana je u [41, 125]. Još neke primene formalnih jezika, recimo u prepoznavanju oblika, opisane su u [3].

10

Teorija složenosti izračunavanja

Kao što je prikazano u poglavlju 2 teorija algoritama omogućava da se napravi jasna granica između problema za čije rešavanje postoje algoritmi, odnosno programi za apstraktne mašine, i onih za koje to nije slučaj. Prvu grupu čine odlučivi, a drugu neodlučivi problemi. Razmatrajući Čerčovu tezu u odeljku 2.6 skrenuli smo pažnju da postoji suštinska razlika između praktično izračunljivih funkcija i funkcija koje se mogu izračunati u principu. U ovom poglavlju ćemo prikazati jednu klasifikaciju složenosti odlučivih problema merenu računarskim resursima poput vremena i memorijskog zauzeća koji se koriste tokom rešavanja problema. Tjuringova mašina se koristi kao osnovna apstraktna mašina čiji resursi se procenjuju tokom rada programa. Njene prednosti su relativna jednostavnost i mogućnosti efikasnih simulacija drugih apstraktnih mašina, tako da se osnovne klase složenosti (P, NP i $PSPACE$) ne menjaju promenom apstraktne mašine koja se koristi u definisanju odlučivih problema. Jedina oblast teorije složenosti u kojoj se ne koriste Tjuringove mašine se odnosi na paralelno izračunavanje o čemu u ovom poglavlju neće biti reči. Prema potrebi ćemo u nastavku, kao i u ovom pasusu, govoriti o složenosti problema, složenosti skupa, složenosti izvršavanja programa, odnosno o složenosti izračunavanja funkcija imajući u vidu ranije dokazanu ekvivalentnost formalnih sistema izračunavanja poput Tjuringovih mašina i parcijalno rekurzivnih funkcija. Problemi koji će biti analizirani uglavnom se mogu opisati kao ispitivanje da li neke reči pripadaju odgovarajućim formalnim jezicima.

Pod praktično izračunljivim problemima se podrazumevaju oni kod kojih je dužina rada odgovarajućih programa limitirana nekim stepenom dužine ulaznih podataka. Preostali odlučivi problemi se smatraju praktično neizračunljivim, tj. izračunljivim samo u principu. Za takve probleme se ne preporučuje konstrukcija opštih algoritama za rešavanje, već se pokušava pronalaženje efikasnih rešavača za neke posebne potprobleme. I pored upornog istraživanja, granica između praktično izračunljivih i praktično neizračunljivih problema nije precizno određena kao što je to slučaj sa odlučivim i neodlučivim predikatima. Tako je, recimo, problem zadovoljivosti iskaznih formula u izvesnom smislu reprezent klase praktično neizračunljivih problema. Za sada još nije pokazano, iako se u to duboko veruje, da ovaj

problem ne pripada klasi praktično izračunljivih problema. Ako bi se dokazalo da problem zadovoljivosti ipak pripada i ovoj klasi problema, onda bi granica koja razdvaja praktično izračunljive od praktično neizračunljivih problema morala biti znatno podignuta.

Razvoj Interneta i elektronskog poslovanja učinio je da je sigurnost komunikacija izbila u prvi plan interesovanja. Neke primene teorije složenosti nalazi upravo u toj oblasti, o čemu će biti reči u odeljku 10.8.

Na kraju poglavlja, u odeljku 10.9, spomenućemo i drugačije pristupe složenosti.

10.1 Opis problema

Kao što je već rečeno formalni model izračunavanja koji ćemo koristiti u analizi složenosti su Tjuringove mašine sa više traka, i to:

- nedeterministička Tjuringova mašina $\langle S, q_0, \{q_z, q_{da}, q_{ne}\}, A, \delta\rangle$ opisana:
  - konačnim brojem $k \geq 1$ traka koje su ograničene sa leve strane i od kojih prva traka sadrži ulazne podatke, a poslednja eventualni rezultat,
  - konačnim skupom S stanja, od kojih je q_0 početno, a $\{q_z, q_{da}, q_{ne}\}$ je skup završnih stanja koja redom označavaju završetak rada, pozitivan odgovor na pitanje i negativan odogovor na pitanje,
  - alfabetom A koji sadrži znake koji se upisuju u ćelije mašine među kojima su marker levog kraja trake $\triangleright$ i blanko znak i
  - programom u kome se svakoj kombinaciji tekućeg stanja i sadržaja ćelija nad kojima se nalaze glave mašine pridružuje jedna, ili više akcija; svaka akcija opisuje da li se ostaje u istom ili prelazi u novo stanje, upisuju novi sadržaji u ćelije nad kojima se nalaze glave i, konačno, svaka glava pomera levo ili desno ili ostaje na istom mestu,
- deterministička Tjuringova mašina koja se od nedeterminističke razlikuje smo po tome što se svakoj kombinaciji tekućeg stanja i sadržaja ćelija nad kojima se nalaze glave mašine pridružuje najviše jedna akcija i
- Tjuringova mašina sa ulazom i izlazom, koja može biti deterministička ili nedeterministička, a koja se od prethodnih mašina razlikuje samo po tome što ima bar dve trake i kod koje se prva traka može samo čitati, a u poslednju traku samo upisivati; ovo poslednje se obezbeđuje tako što se glava poslednje trake ne sme kretati ulevo.

Završno stanje q_z se koristi pri analizi složenosti izračunavanja funkcija, kada prelazak u to stanje označava završetak rada programa.

U odeljku 2.3.8 je pokazano da mašina sa jednom trakom može simulirati prethodno spomenute tipove Tjuringovih mašina. Zato se njihovim korištenjem ne povećava izražajnost u odnosu na osnovni model. U istom odeljku diskutovana je i efikasnost simulacije, a zaključci se sumiraju u naredne dve teoreme. Pojam vremenske granice složenosti se uvodi u odeljku 10.4.

Teorema 10.1.1 Za datu deterministič̆ku Tjuringovu mašinu M sa k-traka i vremenskom granicom složenosti $f(n)$ može se konstruisati deterministič̆ka Tjuringova mašina M' sa jednom trakom koja simulira rad mašine M i ima vremensku granicu složenosti $O(f(n)^2)$.

Za nedeterministič̆ke Tjuringove mašine nije poznat ovako efikasan način simuliranja deterministič̆kim mašinama. Postupak, opisan u odeljku 2.3.8, eksponencijalno povećava složenost rada pri deterministič̆koj simulaciji nedeterministič̆ke mašine. Međutim, koncept nedeterministič̆kih mašina se, iako nerealistič̆an u smislu realizacije na nekom od stvarnih računara, pokazao veoma korisnim u određivanju granica složenosti izračunavanja jer se pojavlju problemi koji se elegantno rešavaju nedeterministič̆kim, a veoma teško deterministič̆kim mašinama.

Teorema 10.1.2 Za datu nedeterministič̆ku Tjuringovu mašinu M sa k-traka i vremenskom granicom složenosti $f(n)$ može se konstruisati deterministič̆ka Tjuringova mašina M' sa jednom trakom koja simulira rad mašine M i ima vremensku granicu složenosti $O(c^{f(n)})$, za $c > 1$ koje zavisi od mašine M.

Pored nedeterministič̆kih i deterministič̆kih mašina u opisima klasa složenosti se koriste i druge varijante Tjuringovih mašina. Recimo, u odeljku 10.7 opisaćemo verovatnosne Tjuringove mašine.

Problemi koji se analiziraju u teoriji složenosti izračunavanja karakterišu se pitanjima na koja se obič̆no odgovara sa 'da' ili 'ne'. Recimo, jedan problem se odnosi na ispitivanje da li je graf povezan. Svaki konkretan graf za koji se postavi ovo pitanje je *primerak* problema. U nekim situacijama, kao kod optimizacije, rešenje problema je neki numerič̆ki rezultat. Ovaj slučaj se može svesti na prethodni tako što se pitanje preformuliše u oblik: 'ako je data konstanta c, da li je x rešenje problema za koje je vrednost funkcije koja se optimizuje jednaka sa (veća od, manja od) c?', ali se može rešavati i direktno konstrukcijom odgovarajuće funkcije. Predstavljanje problema se vrši u nekom formalnom jeziku na alfabetu neke Tjuringove mašine, što se formalizuje sledećom definicijom.

Definicija 10.1.3 *Problem* L za koji se ispituje složenost je podskup skupa svih reči nekog alfabeta[1]. *Komplement problema* L, u oznaci $\overline{L}$ na nekom alfabetu je skup svih reči na tom alfabetu koje nisu u L.

Neka je dat alfabet A i neka je L problem. Tjuringova mašina *prihvata* ulazni podatak, tj. reč, x ako postoji izračunavanje[2] u kome se, polazeći od reči x upisane na ulaznoj traci u početnom stanju q_0, dolazi do završnog stanje q_{da}, a *odbacuje* x ako uvek dolazi do završnog stanje q_{ne}. Ako Tjuringova mašina M prihvata sve reči x jezika koje pripadaju problemu L, a odbacuje svaku reč koja nije u problemu L, kaže se da M *odlučuje* problem L.

Ako Tjuringova mašina M za ulazni podatak x u izračunavanju dolazi do stanja q_z, onda je sadržaj poslednje trake rezultat rada mašine i označava se sa $M(x)$. Sa $L(x)$ ćemo označavati primerak problema L za ulazni podatak x, odnosno pitanje da li je $x \in L$.

[1] Imajući u vidu rečeno u poglavlju 9 problem je zapravo jezik na nekom alfabetu.

[2] U slučaju deterministič̆kih Tjuringovih mašina, to izračunavanje je jedinstveno.

Primer 10.1.4 Neka je L problem ispitivanja povezanosti dva čvora grafa i x opis nekog grafa i njegova dva izabrana čvora. Tada je $L(x)$ primerak problema L u kome se ispituje da li su u grafu opisanom sa x povezani izabrani čvorovi. Ξ

Ako je $\overline{L}$ komplement problema L, onda je za svaki primerak x problema odgovor na pitanje da li je $\overline{L}(x)$ pozitivan, odnosno negativan, ako i samo ako je odgovor na pitanje $L(x)$ negativan, odnosno pozitivan.

Primer 10.1.5 Komplement problema ispitivanja zadovoljivosti formule je ispitivanje nezadovoljivosti formule. Ξ

Da bi opis problema koji koristimo bio univerzalan potrebno je proizvoljan zadatak predstaviti kao niz reči u nekom alfabetu. Recimo, graf bez izolovanih čvorova se može prikazati kao niz ivica, odnosno niz uređenih parova čvorova, elementi konačnog skupa se prikazuju kao prirodni brojevi koji se opet prikazuju u binarnom obliku itd.

10.2 Apstraktna složenost izračunavanja

U ovom odeljku ćemo iskazati nekoliko tvrđenje koja ne zavise od izbora mere kojom se procenjuje složenost izračunavanja. Zato se ovaj deo teorije složenosti izračunavanja naziva apstraktna složenost.

Definicija 10.2.1 Neka je f_0, f_1, f_2, ... bilo koje nabrajanje unarnih parcijalno rekurzivnih funkcija. Binarna parcijalno izračunljiva funkcija C je *mera složenosti* ako zadovoljava *Blumove aksiome*:

- $C(x,i)\downarrow$ ako i samo ako $f_i(x)\downarrow$ i
- za date x, i i y odlučivo je da li je $C(x,i)\leq y$

Kao i ranije možemo poistovetiti programe sa funkcijama koje oni izračunavaju. Zato se funkcija $C(x,i)$ može shvatiti kao složenost izvršavanja programa za neku apstraktnu mašinu koji ima kod i i ulazni podatak x.

Primer 10.2.2 Primeri mera koje zadovoljavaju prethodnu definiciju su broj koraka u izvršavanju programa, memorijsko zauzeće, najveća vrednost koju imaju sve promenljive u programu, maksimalan broj promena pravca kretanja glava nad trakama Tjuringove mašine itd., dok to nije slučaj sa funkcijom $C(x,i)=f_i(x)$. Za ovo je dovoljno razmotriti funkciju

$$f_i(x)=\begin{cases} 0 & \text{ako } x\in S \\ \text{nedefinisano} & \text{inače}\end{cases}$$

gde je S bilo koji parcijalno odlučiv predikat. Predikat $C(x,i)\leq 0$ je ekvivalentan sa $x\in S$ i nije rekurzivan. Ξ

Definicija 10.2.3 Rekurzivna funkcija $f(x)$ je *faktor skaliranja* ako je ispunjeno:

- f je neopadajuća funkcija, tj. $f(n)\leq f(n+1)$ za svaki prirodan broj n i

- f nije ograničena odozgo, odnosno za svaki prirodan broj m postoji prirodan broj n tako da je $f(n) > m$.

Sledeća teorema opisuje vezu između proizvoljnih mera složenosti.

Teorema 10.2.4 Neka je $C(x, i)$ proizvoljna mera složenosti i $f(x)$ faktor skaliranja. Tada je $D(x, i) = f(C(x, i))$ takođe mera složenosti.

Neka su C i D dve proizvoljne mere složenosti. Tada postoji binarna rekurzivna funkcija r strogo rastuća po drugoj koordinati[3] takva da za svaki prirodan broj i važi:

- počev od nekog n_1 za svaki $x > n_1$ je $C(i, x) \leq r(x, D(x, i))$ i
- počev od nekog n_2 za svaki $x > n_2$ je $D(i, x) \leq r(x, C(x, i))$.

U teoremi 10.2.5 je formulisano jedno iznenađujuće tvrđenje. Pretpostavimo da je C neka fiksirana mera složenosti i da je t neka unarna rekurzivna funkcija kojom ograničamo složenost izračunavanja. Ako razmatramo samo one funkcije za koje je ispunjeno $C(x, i) \leq t(x)$ kad god $f_i(x) \downarrow$, razumno je pretpostaviti da ćemo povećavajući ograničenje t, recimo na $2^{t(x)}$ ili čak i više, biti u stanju da sprovedemo mnogo više izračunavanja. Međutim, pokazuje se da za neke funkcije t to nije slučaj, naime da za svaku funkciju f postoji samo konačno mnogo prirodnih brojeva takvih da je složenost izračunavanja $f(x)$ veća od $t(x)$ a manja do jednaka od nove granice[4]. Ovakva situacija govori da postoji izvesna praznina u mogućnosti izračunavanja funkcija $f(x)$, pa se naredna teorema naziva i teorema o praznini[5].

Teorema 10.2.5 Neka su $C(x, i)$ mera složenosti i $g(x, y)$ rekurzivna funkcija za koju je $g(x, y) > y$ za svaki $y \in \mathbb{N}$. Tada postoji rekurzivna funkcija $t(x)$ takva da kad god je $x > i$ i $C(x, i) \leq g(x, t(x))$, onda je $C(x, i) \leq t(x)$.

Drugim rečima, u teoremi se kaže da za dovoljno veliko x, ako se $f(x)$ ne može izračunati resursima ograničenim funkcijom t, tada se $f(x)$ ne može izračunati ni korištenjem neuporedivo snažnijih resursa. Teoremu možemo interpretirati i na sledeći način. Zamislimo da posedujemo dva personalna računara, od kojih je jedan zastareli IBM XT, a drugi zasnovan na nekom od najnovijih Intelovih procesora. Neka su $C(x, i)$ i $D(x, i)$ vremena izvršavanja programa i na prvom i drugom računaru. Prema drugom delu teoreme 162 postoji rekurzivna funkcija r koja ove dve mere složenosti dovodi u vezu. Funkcija g definisana sa $g(x, y) = r(x, y) + y + 1$ je strogo rastuća po drugoj koordinati i ispunjava uslov teoreme 10.2.5 da je $g(x, y) > y$. Zbog toga postoji $n \in \mathbb{N}$ tako da je za svaki $i \in \mathbb{N}$ i $x > n$ ispunjeno $C(x, i) \leq r(x, D(x, i)) \leq r(x, D(x, i)) + D(x, i) + 1 = g(x, D(x, i))$. Neka funkcija $t(x)$ ispunjava uslov teoreme 10.2.5 za meru složenosti C i funkciju $g(x, y)$. Pretpostavimo da se neki program čiji je broj i za dovoljno veliki ulaz x izvršava u vremenu $D(x, i) \leq t(x)$ na bržem računaru. Tada će vreme izvršavanja $C(x, i)$ na sporijem računaru biti ograničeno sa $g(x, D(x, i))$, pa i sa $g(x, t(x))$.

[3] Za svaki $y \in \mathbb{N}$ je $r(x, y) < r(x, y + 1)$.

[4] Koristeći oznake koje se uvode u odeljku 10.4, ovo zapravo znači da $TIME(t(x)) = TIME(2^{t(x)})$.

[5] Gap theorem.

Sada primenom teoreme 10.2.5 dobijamo i da je $C(x,i) \leq t(x)$, odnosno da svaki program koji se za dovoljno veliko x izvršava u vremenu kraćem od $t(x)$ na bržem računaru, to isto radi i na sporijem računaru. Prema tome, kupovina novog, brzog i skupog računara ne pomaže uvek. Ovde treba obratiti pažnju da granica $t(x)$ nije proizvoljna, već jednog posebnog oblika koji se formuliše u dokazu teoreme 10.2.5.

Teorema 10.2.6 sa druge strane opravdava teorijsko razmatranje problema i stalno usavršavanje algoritama za njihovo rešavanje. U teoremi se kaže da u opštem slučaju ne postoji najbolji algoritam, odnosno njegova programska realizacija, tj. da postoje problemi za koje se za svaki algoritam koji ih rešava može konstruisati još efikasniji algoritam. Otuda nije čudo što se teorema 10.2.6 naziva i teorema o ubrzanju[6].

Teorema 10.2.6 Neka su $g(x,y)$ proizvoljna rekurzivna funkcija i C proizvoljna mera složenosti. Tada postoji rekurzivna funkcija $f(x)$ takva da je $f(x) \leq x$ i za svaki program sa indeksom i koji izračunava funkciju f postoji program sa indeksom j takav da je za sve x veće od nekog $n \in \mathbb{N}$ ispunjeno $g(x, C(x,j)) \leq C(x,i)$.

Uslovi teoreme 10.2.6 znače da se i za neku funkciju čije su vrednosti relativno male, ograničene veličinom ulaznog podatka, proizvoljan resurs može proizvoljno mnogo optimizovati.

10.3 O-notacija

U teoriji složenosti izračunavanja često se razmatra brzina rasta funkcija. Brzina rasta se analizira asimptotski, pri čemu se često koriste različite aproksimacije koje opisujemo narednim definicijama i tvrđenjima.

Definicija 10.3.1 Neka su f i g aritmetičke funkcije. Tada je *funkcija f u velikom O od g* (u oznaci $f(x) = O(g(x))$) ako postoje brojevi c i n takvi da za svaki $x > n$ važi $f(x) \leq c \cdot g(x)$. Ako takvi brojevi ne postoje, onda je $f(x) \neq O(g(x))$.

Umesto funkcija f je u velikom O od g kaže se *funkcija f je reda funkcije g* ili *funkcija g je asimptotska gornja granica funkcije f*.

Definicija 10.3.2 Funkcija f *raste brže* od funkcije g ako $f(x) \neq O(g(x))$. Funkcije f i g *rastu istom brzinom*, u oznaci $f(x) = \Theta(g(x))$, ako važi $f(x) = O(g(x))$ i $g(x) = O(f(x))$.

Primer 10.3.3 Jednostavnom analizom se zaključuje da funkcije n^4 i $1345 \cdot n^4 + 2007 \cdot n^3 - 7n + 5$ rastu istom brzinom, dok funkcija $0.0001 \cdot n^5$ raste brže od funkcije $1345 \cdot n^4 + 2007 \cdot n^3 - 7n + 5$. Ξ

Sledeće teoreme formulišu neke kriterijume za upoređivanje brzina rasta funkcija, a mogu se dokazati sredstvima koja se standardno koriste u realnoj analizi.

Teorema 10.3.4 Neka su f i g aritmetičke funkcije i neka je $\lim_{n\to\infty} \dfrac{f(n)}{g(n)} = \beta$. Ako je β pozitivan realan broj, onda funkcije f i g rastu istom brzinom. Ako je $\beta = \infty$, onda važi $g(x) = O(f(x))$ i $f(x) \neq O(g(x))$, tj. funkcija f raste brže od g.

[6] Speedup theorem.

Teorema 10.3.5 Neka je $P(n) = a_0 + a_1 \cdot n + \ldots + a_r \cdot n^r$, $a_r \neq 0$, polinom stepena r sa celobrojim koeficijentima. Tada za $P(n)$ i n^m važi:

1. ako je $m = r$, $P(n)$ i n^m rastu istom brzinom,
2. ako je $m < r$, $P(n)$ raste brže od n^m i
3. ako je $m > r$, n^m raste brže od $P(n)$.

Teorema 10.3.6 Neka je $k > 1$. Funkcije k^n raste brže od bilo kog polinoma sa celobrojnim koeficijentima. Svaki polinom sa celobrojnim koeficijentima raste brže od bilo koje logaritamske funkcije.

Kako je $\log_c x = \log_c d \cdot \log_d x$, direktno sledi sledeće tvrđenje.

Teorema 10.3.7 Za svake dve realne konstante $c, d > 1$ važi $\log_c(x) = \Theta(\log_d(x))$.

Funkcije koje se obično javljaju u O-notaciji prilikom analize složenosti su: logaritamska funkcija $\log_2 n$[7], linearna funkcija $k \cdot n$, njihov proizvod $n \log_2 n$, stepena funkcija n^k, eksponencijalna funkcija k^n itd. Svima njima je zajedničko:

- $\lim_{n\to\infty} f(n) = \infty$, što ima razumljivo intuitivno opravdanje: što je problem većih dimenzija složenost izračunavanja je veća,
- funkcije su neopadajuće i
- postoje Tjuringove mašine koje ih izračunavaju u prostoru i vremenu koji su proporcionalni vrednostima funkcija.

Ovakve funkcije se nazivaju *prave funkcije složenosti*[8] i upotrebljavaju se u analizi složenosti izračunavanja.

10.4 Klase složenosti

Najčešće korištene mere složenosti se odnose na vreme, tj. broj koraka izvršavanja programa, i prostor, tj. količinu memorije koju koristi program. Uobičajeno je da se složenost izražava kao funkcija veličine ulaznog podatka. Ako je x ulazni podatak programa, njegova veličina se označava sa $|x|$. U slučaju da je ulazni podatak opis grafa, pogodno je da $|x|$ bude broj čvorova grafa. Slično, ako je ulazni podatak reč, $|x|$ označava dužinu, tj. broj znakova reči.

Definicija 10.4.1 *Vreme izvršavanja* izračunavanja Tjuringove mašine M koja kao ulaz dobija podatak x jednako je dužini niza konfiguracija koje predstavljaju to izračunavanje.

Neka je f unarna aritmetička funkcija, koja zadovoljava uslove za pravu funkciju složenosti. Tjuringova mašina M radi *u vremenu* $f(n)$, ako je za bilo koji ulazni

[7] Prema teoremi 10.3.7 konstanta koja je baza logaritma nije bitna, pa je moguće ravnopravno koristiti i druge logaritamske funkcije bez promene klase koja se njima određuje. Ponekad se u literaturi ovo ogleda u tome što se piše samo $\log n$.

[8] Proper complexity functions.

podatak x vreme izvršavanja izračunavanja mašine najviše $f(|x|)$. Za nedeterminističku Tjuringovu mašinu M se kaže da radi u vremenu $f(n)$, ako je za bilo koji ulazni podatak x vreme izvršavanja bilo kog izračunavanja mašine najviše $f(|x|)$. Funkcija f je *vremenska granica složenosti* za M.

$TIME(f(n))$ je skup problema za koje postoje determinističke Tjuringove mašine koje ih odlučuju, a za koje je vremenska granica složenosti $f(n)$. $NTIME(f(n))$ se definiše analogno, u odnosu na nedeterminističke Tjuringove mašine.

Prostorna složenost nekog problema se definiše na donekle izmenjen način u odnosu na vremensku složenost. Razlog za to je želja da se izbegne uključivanje prostora u koji je upisan ulazni podatak, odnosno u koji se smešta rezultat, u razmatranje prostorne složenosti. Za to su pogodne Tjuringove mašine sa ulazom i izlazom. Restrikcija o kojoj je reč ne smanjuje izražajne sposobnosti, pošto je trivijalno da za svaku Tjuringovu mašinu sa k traka koja radi u vemenu $f(n)$ postoji Tjuringova mašina sa ulazom i izlazom sa $k+2$ trake koja rešava isti problem u vremenu $O(f(n))$.

Definicija 10.4.2 *Prostor izvršavanja* izračunavanja Tjuringove mašine M sa ulazom i izlazom koja kao ulaz dobija podatak x jednak je broju različitih ćelija traka, sem prve - ulazne i poslednje - izlazne trake, nad kojima se tokom izračunavanje nađu glave traka.

Neka je f unarna aritmetička funkcija koja zadovoljava uslove za pravu funkciju složenosti. Tjuringova mašina M radi *u prostoru* $f(n)$, ako je za bilo koji ulazni podatak x prostor izvršavanja izračunavanja mašine najviše $f(|x|)$. Za nedeterminističku Tjuringovu mašinu M se kaže da radi u prostoru $f(n)$, ako je za bilo koji ulazni podatak x prostor izvršavanja bilo kog izračunavanja mašine najviše $f(|x|)$. Funkcija f je *prostorna granica složenosti* za M.

$SPACE(f(n))$ je skup problema za koje postoje determinističke Tjuringove mašine koje ih odlučuju, a za koje je prostorna granica složenosti $f(n)$. Skup problema $NSPACE(f(n))$ se definiše analogno, u odnosu na nedeterminističke Tjuringove mašine.

Ovakav pristup prostornom zauzeću omogućava razmatranje mašina koje koriste manje od $|x|$, recimo $\log_2 |x|$, ćelija, gde se pod korištenjem podrazumeva da su te ćelije radni prostor, odnosno da se u njih privremeno smeštaju podaci koji se upotrebljavaju tokom izračunavanja. Pri tome se u prostor čija se veličina meri ne uključuju, recimo, ćelije prve trake koje sadrže ulazni podatak.

Primer 10.4.3 Neka je M mašina sa ulazom i izlazom koja sadrži četiri trake i ispituje da li je ulazna reč palindrom. Prva traka sadrži ulaznu reč i moguće ju je samo čitati. Druga traka sadrži binarni zapis indeksa i koji označava redni broj ciklusa rada, treća binarni zapis indeksa j, dok se četvrta traka ne koristi. Na početku rada indeks i se postavlja na 1, a zatim se rad obavlja u ciklusima. Svaki ciklus počinje inicijalizovanjem indeksa j na 1 i postavljanjem glave prve trake nad najlevlju ćeliju ulazne reči. Ako je $j < i$, uvećava se j za 1 i pomera glava prve trake nadesno. Ako je $j = i$ pamti se simbol prve trake koji se trenutno čita i ponovo postavlja j na 1. Zatim se analogno pronalazi i-ti znak ulazne reči brojano sa desne strane i upoređuje sa zapamćenim simbolom. Postupak se prekida kada

su upoređeni znaci različiti, kom prilikom se prelazi u stanje q_{ne}, odnosno kada je i-ti znak ulazne reči blanko znak, pri čemu se prelazi u stanje q_{da}. U prvom slučaju reč nije, a u drugom reč jeste palindrom. Prostor izvršavanja je u $O(\log_2 n)$ koliko je potrebno za binarno predstavljanje indeksa i i j. Ξ

Definicija 10.4.4 *Klasa složenosti* je skup problema sa zajedničkom vremenskom ili prostornom granicom.

Primer 10.4.5 Skupovi problema $TIME(f(n))$, $NTIME(f(n))$, $SPACE(f(n))$ i $NSPACE(f(n))$ su neke klase složenosti. Ξ

U definisanju klase složenosti se pretpostavlja da za granice složenosti $f(n)$ važi:

- $f(n) \geq n$, ako je reč o vremenskoj složenosti i
- $f(n) \geq \log_2 n$, ako je reč o prostornoj složenosti.

Intuitivno rečeno, nedeterminističke klase složenosti sadrže probleme kod kojih je broj kandidata za rešenje veliki, ali kada se kandidat izabere, onda je problem njegovog testiranja, (verifikacije, provere) u okviru odgovarajuće determinističke klase problema. Pri tome za svaki x koji je primerak problema postoji izračunavanje koje dovodi do prihvatanja, a problem predstavlja izbor izračunavanja kojim se x prihvata. Ni za jedan x koji nije primerak problema ne postoji takvo izračunavanje.

Primer 10.4.6 Primer problema koji se nalazi u nedeterminističkoj klasi je testiranje zadovoljivosti iskaznih formula, o čemu će više biti rečeno u odeljku 10.6.4. Za proizvoljnu formulu postoji relativno veliki broj interpretacija koje treba ispitati, ali ako se izabere pogodna interpretacija pri kojoj je formula zadovoljena, sama provera nije komplikovana. Slično, i problem trgovačkog putnika u kome se ispituje da li postoji put u grafu koji kroz svaki čvor prolazi tačno jednom i koji je kraći od neke unapred zadate konstante se nedeterministički lako rešava. Nedeterministička Tjuringova mašina treba da izabere jednu permutaciju čvorova grafa i proveri dužinu odgovarajućeg puta. Iako je broj permutacija n čvorova jednak $n!$, nedeterministički postupak ima polinomijalnu vremensku granicu složenosti. Ξ

Definicija 10.4.7 Neka je $\mathcal{C}$ neka klasa složenosti, njen komplement, u oznaci *co-*$\mathcal{C}$ je skup problema oblika $\{\overline{L} : L \in \mathcal{C}\}$.

Očigledno je da za sve determinističke klase složenosti važi $\mathcal{C} = co\text{-}\mathcal{C}$ jer se komplement svakog problema iz klase $\mathcal{C}$ rešava istom Tjuringovom mašinom koja dodatno menja završno stanje q_{da} u q_{ne} i obrnuto. Zato se kaže da su determinističke klase složenosti zatvorene za komplement. Nije poznato da li u opštem slučaju isto važi i za nedeterminističke klase složenosti.

10.4.1 Odnosi između klasa složenosti

Određivanje odnosa između klasa složenosti je jedno od osnovnih pitanja kojima se bavi teorija složenosti izračunavanja. Tvrđenje 10.4.8 je u duhu teoreme 10.2.6 i govori da se iz granice složenosti $f(n)$ može eliminisati konstantni faktor kojim se množi najsloženiji deo funkcije, tj. da je red brzine rasta $O(f(n))$ ono što je u granici složenosti $f(n)$ bitno.

Teorema 10.4.8 Neka je problem $L \in TIME(f(n))$. Tada je za proizvoljno $\epsilon > 0$, $L \in TIME(\epsilon f(n) + n + 2)$.

Neka je problem $L \in SPACE(f(n))$. Tada je za proizvoljno $\epsilon > 0$, $L \in SPACE(\epsilon f(n) + 2)$.

Sa druge strane, teorema 10.4.9, takozvana *teorema hijerarhije* govori da sa dovoljnim povećanjem granice složenosti klase složenosti šire. Ovo je, naizgled, u suprotnosti sa teoremom 10.2.5, ali smo već naglasili da se u toj teoremi govori o posebnim funkcijama koje nisu prave funkcije složenosti.

Teorema 10.4.9 Neka je $f(n)$ prava funkcija složenosti. Tada važi:

1. ako je $f(n) \geq n$, onda je[9] $TIME(f(n)) \subsetneqq TIME((f(2n+1))^3)$ i
2. $SPACE(f(n)) \subsetneqq SPACE(f(n) \cdot \log_2 f(n))$.

U opisu granica složenosti O-notacija se koristi na sledeći način: $TIME(O(f(n)) = \cup_{c>0} TIME(c \cdot f(n))$ ili $SPACE(2^{O(n)}) = \cup_{c>0} SPACE(2^{c \cdot n})$. Često se umesto neke posebne funkcije koja definiše granicu složenosti koristi familija funkcija, recimo familija svih stepenih funkcija, svih eksponencijalnih funkcija itd., što podrazumeva da je takva klasa složenosti unija klasa određenih elementima familije. Neke od važnijih klasa složenosti su:

- $L = SPACE(O(\log_2 n))$
- $NL = NSPACE(O(\log_2 n))$
- $P = \cup_i TIME(n^i)$
- $NP = \cup_i NTIME(n^i)$
- $PSPACE = \cup_i SPACE(n^i)$
- $NPSPACE = \cup_i NSPACE(n^i)$
- $EXP = \cup_i TIME(2^{n^i})$,
- $NEXP = \cup_i NTIME(2^{n^i})$,
- $EXPSPACE = \cup_i SPACE(2^{n^i})$,
- $2 - EXP = \cup_i TIME(2^{2^{n^i}})$,
- $2 - NEXP = \cup_i NTIME(2^{2^{n^i}})$ itd.

[9] Može se pokazati i da je gustina različitih klasa veća. Recimo, svaka klasa $TIME(f(n))$ je pravi podskup kalse $TIME(f(n) \log_2^2 f(n))$. Nama je ovde dovoljno i slabije tvrđenje.

Dakle, P je klasa složenosti koja sadrži one probleme za koje je vremenska granica složenosti programa koje ih rešavaju neka stepena[10] funkcija. Primetimo da su, zbog određenih tehničkih pogodnosti, u klasama složenosti EXP i $NEXP$ stepeni funkcije polinomi. Nazivima klasa koje odgovaraju deterministčkim Tjuringovim mašinama ponekada se dodaje slovo D, tako da se umesto $TIME$ koristi $DTIME$, a umesto $SPACE$, $DSPACE$.

U hijerarhiji klasa složenosti ima otvorenih pitanja o tome da li je neki stepen hijerarhije jednak nekom drugom stepenu. Često se zna da je jedan stepen sadržan u drugom, ali se ne zna da li, ili ne, važi i ubrnuto, tj. da li se stepeni poklapaju, ili je jedan pravi podskup od drugog. Neke od dokazanih relacija su iskazane u sledećem tvrđenju:

Teorema 10.4.10 Za klase složenosti (gde f označava pravu funkciju složenosti) važi:

1. $TIME(f(n)) \subset NTIME(f(n))$,
2. $NTIME(f(n)) \subset TIME(2^{O(f(n))})$,
3. $SPACE(f(n)) \subset NSPACE(f(n))$,
4. $NTIME(f(n)) \subset SPACE(f(n))$,
5. $NSPACE(f(n)) \subset TIME(2^{O(f(n))})$,
6. $NSPACE(f(n)) \subset SPACE(O(f(n) \cdot f(n)))$, za $f(n) \geq \log_2 n$,
7. $PSPACE = NPSPACE$,
8. $TIME(O(n)) \subsetneq NTIME(O(n))$,
9. $P \subsetneq EXP$,
10. $NL \subsetneq PSPACE$,
11. $NL \subset P$,
12. $PSPACE \subsetneq EXPSPACE$,
13. $co\text{-}NSPACE(f(n)) = NSPACE(f(n))$, za $f(n) \geq \log_2 n$ i
14. $co\text{-}NL = NL$, $co\text{-}NPSPACE = NPSPACE$.

Dokaz. Daćemo skicu dokaza za neke od navedenih (ne)jednakosti.
(1) i (3) trivijalno slede iz činjenice da je svaka deterministička Tjuringova mašina istovremeno i nedeterministička.
(2) je u vezi sa intuitivnim objašnjenjem nedeterministiške klase problema - iako je testiranja izabranog slučaja relativno jednostavno, broj potencijalnih rešenja problema je veliki, tako da se odgovarajuće pretraživanje obavlja u eksponencijalnom vremenu.

[10] Pošto je $O(n^k) = O(a_n \cdot n^k + \ldots + a_1 \cdot n + a_0)$, preciznije je reći neka polinomijalna funkcija.

(4) Ako je broj koraka u kojima se bira potencijalno rešenje i zatim vrši njegovo testiranje ograničen sa $f(n)$, onda se tokom tih koraka ne može posetiti više od $f(n)$ ćelija Tjuringove mašine.
(5) proizilazi iz činjenice da postoji samo konačno mnogo konfiguracija bilo koje Tjuringove mašine M u klasi složenosti $NSPACE(f(n))$. Konfiguraciju mašine sa k traka opišimo kao $2k+1$-torku oblika $\langle q, w_1, u_1, \ldots, w_k, u_k\rangle$ u kojoj je prva komponenta stanje mašine, a preostale u parovima (w_i, u_i) predstavljaju sadržaj i-te trake levo i desno od trenutne pozicije glave za tu traku. Dužina tih parova je najviše $f(n)$. Ako sa $|A|$ označimo broj znakova u alfabetu mašine, a sa $|Q|$ broj stanja, onda je broj konfiguracija ograničen sa $|Q|\cdot|A|^{2kf(n)}$, pa i sa $c_1^{f(n)}$, za neku konstantu c_1 koja zavisi samo od mašine M. Konfiguracije nazovimo susednim ako postoji naredba koja jednu prevodi u drugu. M prihvata reč x ako i samo ako postoji put od početne do neke konfiguracije sa stanjem q_{da}. Za ispitivanje dostižnosti u grafu može se koristiti algoritam u kome se krećući od polaznog čvora u svakom ciklusu označavaju neposredno dostižni čvorovi. Ovaj algoritam je u klasi $TIME(n^2)$, pa je ispitivanje da li M prihvata ulazni podatak x u $TIME(2^{O(f(n))})$. Ovakav metod simulacije rada klase sa prostornom granicom složenosti se naziva *metod dostižnosti*[11], a graf koji se sastoji od konfiguracija koje su povezane ako su susedne u spomenutom smislu se naziva *konfiguracioni graf* mašine M.
(6) Na osnovu koraka (5), (1) i (4) neposredno se dobija da je $NSPACE(f(n)) \subset SPACE(2^{O(f(n))})$, ali se može dobiti i preciznija granica. Slučaj (6) se dokazuje primenom teoreme 10.6.3 na konfiguracioni graf mašine M iz klase $NSPACE(f(n))$. Taj graf ima $2^{O(f(n))}$ čvorova, što predstavlja sve moguće rasporede $|A|$ znaka alfabeta A mašine na $f(n)$ registara koje mašina koristi. Za ispitivanje konfiguracionog grafa, prema teoremi 10.6.3, je potrebna rekurzija najveće dubine $\log_2(2^{O(f(n))})$. Svaki rekurzivni poziv se realizuje postavljanjem na stek trojke oblika n_1, n_2, n, gde su n_1 i n_2 redni brojevi čvorova koji sadrže konfiguracije, a n dužina puta koji ih spaja. Sva tri broja imaju najveću vrednost $2^{O(f(n))}$, pa njihova binarna reprezentacija zauzima najviše $\log_2(2^{O(f(n))})$. Odatle direktno sledi da se ispitivanje može izvesti u klasi složenosti $SPACE(O(\log_2^2(2^{O(f(n))})) = SPACE(O(f(n)\cdot f(n)))$.
(7) je direktna posledica slučaja (6) kada je $f(n)$ polinomijalna granica.
(9) je posledica teoreme 10.4.9.168. Svaki polinom je u $O(2^n)$, pa je $P \subset TIME(2^n) \subset EXP$. Kako je $TIME(2^n)$ pravi podskup od $TIME((2^{2n+1})^3) \subset TIME(2^{n^2}) \subset EXP$, direktno sledi da je P pravi podskup od EXP.
(10) se pokazuje u dva koraka. Najpre je prema slučaju (6), $NL \subset SPACE(O(\log_2^2 n))$, a zatim je primenom teoreme o hijerarhiji 10.4.9.2, $SPACE(O(\log_2^2 n)) \subsetneq SPACE(O(n)) \subset PSPACE$.
(11) je posledica slučaja (5).
(12) je posledica teoreme 10.4.9.2.
(14) je posledica slučaja (13). Ξ

Međutim, čitav niz problema je ostao nerešen, bez obzira na intenzivna istraživanja koja se sprovode. Neka od tih pitanja su:

- Da li je $P = NP$?

[11] Reachability method.

- Da li je $P = PSPACE$?
- Da li je $L = NL$?
- Da li je $EXP = NEXP$?

Prvo pitanje je od posebnog značaja s obzirom na ranije spomenutu granicu između praktično izračunljivh problema i onih koji su to samo u principu. Dokaz da je $P \neq NP$ bio bi potvrda takvih shvatanja, dok bi suprotan rezultat, mada malo verovatan, doveo do prave revolucije u razvoju algoritama. Odnos klasa $TIME(O(n))$ i $NTIME(O(n))$ opisan sa (8) je jedan od rezultata koju sugerišu da je $P \neq NP$. Zanimljivo je da iz $P = NP$ sledi $EXP = NEXP$.

Pregled dela hijerarhije klasa složenosti u kome se nalaze najvažnije klase dat je sa:

$$L \subset NL \subset P \subset NP \subset PSPACE \subset EXP \subset NEXP.$$

Pored već navedenih tvrđenja o tome u kojim od ovih slučajeva se naslednik u hijerarhiji zaista razlikuje od prethodnika, pod sumnjom su svi preostali. Primetimo da, pošto važi $NL \neq PSPACE$ i $P \neq EXP$, na bar nekim mestima u hijerarhiji relacija podskup mora biti striktna.

10.4.2 Pozicioniranje složenosti problema

Među pitanja kojima se bavi teorija složenosti izračunavanja spada i određivanje kojoj klasi složenosti pripada neki problem. Pri tome se obično određuju neke gornje i donje granice složenosti tako da budu što bliže jedna drugoj, ali se dešava da su one međusobno dosta udaljene, te da se složenost problema ne može uvek precizno odrediti. *Gornja granica*[12] složenosti nekog odlučivog problema se određuje konstrukcijom algoritma za njegovo rešavanje i analizom koliko vremena i/ili memorije taj algoritam koristi. Ovde treba primetiti da različiti algoritmi za rešavanje nekog problema mogu dati i različite gornje granice složenosti.

Primer 10.4.11 Razmotrimo Euklidov algoritam za nalaženje najvećeg zajedničkog delioca dva prirodna broja:

```
function Euklid(m,l)
begin
        while m > 0 do
                t := l mod m
                l := m
                m := t
        return l
end
```

Ako je $l \geq m$, uvek je $l \bmod m < \frac{l}{2}$. Neka k predstavlja ukupan broj prolazaka funkcije kroz petlju za ulazne podatke m i l, i neka su za $i \leq k$, m_i i l_i vrednosti od m i l na kraju i-te petlje. Uslov za izlazak iz petlje u koraku k je da je $m_k = 0$

[12] Upper bound.

i $m_i \geq 1$, za $i < k$. Vrednosti m_i i l_i su definisane na sledeći način: $l_i = m_{i-1}$ i $m_i = l_{i-1} \mathrm{mod}\ m_{i-1}$, za $1 \leq i \leq k$. Očigledno je da je za svaki $i \geq 1$, $l_i > m_i$. Zbog toga je

$$m_i = l_{i-1} \mathrm{mod}\ m_{i-1} < \frac{l_{i-1}}{2} = \frac{m_{i-2}}{2}$$

za svaki $i \geq 2$. Ako je $k = 2d + 1$ imamo

$$m_{k-1} < \frac{m_{k-3}}{2} < \frac{m_{k-5}}{4} < \ldots < \frac{m_0}{2^d}.$$

Pošto je $m_{k-1} \geq 1$, važi $m_0 \geq 2^d$. Odatle sledi $k = 2d + 1 \leq 1 + 2 \log_2 m_0$. Slično se analizira i slučaj za $k = 2d$, imajući u vidu da je $m_1 = l_0 \mathrm{mod}\ m_0 < m_0$.

Dakle, broj prolazaka kroz petlju je reda $\log_2 m$. Kako je dužina binarnog zapisa broja m upravo reda $\log_2 m$, broj prolazaka kroz petlju je u $O(n)$, gde je $n = |m|$ veličina binarne reprezentacije ulaznog podatka. Potrebno vreme za operaciju deljenja koje se vrši u petlji prilikom izračunavanja modula je u $O(\log_2^2 m)$, pa je složenost celog postupka u $O(n^3)$, za $n = |m|$. Ξ

Donja granica[13] složenosti nekog odlučivog problema se određuje tako što se pokaže da su izvesno vreme i/ili memorijski prostor neophodni za rešavanje tog problema bilo kojim algoritmom. Određivanje donje granice složenosti je često teško i nije poznat neki univerzalni postupak za to. Jedna od metoda koja se primenjuje je *metoda brojanja* u kojoj se definiše neka karakteristika ponašanja mašine, pa se analizira koliko puta se ta karakteristika mora ispuniti prilikom prihvatanja ulaza veličine n. Napomenimo i to da su česte situacije, recimo u logičkim teorijama, u kojima donja granica nije valjana za sve ili za skoro sve ulazne podatke, već samo neograničeno mnogo puta, što dalje komplikuje problem.

Druga vrsta pozicioniranja u hijerarhiji složenosti je relativna i izvodi se poređenjem odnosa složenosti problema u čemu postupak redukcije ima značajnu ulogu.

10.5 Redukcija problema

U analizi složenosti problema često se koristi svođenje jednog na drugi problem.

Definicija 10.5.1 Problem A se *redukuje* na problem B, u oznaci $A \leq B$, ako postoji izračunljiva funkcija f takva da je $A(x)$ tačno ako i samo ako je tačno i $B(f(x))$. Funkcija f se tada naziva *funkcija redukcije*.

Primetimo da redukovanje ima smisla samo ako je složenost izračunavanja funkcije redukcije zanemarljiva u odnosu na složenost problema B. Složenost izračunavanja funkcije redukcije se ograničava tako da pripada klasi L. Prema opisanoj hijerarhiji, funkcije redukcije pripadaju i klasi P, što je pogodnije za analizu problema u klasama sa vremenskim granicama složenosti. Uz to, ova klasa složenosti obezbeđuje da je veličina rezultata $f(x)$ takođe polinomijalno ograničena u odnosu na $|x|$.

[13] Lower bound.

Definicija 10.5.2 Funkcija redukcije f problema A na problem B je *efikasna*, a problem A je *efikasno reducibalan* na problem B, u oznaci $A \leq_{ef} B$, ako je složenost funkcije f u klasi L.

Primer 10.5.3 Verzija problema trgovačkog putnika u kojoj se ispituje da li postoji Hamiltonov put u grafu, tj. put koji kroz svaki čvor grafa prolazi tačno jednom se efikasno redukuje na problem SAT koji se odnosi na ispitivanje da li je proizvoljna klasična iskazna formula zadovoljiva. Neka graf G sadrži n čvorova. Odgovarajuća iskazna formula $R(G)$ će sadržati n^2 iskaznih slova $x_{i,j}$, $1 \leq i, j \leq n$ čije značenje je 'čvor j je i-ti čvor u Hamiltonovom putu'. $R(G)$ je formula u konjunktivnoj formi čije konjunkti su oblika:

- $x_{1,j} \vee \ldots \vee x_{n,j}$, za svako j, što znači da se svaki čvor mora pojaviti u putu,
- $\neg x_{i,j} \vee \neg x_{k,j}$, za svako j i $i \neq k$, što znači da se svaki čvor pojavljuje tačno jednom u putu,
- $x_{i,1} \vee \ldots \vee x_{i,n}$, za svako i, što znači da jedan čvor mora biti i-ti u putu,
- $\neg x_{i,j} \vee \neg x_{i,k}$, za svako i i $j \neq k$, što znači da se samo jedan čvor može biti i-ti u putu i
- $\neg x_{k,i} \vee \neg x_{k+1,j}$, za sve čvorove i i j koji nisu susedni u grafu G i $1 \leq k \leq n-1$.

Lako se pokazuje da interpretacija koja zadovoljava formulu $R(G)$ opisuje jedan Hamiltonov put u grafu. Slično, svaki Hamiltonov put u grafu definiše jednu interpretaciju koja zadovoljava $R(G)$. Sledeća Tjuringova mašina koja za ulaz koji opisuje graf G generiše na izlaznoj traci $R(G)$ pripada klasi L. Mašina na radnoj traci najpre predstavi u binarnoj formi broj n. Na osnovu toga se izgenerišu svi konjunkti formule $R(G)$ koji ne zavise od grafa G, za šta su potrebna tri indeksa i, j i k. U poslednjem koraku, pomoću istih indeksa se generišu na radnoj traci redom formule $\neg x_{k,i} \vee \neg x_{k+1,j}$, za sve čvorove i i j i $1 \leq k \leq n-1$, a zatim se proverava da li su odgovarajući čvorovi povezani u grafu G. Ako to nije slučaj, formula se prepiše na izlaznu traku. □

Definicija 10.5.4 Klasa problema $\mathcal{C}$ je *zatvorena za* $\leq_{ef}$ ako za svaki problem $B \in \mathcal{C}$ i svaki problem A važi da ako je $A \leq_{ef} B$, onda je i $A \in \mathcal{C}$.

Može se pokazati da su klase složenosti L, NL, P, NP, *co-NP*, $PSPACE$ i EXP zatvorene za redukciju.

Pod pretpostavkoma da je $A \leq_{ef} B$ upotrebom funkcije f, složenost problema A je odozgo ograničena zbirom složenosti problema B i funkcije redukcije f. Naime, za ispitivnje da li važi $A(x)$ najpre se x preslika u $f(x)$, a zatim se primeni program za utvrđivanje da li je $B(f(x))$. Dakle, ako su poznate složenosti problema B i funkcije f moguće je odrediti jednu gornju granicu složenosti problema A. Redukcija se može iskoristiti i za utvrđivanje donje granice složenosti. Ako je poznato da je složenost problema A veća od nekog zadatog nivoa, onda se kontrapozicijom može odrediti i jedna donja granica složenosti problema B.

10.6 Kompletni problemi

Definicija 10.6.1 Neka je B problem i $\mathcal{C}$ klasa složenosti. Tada kažemo:

- problem B je $\mathcal{C}$-*težak*[14], u oznaci $\mathcal{C} \leq_{ef} B$, ako je za svaki problem $A \in \mathcal{C}$ ispunjeno $A \leq_{ef} B$ i
- problem B je $\mathcal{C}$-*kompletan*[15] ako je $\mathcal{C} \leq_{ef} B$ i $B \in \mathcal{C}$.

Pojam kompletnog problema je značajan pošto svaki takav problem predstavlja klasu u onosu na koju je kompletan. Tako, na primer, NP-kompletan problem pripada klasi P ako i samo ako $P = NP$. Ovo je posledica tvrđenja 10.6.2 i činjenice da je klasa P zatvorena za $\leq_{ef}$.

Teorema 10.6.2 Neka su $\mathcal{C}$ i $\mathcal{D}$ klase složenosti, takve da je $\mathcal{D} \subset \mathcal{C}$ i $\mathcal{D}$ zatvorena za $\leq_{ef}$ i neka je B jedan $\mathcal{C}$-kompletan problem. Tada važi $B \in \mathcal{D}$ ako i samo ako $\mathcal{C} = \mathcal{D}$.

Postojanje prirodnih problema koji su kompletni za neku klasu složenosti daje klasi odgovarajući značaj, iako on možda nije jasan samo na osnovu njene definicije. Takav slučaj je, na primer, sa raznim nedeterminističkim klasama. U nastavku ćemo prikazati primere kompletnih problema za neke klase složenosti. Videćemo da su ti problemi proistekli iz stvarnih istraživanja, odakle proističe i značaj odgovarajućih klasa složenosti.

10.6.1 Klasa složenosti L

U klasi složenosti L se nalazi problem opisan u primeru 10.4.3 koji sadrži sve reči koje su palindromi, kao i svi problemi za grafove koji se mogu formulisati u klasičnom jeziku prvog reda[16]. Problem kompletnosti u ovoj klasi složenosti nije značajan pošto redukcija ima smisla samo u klasi koja je složenija od same redukcije. L je najmanja prirodna klasa složenosti jer je veličina binarne reprezentacije pokazivača na ulazni podatak x reda $\log_2 |x|$.

10.6.2 Problem GAP i NL-kompletnost

Problem GAP[17] se odnosi na utvrđivanje da li postoji put između dva zadata čvora grafa. Graf sa n čvorova se opisuje kvadratnom matricom u kojoj 1 znači da postoji put, a 0 da put između odgovarajućih čvorova ne postoji, dok se matrica reprezentuje kao binarna reč dužine n^2. Sada se GAP može definisati kao skup grafova u kojima postoji put od čvora 1 do čvora n.

Teorema 10.6.3 $GAP \in SPACE(O(\log_2^2 n))$.

[14] $\mathcal{C}$-*hard*.
[15] $\mathcal{C}$-*complete*.
[16] Recimo simetričnost grafa se opisuje sa $(\forall x)(\forall y)(G(x,y) \rightarrow G(y,x))$.
[17] Graph accessibility problem.

Dokaz. Pitanje da li su čvorovi x i y grafa G povezani, putem ne dužim od 2^i zapišimo u formi $Put(x, y, i)$. To se može rešiti rekurzivno, postavljanjem pitanja da li postoji čvor z takav da je $Put(x, z, i-1)$ i $Put(z, y, i-1)$. Kako je maksimalna dužina puta u grafu sa n čvorova $n-1$, čvorovi x i y su povezani ako i samo ako važi $Put(x, y, \log_2 n)$. Jedna implementacija ovog postupka podrazumeva da jedna radna traka Tjuringove mašine simulira izgled steka pri izvršavanju opisanog rekurzivnog postupka. Stek će sadržati najviše $\log_2 n$ trojki oblika (t_1, t_2, k). Svaki član ove trojke nije veći od n, pa je svaka trojka dugačka najviše $3\log_2 n$, što je dužina binarne reprezentacije broja n. Odatle, $GAP \in SPACE(O(\log_2^2 n))$. Ξ

Problem GAP karakteriše klasu NL.

Teorema 10.6.4 $GAP \in NL$.

Dokaz. Skiciraćemo dokaz postojanja nedeterminističke Tjuringove mašine koja prihvata niz od n različitih čvorova $z_1, \ldots, z_k$ (z_1 je čvor 1, z_k je čvor n) koji ima osobinu da povezuje čvorove 1 i n ako i samo ako graf koji se obrađuje pripada problemu GAP. Izbor čvorova počinje čvorom 1, nakon čega se u ciklusu nedeterministički bira sledeći čvor. Radna traka sadrži uvek po dva čvora predstavljena u binarnom obliku, zbog čega zauzimaju po $\log_2 n$ ćelija. U svakom ciklusu se proverava da li je na odgovarajućem mestu u matrici 1 ili 0. Rad mašine traje najviše $n-1$ ciklus i ako se pri tome pokaže povezanost, odgovor je pozitivan. Ξ

Teorema 10.6.5 GAP je NL-kompletan problem.

Dokaz. Zbog teoreme 10.6.4 potrebno je još samo pokazati da se svaki problem $B \in NL$ može redukovati na GAP, za šta se koristi metoda dostižnosti. Neka je M nedeterministička Tjuringova mašina koja rešava problem B u okviru logaritamski ograničenog prostora i neka je x ulazni podatak za M dužine $|x|$. Može se pokazati da klasi složenosti L pripada postupak kojim se konstruiše graf $G(M, x)$ koji sadrži konfiguracije u kojima se mašina M može naći obrađujući x. Očigledno je $x \in B$ ako i samo ako $G(M, x) \in GAP$, pa važi da je $B \leq_{ef} GAP$. Ξ

Među drugim primerima NL-kompletnih problema se nalazi i 2-SAT, problem ispitivanja zadovoljivosti klasične iskazne formule u konjunktivnoj normalnoj formi u kojoj svaki konjunkt ima najviše dva literala.

10.6.3 Problem CV i P-kompletnost

Problem CV[18] se odnosi na izračunavanje vrednosti izlaza logičkog kola u kome ulazne promenljive imaju fiksirane vrednosti. Formalnije, CV je skup sistema jednačina oblika

- $X_i = \top$,
- $X_i = \bot$,
- $X_i = \neg X_j$, za $j < i$ i

[18] Circuit value problem.

- $X_i = X_j \times X_k$, za $j, k < i$ i bilo koju binarnu logičku operaciju $\times$,

sa promenljivim X_1, ..., X_m, u kojima je vrednost promenljive X_m jednaka $\top$. Uslovi $j < i$ i $j, k < i$ u poslednja dva tipa jednačina obezbeđuju da ne postoji ciklična zavisnost promenljivih.

Teorema 10.6.6 $CV \in P$.

Dokaz. Sledeći postupak ispitivanja da li je neki sistem jednačina zaista pripada CV ima očigledno polinomijalnu vremensku granicu: vrednosti promenljivih X_i se izračunavaju u rastućem redosledu indeksa i. Ξ

Teorema 10.6.7 CV je P-kompletan problem.

Dokaz. Na osnovu teoreme 10.6.6 potrebno je još dokazati da se svaki problem $B \in P$ može redukovati na CV, odnosno pokazuje se da se svako izračunavanje iz klase P može opisati u formi jednog primerka problema CV. Ovde se razmatra deterministička Tjuringova mašina sa jednom trakom koja ima početak sa leve strane, što zbog teoreme 10.1.1 nije nikakvo ograničenje. Neka je M takva mašina i x ulazni podatak. Izračunavanje kojim mašina M prihvata x se može opisati kao niz reči oblika wqu, gde su w i u reči na alfabetu mašine, a sa q označena pozicija glave mašine u stanju q. Spomenute reči su zapravo konfiguracije mašine. Očigledno je da se ovaj opis može dati u formi tabele, pa se i metoda prikazivanja izračunavanja naziva *tablična metoda*[19] Ukupna dužina reči wqu je ograničena polinomijalno jer mašina M radi u polinomijalnom vremenu, a glava se ne može naći nad brojem ćelija većim od broja koraka rada. Svaka ćelija trake u kojoj su smeštene ovakve reči se može nazvati imenom neke promenljive čiji indeksi određuju korak izračunavanja i poziciju u konfiguraciji. Vrednost takve jedne promenljive $y_{i,j}$ je funkcija nekih promenljivih $y_{i-1,j-1}$, $y_{i-1,j}$ i $y_{i-1,j+1}$ iz prethodne konfiguracije. To ne odgovara u potpunosti definiciji problema CV, kao ni činjenica da vrednosti promenljivih nisu binarne. Neka je A skup svih znakova koji se javljaju u tabeli. Svaki znak iz A se može kodirati binarno, pri čemu je broj bitova ograničen sa $m = \log_2 |A|$. Sada je svaki bit promenljive $y_{i,j}$ funkcija $3m$ bita prethodnih promenljivih. Pošto je svaka logička funkcija, prema teoremama o normalnim formama, izraziva u obliku logičkog izraza, pa i logičkog kola, svih ovih m funkcija se može prikazati pomoću logičkog kola čija veličina zavisi samo od mašine M, a ne i od veličine ulaznog podatka x. Na ovaj način se svakoj mašini M i ulaznom podatku x pridružuje logičko kolo. Može se pokazati da je ova redukcija efikasna i da $M(x) \downarrow$ ako i samo ako je njoj pridružen problem u CV. Ξ

Drugi zanimljivi problemi koji pripadaju klasi P su: problem $PRIMES$ u kome se proverava da li je je dati prirodni broj prost, problem određivanja da li neka reč pripada kontekstno-slobodnom jeziku, nalaženje maksimalnog protoka u mreži, provera unifikabilnosti dva terma, provera da li sistem linearnih nejednačina sa racionalnim koeficijentima ima racionalna rešenja, problem $HORNSAT$ koji se odnosi na ispitivanje zadovoljivosti klasičnih iskaznih formula datih u obliku konjunktivne normalne forme u kojoj svaki konjunkt sadrži najviše jedno nenegiranano iskazno slovo itd.

[19] Table method. Ova metoda se obično koristi pri analizi klasa složenosti sa vremenskom granicom složenosti.

10.6.4 Problem SAT i NP-kompletnost

Problem SAT se precizno definiše kao skup svih zadovoljivih klasičnih iskaznih formula. Za njega, uprkos obimnom istraživanju, još uvek nije pokazano da li je, ili nije, u klasi P. Na ovom primeru se lako uočava razlika između determinističkog i nedeteminističkog izračunavanja. Umesto razmatranja cele istinitosne tablice koje se vrši u determinističkom slučaju, izborom interpretacije, tj. reda tablice, pri kojoj formula važi, lako se pokazuje da je formula zadovoljiva, što je zapravo pozivanje na teoremu 10.6.6:

Teorema 10.6.8 $SAT \in NP$.

U sledećoj teoremi je pokazano da je SAT jedan NP-kompletan problem, pa iz nje sledi da, ako bi bilo $SAT \in P$, onda bi važilo i $P = NP$. Ovo poslednje pitanje još nije razrešeno.

Teorema 10.6.9 (Kukova teorema) SAT je NP-kompletan problem.

Dokaz. Na osnovu teoreme 10.6.8 potrebno je pokazati da se svaki problem $B \in NP$ može u polinomijalnom vremenu redukovati na SAT. Neka je M nedeterministička Tjuringova mašina sa jednom trakom koja rešava problem B i neka je polinom $P(n)$ ograničenje za broj koraka u kojem mašina daje odgovor. Pretpostavimo bez gubitka opštosti da je $P(n) \geq n$. Opisaćemo konstrukciju funkcije izračunljive u polinomijalnom vremenu koja ulazni podatak x za B prevodi u klasičnu iskaznu formulu α tako da je α zadovoljivo ako i samo ako je B za ulaz $x = x_1x_2\ldots$ tačan. Neka alfabet mašine M ima r elemenata $\{s_1, \ldots, s_r\}$ i neka s_0 označava znak blanko. Neka je $|x|$ dužina ulaza za M i $t = P(|x|)$. Ako M pozitivno odgovara na problem B za x, onda to radi u najviše t koraka udaljavajući se najviše t ćelija udesno od polazne pozicije glave. Zato je dovoljno posmatrati deo trake koji sadrži $t+1$ ćeliju (polaznu i t ćelija desno od nje). Taj deo trake sigurno sadrži celi ulazni podatak jer je $t = P(|x|) \geq |x|$. Prema tome celo izračunavanje mašine M se može opisati matricom koja sadrži t redova i $t + 1$ kolonu. Neka su stanja mašine označena sa q_1, $\ldots$, q_m i neka je q_m jedinstveno završno stanje. Definišimo interpretaciju I iskaznih slova $\rho_{e,j,k}$ i $\sigma_{s_i,j,k}$ za $1 \leq e \leq m$, $0 \leq i \leq r$, $1 \leq j \leq t+1$ i $1 \leq k \leq t$, koja će se pojavljivati u formuli α:

$$I(\rho_{e,j,k}) = \begin{cases} \top & \text{ako je } M \text{ u stanju } q_e, \text{ nad ćelijom } j \\ & \text{u } k\text{-tom koraku izračunavanja} \\ \bot & \text{inače} \end{cases}$$

$$I(\sigma_{s_i,j,k}) = \begin{cases} \top & \text{ako je znak } s_i \text{ u ćeliji } j \text{ u } k\text{-tom koraku izračunavanja} \\ \bot & \text{inače} \end{cases}$$

Sa $\underline{\vee_{i=1}^{l}}\beta_i$ označićemo formulu koja je istinita ako i samo ako je istinita tačno jedna od formula β_i, $1 \leq i \leq l$. Ako su β_i iskazna slova ova formula se u konjunktivnoj normalnoj formi može zapisati kao konjunkcija formula oblika $\neg\beta_i \vee \neg\beta_t$, za $i \neq t$ i formule $\vee_{i=1}^{l}\beta_i$. Broj formula prvog oblika je $l(l-1)/2$. Svaka od njih se se može prikazati u 3 ćelije u obliku para i, j za kojim sledi znak / kojim se uzastopni parovi razdvajaju. Slično, druga formula se predstavlja u $l + 1$ ćeliji kao niz 1, 2, $\ldots$, l, /, pa je ukupan broj ćelija za prikazivanje formule $\underline{\vee_{i=1}^{l}}\beta_i$ u $O(l^2)$.

Simulacija rada mašine M biće opisana konjunkcijom formula koje su u konjunktivnoj normalnoj formi ili se u nju prevode i koje su sve tačne pri interpretaciji I. Ovu konjunkciju označimo sa α i primetimo da je i ona u konjunktivnoj normalnoj formi. Formule koje ulaze u konjunkciju su:

- Formulom
$$(\wedge_{j=1}^{|x|}\sigma_{x_j,j,1}) \wedge (\wedge_{j=t-|x|}^{t+1}\sigma_{s_0,j+1,1}) \wedge \rho_{1,1,1}$$
se kaže da je u polaznoj konfiguraciji u glava mašine nad najlevljom ćelijom koja ne sadrži blanko simbol. Dužina ove formule je u $O(t)$.

- Formulom
$$\wedge_{k=1}^{t} \underline{\vee}\{\rho_{e,j,k} : e = 1, m, j = 1, t+1\}$$
se kaže da s u svakom koraku izračunavanja mašina nalazi u tačno jednom od m stanja, a da je glava nad tačno jednom od posmatranih $t+1$ ćelija. Dužina ove formule je u $O(t^3)$.

- Formulom
$$\wedge_{k=1}^{t} \wedge_{j=1}^{t+1} \underline{\vee}\{\sigma_{s_i,j,k} : i = 0, r\}$$
se kaže da se u svakom trenutku u svakoj ćeliji nalazi tačno jedan simbol. Dužina ove formule je u $O(t^2)$.

- Formulom
$$\vee_{j=1}^{t+1}\rho_{m,j,t}$$
se kaže da se nakon t koraka rada mašina nalazi u završnom stanju q_m. Dužina ove formule je u $O(t)$.

- Poslednjom formulom se kaže da se svaka, sem početne konfiguracije, dobija iz prethodne primenom neke naredbe:
$$\wedge_{k=1}^{t-1} \wedge_{j=1}^{t+1} (NG(j,k) \vee ID(j,k) \vee A(j,k) \vee B(j,k) \vee C(j,k))$$
gde su:
  - $NG(j,k) = (\wedge_{e=1}^{m}\neg\rho_{e,j,k}) \wedge (\vee_{i=0}^{r}(\sigma_{s_i,j,k} \wedge \sigma_{s_i,j,k+1}))$ formula koja je tačna ako i samo mašina M nije u k-tom koraku iznad ćelije j ni u kom stanju i sadržaj ćelije j se ne menja nakon izvršavanja k-tog koraka rada mašine.
  - $ID(j,k) = \vee_{e=1}^{m} \vee_{i=0}^{r} (\rho_{e,j,k} \wedge \sigma_{s_i,j,k} \wedge \rho_{e,j,k+1} \wedge \sigma_{s_i,j,k+1})$ formula koja je tačna ako se mašina i u k-tom i u $k+1$-om koraku rada nalazi iznad ćelije j u stanju e, pri čemu se ne menja sadržaj ćelije.
  - Formule $A(j,k)$, $B(j,k)$ i $C(j,k)$ se odnose na akcije koje se dešavaju izvršenjem naredbi, i to A za upisivanje novog simbola, B za kretanje glave trake udesno i C za pokretanja glave trake ulevo:
    $A(j,k) = \vee_{a=1}^{a_0}(\rho_{e_a,j,k} \wedge \sigma_{s_{i,a},j,k} \wedge \rho_{e'_a,j,k+1} \wedge \sigma_{s'_{i,a},j,k+1})$,
    $B(j,k) = \vee_{b=1}^{b_0}(\rho_{e_b,j,k} \wedge \sigma_{s_{i,b},j,k} \wedge \rho_{e'_b,j+1,k+1} \wedge \sigma_{s_{i,b},j,k+1})$ i
    $C(j,k) = \vee_{c=1}^{c_0}(\rho_{e_c,j,k} \wedge \sigma_{s_{i,c},j,k} \wedge \rho_{e'_c,j,k+1} \wedge \sigma_{s'_{i,c},j-1,k+1})$.
    U sve tri formule disjunkcije se odnose na sve naredbe odgovarajućeg oblika.

Pošto je dužina svakog od disjunkata konstantna, dužina cele ove formule je u $O(t^2)$.

Intuitivno je jasno da je formula α zadovoljiva pri interpretaciji I ako Tjuringova mašina M konvergira za ulaz x. Važi i obrnuto, tj. ako je $I(\alpha) = \top$, mašina M konvergira za ulaz x. Naime, na osnovu potformula formule α može se jednoznačno rekonstruisati niz konfiguracija koje od polaznog dovode do završnog stanja.

Konačno, za sve formule koje sačinjavaju α važi da su im dužine polinomijalno ograničene u odnosu na $|x|$, pa se mogu konstruisati Tjuringovom mašinom čija je dužina rada ograničena polinomijalnom funkcijom od $|x|$. Ξ

Zanimljivo je da teorema slična tvrđenju 10.6.9 važi i za takozvane CNF-SAT i 3-SAT probleme u kojima se ispituje zadovoljivost formula u konjunktivnoj normalnoj formi, odnosno u konjunktivnoj normalnoj formi u kojoj ni jedan konjunkt ne sadrži više od 3 literala.

Teorema 10.6.10 CNF-SAT i 3-SAT su NP-kompletni problemi.

Dokaz. Primetimo najpre da se formula iz dokaza teoreme 10.6.9 može transformisati u konjunktivnu normalnu formu. Jedino poslednja formula nije u ovom obliku, ali svi njeni članovi su konstantne dužine koja zavisi samo od m, broja stanja mašine, i r, broja elemenata alfabeta mašine, a ne i od ulaznog podatka. Te formule se standardnim postupkom prevode u konjunktivnu normalnu formu u kojoj je svaki konjunkt konstantne dužine. Odatle direktno sledi da je problem CNF-SAT takođe NP-kompletan. Nastavak dokaza počiva na zadatku 11.59 u kome se pokazuje da se u konjunktivnoj normalnoj formi svaki konjunkt D koji sadrži bar 4 literala može prevesti u formulu u konjunktivnoj normalnoj formi čiji svaki konjunkt sadrži tačno 3 literala. Dužina te formule je u $O(|D|)$, pa se transformacija može izvesti u linearnom vremenu. Dobijena formula je zadovoljiva ako i samo ako je zadovoljiv polazni konjunkt. Ξ

Dalja redukcija problema SAT na problem 2-SAT je drugačijeg karaktera. U odeljku 10.6.2 je spomenuto da je 2-$SAT \in NL$, pa bi se tom prilikom, ako je tačna pretpostavka da je $P \neq NP$, smanjila složenost problema.

Klasa NP sadrži veliki broj realnih i značajnih problema među kojima su: problem trgovačkog putnika, problem određivanja Hamiltonovog puta, provera da li se čvorovi grafa mogu obojiti korištenjem 3 boje, tako da susedni čvorovi nisu iste boje, problem $CLIQUE$[20], uopšte svi problemi za grafove koji se mogu formulisati kao izrazi logike drugog reda u formi $(\exists P)\alpha$, gde je P relacijska promenljiva drugog reda i α formula prvog reda[21], problem *celobrojnog programiranja*[22], tj. problem

[20] Clique u neorijentisanom grafu je podgraf čija su svaka dva čvora direktno povezani. k-clique je clique koji ima tačno k čvorova. $CLIQUE = \{\langle G, k\rangle : G$ je neorijentisani graf koji ima k-clique$\}$.

[21] Recimo, problem nepostojanja puta u grafu G između dva čvora se opisuje formulom $\phi(x, y) = (\exists P)(\forall u)(\forall v)(\forall w)(P(u, u) \wedge (G(u, v) \rightarrow P(u, v) \wedge (P(u, v) \wedge P(v, w) \rightarrow P(u, w)) \wedge \neg P(x, y)$ u kojoj P predstavlja refleksivno i tranzitivno zatvorenje grafa G u kome čvorovi x i y nisu direktno povezani. Zanimljivo je da je dokazano da se klasa složenosti NP poklapa sa klasom problema koji se mogu opisati kao izrazi logike drugog reda u formi $(\exists P)\alpha$.

[22] Integer programming.

ispitivanja da li sistem linearnih jednačina sa celobrojnim koeficijentima ima nenegativna celobrojna rešenja[23], zadovoljivost u modalnoj logici $S5$ i mnogi drugi. Na primer, za poslednji problem, $S5SAT$, pripadanje klasi NP, tj. nedeterministička polinomijalna granica složenosti se određuje tako što se dokaže da je modalna formula α $S5$-zadovoljiva ako i samo ako je zadovoljiva u Kripkeovom $S5$-modelu koji ima najviše $|\alpha|$ svetova, gde je $|\alpha|$ dužina zapisa formule. Sa druge strane, pošto logika $S5$ uključuje klasičnu logiku, važi $NP \leq_{ef} S5SAT$. Odatle je $S5SAT$ jedan NP-kompletan problem.

S obzirom da je pretpostavka da $P \neq NP$, klasa složenosti NP ima za kompletne probleme one za koje se pretpostavlja da nije korisno tražiti opšti algoritam koji bi bio efikasan, već da je bolje koncentrisati se na neke specijalne slučajeve (kao što je 2-SAT u odnosu na SAT), razvijati aproksimativne postupke ili eksponencijalne koji su efikasni za manje primerke problema.

10.6.5 Klase složenosti *co*-NP i $NP \cap$ *co*-NP

Klasa složenosti *co*-NP sadrži probleme kod kojih je potvrda da x ne pripada problemu relativno laka. Pošto je iskazna formula tautologija ako i samo ako njena negacija nije zadovoljiva, a za izabranu interpretaciju se lako proverava da li zadovoljava formulu, problem tautologičnosti je u klasi *co*-NP. Slično, komplement problema određivanja Hamiltonovog puta koji sadrži sve grafove koji nemaju Hamiltonov put je u klasi složenosti *co*-NP. Provera da graf ne pripada ovom problemu se sprovodi utvrđivanjem da je izabrani put u njemu Hamiltonov. Dokazuje se da su komplementi NP-kompletnih problema *co*-NP-kompletni, pa su i navedeni problemi takvi.

Za svaki primerak problema koji pripada klasi složenosti $NP \cap$*co*-NP važi tačno jedno od:

- primerak ima laku potvrdu ili
- primerak ima lako opovrgavanje.

Svaki problem iz P je u klasi $NP \cap$ *co*-NP, ali postoje problemi iz klase $NP \cap$ *co*-NP za koje se ne zna da li su u P. Do sada nije odgovoreno da li je $NP \neq$ *co*-NP, mada se u to veruje.

10.6.6 Problem QBF i $PSPACE$-kompletnost

Za rešavanje problema koji pripadaju klasi $PSPACE$ dovoljno je raspolagati memorijom polinomijalno ograničene veličine, ali se vreme izračunavanja ne ograničava.

Problem QBF[24] je skup klasičnih iskaznih formula α za koje važi $(\exists x_1)(\forall x_2)$ $(\exists x_3)$ $(\forall x_4) \ldots (Qx_m)\alpha$ odnosno, postoji istinitosna vrednost promenljive x_1 takva da za svaku isitinitosnu vrednost promenljive x_2 postoji istinitosna vrednost promenljive x_3 takva da za svaku isitinitosnu vrednost promenljive x_4 ... formula α bude zadovoljiva. Primetimo da je redosled kvantifikatora u formuli alternirajući, te da

[23] Problem se može ekvivalentno formulisati i tako da se razmatraju linearne nejednačine, a rešenja ne moraju biti nenegativna.

[24] Quantified Boolean formula. Ovaj problem se nekada naziva i $QSAT$ čime se naglašava da je i ovo jedna verzija problema zadovoljivosti.

je kvantifikator označen sa Q univerzalni ako je m parno, odnosno egzistencijalni, ako je m neparno.

Teorema 10.6.11 QBF je $PSPACE$-kompletan problem.

Dokaz. Pretraživanje kojim se analiziraju sve interpertacije i utvrđuje da li primerak problema zadovoljava uslove problema može se realizovati rekurzivnim programom koji kao argument dobija formulu i:

- ako ne sadrži kvantifikatore, izračunava i vraća istinitosnu vrednost formule,
- ako formula počinje sa $(\exists x_i)$ rekurzivno poziva sam sebe, najpre zamenivši x_i sa $\top$, a zatim sa $\bot$; program vraća vrednost $\top$, ako je bar jednom vraćen rezultat $\top$, inače vraća vrednost $\bot$ i
- ako formula počinje sa $(\forall x_i)$ rekurzivno poziva sam sebe, najpre zamenivši x_i sa $\top$, a zatim sa $\bot$; program vraća vrednost $\top$, ako je oba puta vraćen rezultat $\top$, inače vraća vrednost $\bot$.

Pošto je maksimalna dubina rekurzivnih poziva jednaka broju promenljivih u formuli, za izvršavanje programa se ne zahteva više od prostora polinomijalno ograničenog veličinom ulaza, pa je $QBF \in PSPACE$. U dokazu da se svaki problem $B \in PSPACE$ redukuje na QBF može se primeniti metod dostižnosti. Ξ

Klasa složenosti $PSPACE$ je zanimljiva jer sadrži više problema koji se mogu shvatiti kao igre u kojima učestuju dva igrača, kakve su igra 'iks-oks' i go[25]. I problem QBF se može shvatiti kao jedna takva igra koju igraju osobe $\exists$ i $\forall$. Prvi potez pripada igraču $\exists$, nakon čega se igra naizmenično. U koraku $i = 2k+1$ igrač $\exists$ bira vrednost promenljive x_i pokušavajući da formula α bude tačna, dok u koraku $i = 2k + 2$ igrač $\forall$ bira vrednost promenljive x_i pokušavajući da formula α bude netačna. Nakon m koraka igra završava i jedan od igrača pobeđuje. U ovakvim igrama rešenje nije neki objekat za koji se lako proverava da li zadovoljava uslov, kao što je to slučaj u klasi NP, već strategija kojom igrač dolazi do pobede. Igrač $\exists$ ima pobedničku strategiju ako pobeđuje u slučaju da oba igrača igraju optimalno.

Prema teoremi 10.6.11, problem koji sadrži sve formule u kojima $\exists$ ima pobedničku strategiju je $PSPACE$-kompletan. Slično, problem za igru go sa donekle izmenjenim pravilima, recimo da je tabla dimenzija $n \times n$, za proizvoljno n, koji se odnosi na postojanje strategije za pobedu prvog igrača je $PSPACE$-kompletan. Isto važi (u odnosu na veličinu opisa zahteva) i za problem verifikacije, tj. da li izvršanja programa P zadovoljavaju zahtev, o čemu je bilo reči u odeljku 9.8.1. Dalje, $PSPACE$-kompletni su i problemi RD[26] u kome se ispituje da li je za dati sistem procesa koji komuniciraju i neko inicijalno stanje moguće stići u stanje u kome su svi procesi zaglavljeni čekajući međusobno jedan drugog, provera modela o kojoj je bilo reči u odeljku 9.8.1, zadovoljivost za modalne logike K_n, T_n, $S4_n$, za $n \geq 1$ i za $S5_n$, za $n \geq 2$, u kojima postoji n verzija modalnog operatora $\Box$. Za navedene modalne logike se pokazuje da su odgovarajući problemi zadovoljivosti

[25] Iako i šah pripada ovoj vrsti igara, zbog unapred fiksirane veličine table šah nije pogodno analizirati metodama koje se koriste u teoriji složenosti zasnovanoj na automatima.

[26] Reachable deadlock.

u klasi $PSPACE$ konstrukcijom postupka provere zasnovanom na metodi tabloa. Da je $PSPACE$ i donja granica složenosti sledi iz postojanja formula koje su zadovoljive u spomenutim logikama samo u eksponencijalno velikim modelima koji su drvoidnog oblika, a čije grane imaju polinomijalno ograničenu dužinu. Primetimo i da, dok je $S5$-zadovoljivost NP-kompletan problem, već dodavanje drugog operatora $\Box$ podiže granicu složenosti.

10.6.7 Klasa složenosti EXP i njena proširenja

Problemi koji su EXP-kompletni ili kompletni u odnosu na klase koje su veće složenosti od EXP, prema teoremi 10.4.9.168, sigurno ne pripadaju klasi P zbog čega se smatra da nisu praktično izračunljivi.

Klasa složenosti EXP obuhvata probleme koji su rešivi determinističkim sredstvima u eksponencijalnom vremenu, poput problema zadovoljivosti u nekim modalnim logikama. Na primer, zadovoljivost formula u iskaznoj dinamičkoj logici PDL je jedan EXP-kompletan problem.

Klasa složenosti $NEXP$ je nedeterministička verzija klase EXP. Jedan njen kompletan problem je ispitivanje zadovoljivosti formula oblika $\beta = (\exists x_1) \ldots (\exists x_k)$ $(\forall y_1) \ldots (\forall y_n)\alpha$ na jeziku prvog reda bez funkcijskih simbola i simbola jednakosti, gde je α formula bez kvantifikatora. Dokazuje se da je ovakva formula zadovoljiva ako i samo ako je zadovoljiva u modelu sa najviše $k + m$ elemenata, gde je m broj simbola konstanti u formuli α. Pošto je $k + m \leq |\beta|$, a za testiranje modela je potrebno eksponencijalno vreme, direktno sledi da problem pripada klasi $NEXP$. Kompletnost se pokazuje uobičajenim postupkom, opisivanjem proizvoljnog programa Tjuringove mašine koji pripada klasi $NEXP$ formulom navedenog oblika.

Nakon klase $NEXP$ pojavljuju se šire klase složenosti $EXPSPACE$, 2-EXP, 2-$NEXP$ itd, tako da je ovaj deo hijerarhije klasa složenosti oblika:

$$EXP \subset NEXP \subset EXPSPACE \subset \text{2-}EXP \subset \text{2-}NEXP \subset \ldots$$

Klasa $EXPSPACE$ sadrži probleme za čije rešavanje je potreban eksponencijalno veliki prostor, dok klasa $2 - EXP$ sadrži probleme za čije rešavanje je potrebno dvostruko eksponencijalno vreme. Ova hijerarhija klasa sa eksponencijalnom složenošću se beskonačno širi, a unija svih tih klasa se naziva klasa *elementarnih jezika*. Zanimljivo je da postoje stvarni, odlučivi, problemi koji ne pripadaju čak ni ovoj klasi složenosti[27].

10.7 Verovatnosne klase složenosti

Postoje problemi za koje se ne zna da li pripadaju klasi složenosti P, ali koji se upotrebom slučajnih vrednosti mogu rešavati, do na neku verovatnoću, algoritmima sa polinomijalnom vremenskom granicom složenosti. Algoritmi ove vrste se nazivaju *algoritmi sa slučajnošću*[28] ili *verovatnosni algoritmi*.

Primer 10.7.1 Neka je $ZERO$ skup svih polinoma (od više promenljivih) sa celobrojnim koeficijentima koji su identički jednaki nuli. Na primer, polinom

[27] Recimo, jednakost izraza u nekim regularnim jezicima.
[28] Randomized algorithms.

$(x_1 + x_2)(x_1 - x_2) - x_1^2 + x_2^2 \in ZERO$. Tada je *co-ZERO* skup svih polinoma koji nisu identički jednaki nuli. Neka je polinom $F(x_1, \ldots, x_m) \in$ *co-ZERO*, takav da je maksimalan stepen svake od promenljivih manji do jednak od d. Tada je broj m-torki $(x_1, \ldots, x_m) \in \{0, 1, \ldots, M-1\}^m$ za koje je $F(x_1, \ldots, x_m) = 0$ manji do jednak od mdM^{m-1}. Pretpostavimo da se je $M = 2md$. Tada je verovatnoća izbora m-torke iz skupa $\{0, 1, \ldots, M-1\}^m$ koja je nula polinoma F koji nije nula-polinom manja do jednaka od $\dfrac{2^{m-1}m^m d^m}{2^m m^m d^m} = \dfrac{1}{2}$. Verovatnosno ispitivanje problema se sprovodi na sledeći način. Najpre se izabere m slučajnih celih brojeva $k_1, \ldots, k_m \in \{0, \ldots, 2md-1\}$. Ako je vrednost izraza $F(k_1, \ldots, k_m) \neq 0$, prihvata se da je $F \in$ *co-ZERO*, a u suprotnom da je verovatno da je $F \notin$ *co-ZERO*. U prvom slučaju, odgovor je sigurno tačan, pošto polinom F očigledno nije nula-polinom. U drugom slučaju moguće su dve situacije:

- F je zaista nula-polinom i
- F nije nula-polinom, ali je izabrana m-torka nula polinoma. Verovatnoća da se ovo dogodi nije veća od $\frac{1}{2}$.

Dakle, na pitanje da li je $F \in$ *co-ZERO*, potvrdan odgovor je uvek tačan, dok je negativan odgovor pogrešan sa verovatnoćom manjom do jednakom od $\frac{1}{2}$. Izborom M da bude znatno veće od $2md$ verovatnoća greške se smanjuje, ali je potrebno računati sa većim brojevima. Smanjenje verovatnoće greške se zato postiže ponavljanjem opisanog postupka k puta. Ako se u svim slučajevima dobija da je vrednost polinoma u izabranim tačkama jednaka 0, onda je verovatnoća greške odgovora $F \notin$ *co-ZERO* manja do jednaka od $\frac{1}{2^k}$. Ξ

Algoritmi poput opisanog u kojima ne postoji pogrešan pozitivan odgovor, a verovatnoća pogrešnog negativnog odgovora je manja od 1 se nazivaju *Monte-Karlo algoritmi*. Još jedan primer Monte-Karlo algoritma odnosi se na ispitivanje da li je dati broj prost.

Primer 10.7.2 Neka je $p > 2$ prost broj i neka je $Z_p = \{0, 1, \ldots, p-1\}$. Za bilo koji $a \in Z_p \setminus \{0\}$ *Ležandrov simbol za a i p*, u oznaci $(a|p)$, označava izraz $a^{\frac{p-1}{2}} \mathrm{m}od\ p$. Važi da je $(a|p) \in \{-1, 1\}$[29]. Ako su $M, N \in \mathbb{N}$ i $N = q_1 \cdots q_n$, gde su svi q_i ne nužno različiti neparni prosti brojevi, onda je $(M|N) = \Pi_{i=1}^n (M|q_i)$. Za izračunavanje $(M|N)$ postoji efikasan postupak čija je vremenska granica složenosti $O(\log_2^3 MN)$. Može se pokazati da ako je N neparan složen broj, za barem polovinu brojeva $M \in \{2, \ldots, N-1\}$ važi da je $(M|N) \neq M^{\frac{N-1}{2}} (\mathrm{m}od\ N)$, a ako je N prost, onda za sve brojeve $M \in \{2, \ldots, N-1\}$ važi da je $(M|N) = M^{\frac{N-1}{2}} (\mathrm{m}od\ N)$. Na osnovu toga se formuliše Monte-Karlo algoritam za ispitivanje da li neki broj složen. Neka je dat neparan prirodan broj $N > 1$. Slučajno se izabere prirodan broj $M \in [2, N-1]$ i ispita da li su M i N uzajamno prosti. Ako nisu, N je složen. Ako jesu, izračunaju se $(M|N)$ i $M^{\frac{N-1}{2}} \mathrm{m}od\ N$ i uporede. Ako su brojevi različiti odgovara se 'da, N je složen', a ako su jednaki odgovara se 'ne, N je verovatno

[29] Koristi se Mala Fermaova (Pierre de Fermat, 1601 – 1665) teorema da za svaki prost broj p i $0 < a < p$, $a^{p-1} = 1 (\mathrm{m}od\ p)$. Zato je $(a|p)$ kvadratni koren iz 1 u Z_p, čija je vrednost za prost broj p uvek ± 1.

prost'. Na osnovu iznetog se vidi da je pozitivan odgovor (da je broj složen) uvek tačan, a da je verovatnoća greške negativnog odgovora (proglašenje složenog broja da je prost) najviše $\frac{1}{2}$. Ξ

U analizi složenosti verovatnosnih algoritama se koriste verovatnosne Tjuringove mašine koje se od nedeterminističkih razlikuju samo po kriterijumu prihvatanja primeraka problema.

Definicija 10.7.3 *Verovatnosna Tjuringova mašina*[30] je nedeterministička Tjuringova mašina sa sledećim osobinama:

- za svaki ulazni podatak sva izračunavanja se završavaju u jednakom broju koraka koji je ograničen polinomijalnom funkcijom $p(|x|)$ i
- u svakom koraku izračunavanja koji nije poslednji postoji tačno dva moguća nastavka.

Verovatnosna mašina odlučuje problem B ako za svaki $x \in B$ barem polovina od $2^{p(|x|)}$ izračunavanja završi prihvatanjem, a za svaki $x \notin B$ sva izračunavanja završe odbacivanjem.

Verovatnosna klasa složenosti RP[31] sadrži sve probleme B za čije rešavanje postoji verovatnosna Tjuringova mašine sa polinomijalnom vremenskom granicom složenosti.

Izbor mogućeg nastavka izračunavanja se može shvatiti kao da je realizovan bacanjem novčića čime se bira 0 ili 1. U slučaju složenijeg izbora, recimo broja u nekom opsegu, on se razbija na niz izbora bitova. Prema tome, verovatnoća svakog izračunavanja je jednaka $\frac{1}{2^{p(|x|)}}$. Definicija klase RP obezbeđuje da nema pogrešnih pozitivnih odgovora, jer se odgovara 'da' samo ako postoji izračunavanje koje prihvata ulazni podatak, dok je verovatnoća pogrešnog odgovora 'ne' najviše $\frac{1}{2}$. Primetimo i da verovatnoća pogrešnog negativnog odgovora može biti bilo koji broj iz $[\frac{1}{2}, 1)$, pošto se uzastopnim ponavljanjem postupka može proizvoljno smanjiti.

Opisani problemi *co-ZERO* i provere složenosti su u klasi RP. Očigledno je da se u klasi RP nalaze problemi koji pripadaju klasi NP, ali imaju osobinu da za svaki x koji pripada problemu postoji bar pola izračunavanja koja ga prihvataju, dok ni jedno izračunavanje ne prihvata onaj x koji ne pripada problemu. Istovremeno, svi problemi $B \in P$ zadovoljavaju uslov prihvatanja $x \in B$ sa verovatnoćom 1, dok nema ni jednog izračunavanja koje prihvata $x \notin B$. Iz ovog objašnjenja direktno sledi teorema 10.7.4.

Teorema 10.7.4 $P \subset RP \subset NP$.

U narednoj verovatnosnoj klasi se nalaze problemi za koje je dozvoljena izvesna greška u rešavanju i za one primerke koji pripadaju i za primerke koje ne pripadaju problemu.

[30] Koristi se i nazivi polinomijalna Monte-Karlo Tjuringova mašina i precizna Tjuringova mašina. U odeljku 17 je u ukratko opisano kako se proizvoljna nedeterministička mašina može transformisati u preciznu.

[31] Random polynomial time.

Definicija 10.7.5 *Verovatnosna klasa složenosti* BPP[32] sadrži svaki problem B za čije rešavanje postoji nedeterministička Tjuringova mašine sa polinomijalnom vremenskom granicom složenosti u kojoj su sva izračunavanja iste dužine, takva da:

- za svaki $x \in B$, barem $\frac{3}{4}$ izračunavanja prihvataju x i
- za svaki $x \notin B$, barem $\frac{3}{4}$ izračunavanja ne prihvataju x.

Slično kao i ranije, bilo koji broj $\delta \in (\frac{1}{2}, 1]$ može zameniti $\frac{3}{4}$ u definiciji bez promene klase BPP. Jasno je da je $RP \subset BPP$, ali se ne zna da li je $BPP \subset NP$. Za sada nisu pronađeni kompletni problemi za klase RP i BPP.

Klase RP i BPP su, pod jednim uslovom, veoma praktične generalizacije klase P jer verovatnoća greške eksponencijalno opada u odnosu na broj ponavljanja algoritma[33] i može se učiniti proizvoljno malom, što izgleda kao efikasna alternativa izračunavanju koje je u klasi složenosti NP. Uslov njihove praktičnosti je implementacija izvora slučajnih bitova. Međutim, postizanje istinske slučajnosti može biti teško ili, čak, nemoguće, tako da se slučajni nizovi po pravilu dobijaju determinističkim programima koje nazivamo pseudoslučajnim generatorima. Njihov izlaz, iako nije potpuno slučajan, može imati mnoge karakteristike slučajno generisanog niza bitova. Pokazuje se da postoje pseudoslučajni generatori čiji izlaz primenom algoritama polinomijalne vremenske složenosti ne možemo razlikovati od istinski slučajnih nizova. Na žalost, pseudoslučajni generatori koje koriste računari se ne realizuju uvek zadovoljavajuće kvalitetno.

10.8 Primene teorije složenosti u kriptologiji

Teorija složenosti izračunavanja ima značajne primene u kriptologiji[34]. Recimo, privatnost, odnosno sigurnost, razgovora preko mobilnih telefona i bezbednost kupovine preko Interneta zavise od zaštite informacija kriptološkim merama koje se često zasnivaju na složenosti izračunavanja odgovarajućih algoritama.

10.8.1 Kriptološki sistemi

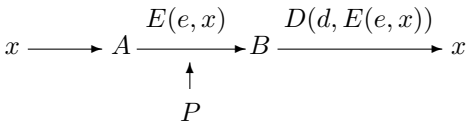
U ovom odeljku razmotrićemo jednu vrstu situacija, prikazanu na slici 10.1 u kojima se koristi kriptologija. Dve osobe[35] komuniciraju i treba da onemoguće da treća osoba, prisluškivač, razume njihove poruke. Recimo da osoba A želi da pošalje poruku osobi B. Poruku shvatamo kao binarnu reč. Osobe A i B se prethodno dogovaraju oko algoritama E i D za kodiranje (kriptovanje), odnosno dekodiranje, poruka koji imaju osobinu da za svaku poruku x važi $D(d, E(e, x)) = x$, gde su e i d dve binarne reči, *ključevi* za kodiranje i dekodiranje, koje se biraju tako da funkcije

[32] Bounded probability error.

[33] Ako je verovatnoća pogreške ograničena sa t, nakon k nezavisnih ponavljanja algoritma, verovatnoća pogreške je ograničena sa t^k.

[34] Kriptologija je nauka koja se bavi konstrukcijom i analizom postupaka koji obezbeđuju bezbednost informacija i komunikacija.

[35] Tradicionalno se osobe nazivaju agenti koji komuniciraju i daju im se imena Alice i Bob.



Slika 10.1. Komunikacija dve strane u prisustvu prisluškivača.

E i D postanu inverzne. Da bi se postigla efikasnost potrebno je da funkcije E i D budu polinomijalne složenosti, dok se privatnost obezbeđuje time što su e i d *tajni*[36], tj. znaju ih samo osobe A i B, a x se ne može efikasno izračunati iz $E(e, x)$ bez poznavanja ključa d.

Primer 10.8.1 Za funkcije kodiranja i dekodiranja i odgovarajuće ključeve mogu se izabrati sabiranje po modulu 2, tj. ekskluzivno ili, i bilo koja binarna reč a koja ima istu dužinu kao i poruka koju treba preneti. Pošto je $D(a, E(a, x)) = (x \oplus a) \oplus a = x$ postignuta je inverznost kodiranja i dekodiranja, a pošto je $E(a, x) \oplus x = (x \oplus a) \oplus x = a$, prisluškivač može dekodirati kodiranu poruku ako i samo ako zna a, pa je obezbeđena i privatnost. Ξ

Primetimo da je u primeru 10.8.1 problematičan dogovor osoba A i B koji se obavlja pre komunikacije, jer je reč o razmeni ključa koji treba da ostane tajan i koji je iste dužine kao i poruka. Ovo nije praktično jer pretpostavlja rešavanje zadatka za čije rešavanje ključ treba da posluži. Problem predstavlja i što postoje postupci, takozvani napadi, koji:

- ako je ključ kratak, vrše pretragu kroz prostor svih mogućih ključeva ili
- ako se ključ koristi više puta u kodiranju, vrše razne analize

i time otkrivaju, ili kako se u žargonu kaže *razbijaju*, tajni ključ. Izlaz iz ovakve situacije je da se svaki bit ključa koristi samo jednom za kodiranje samo jednog bita informacije i zatim odbacuje, što takođe često nije ostvarljiv zahtev.

U teorijskom smislu apsolutna sigurnost kodiranja upotrebom ključa umerene dužine nije moguća, ali se izborom ključa dovoljne dužine, recimo reda veličine 100 bita, postiže praktična sigurnost koja predstavlja verovanje da se ključ ne može brzo otkriti. Ovde teorija složenosti izračunavanja pruža metodologiju za utvrđivanje sigurnosti koda. Već je rečeno da se, prema hipotezi $P \neq NP$, za problem koji je kompletan za klasu složenosti NP smatra da nema efikasnog načina izračunavanja. Zato se redukovanjem nekog NP-kompletnog problema na problem razbijanja ključa utvrđuje i složenost potonjeg. Na žalost, ovo nije dovoljan dokaz pošto se NP-kompletnost odnosi na slučajevi u kojima se algoritmi najgore ponašaju, kao što je objašnjeno u odeljku 10.10, tako da se neki primerci NP-kompletnih problema mogu rešavati lako. Zato su ovde primenljiviji algoritmi koji su složeni u odnosu na prosečni slučaj, kao što je faktorizacija prirodnih brojeva, pa su neki savremeni kriptološki sistemi bazirani upravo na tom problemu.

[36]Secret key.

U stvarnosti je često problematična i razmena tajnih ključeva, pa u kriptološkim sistemima sa *javnim ključem*[37] svaka osoba B ima par ključeva: javni, svima dostupni, ključ e_B za kodiranje i privatni ključ d_B za dekodiranje poruka. Bilo koja osoba koja želi da pošalje osobi B poruku za kodiranje koristi javni ključ e_B.

Teorija složenosti izračunavanja pomaže u objašnjavanju pretpostavki koje obezbeđuju sigurnost i sistema sa tajnim i sistema sa javnim ključem.

Definicija 10.8.2 Funkcija f je *jednosmerna* ako važi:

- f je $1-1$ funkcija,
- vremenska granica složenosti za izračunavanje f je polinomijalna,
- za svaki argument x je $|f(x)|$ najviše polinomijalno duže ili kraće od $|x|$ i
- vremenska granica složenosti za izračunavanje inverzne funkcije f^{-1} nije polinomijalna.

Primer 10.8.3 Jedan (nedokazani) kandidat za jednosmernu funkciju je množenje prostih brojeva koje zadovoljava prva tri uslova definicije, dok za sada nije pokazano ni da zadovoljava ni da ne zadovoljava i poslednji. Ξ

Primetimo da inverz f^{-1} funkcije f pripada klasi NP, jer se za svaki y i izabrani x u polinomijalnom vremenu proverava da li je $f(x) = y$. Očigledno je da, ako je $P = NP$ jednosmerne funkcije ne postoje. Pretpostavka $P \neq NP$, sa druge strane, dozvoljava postojanje ovakvih funkcija, mada još uvek ne garantuje da one zaista i postoje, pa ni konstrukciju sigurnih sistema sa tajnim ključem, kakav je sistem *DES*[38].

Primer 10.8.4 Jednosmerne funkcije se koriste i u sistemu šifri za pristup računarskom sistemu. Naime, šifre se u sistemu čuvaju u kodiranom obliku, pri čemu jednosmerna funkcija za kodiranje obezbeđuje sigurnost. Ξ

U sistemima sa javnim ključem postavljaju se dodatni zahtevi za jednosmerne funkcije kodiranja.

Definicija 10.8.5 Jednosmerna funkcija f je *tp-funkcija*[39] ako važi:

- efikasno se može odrediti domen funkcije f i
- postoji funkcija h sa polinomijalnom vremenskom granicom složenosti koja primenjena na ulazni podataka pojednostavljuje dekodiranje.

Primer 10.8.6 Primer sistema sa javnim ključem je *RSA*[40]. Sistem se zasniva na rezultatima teorije brojeva[41]. Na početku se biraju dva prosta broja p i q, recimo

[37] Public key

[38] Data Encryption Standard.

[39] Tp je skraćenica engleske reči trapdoor koja označava podrumska ili tavanska vrata, a smisao je ulazak na mala ili sporedna vrata, tj. zaobilaženje standardnog postupka.

[40] Naziv sistema dolazi od imena autora: Ron Ravist, Adi Shamir i Len Adleman.

[41] Koristi se posledica Male Fermaove teoreme da za proste brojeve p i q i broj a uzajamno prost sa $(p-1)(q-1)$ važi $a^{(p-1)(q-1)} = 1 (\mathrm{mod}\ pq)$.

$p = 47$ i $q = 71$, i razmatraju njihov proizvod, $n = 47 \times 71 = 3337$, i $(p-1)(q-1)$, $46 \times 70 = 3220$. U stvarnosti, binarni zapis broja n je reda veličine stotina cifara. Zatim se iz skupa $\{1, \ldots, (p-1)(q-1)\}$ bira broj e koji je uzajamno prost sa $(p-1)(q-1)$ i nalazi broj d takav da je $ed = 1(\mathrm{m}od\ (p-1)(q-1))$. Ovo poslednje je moguće pošto skup brojeva iz $\{1, \ldots, (p-1)(q-1)\}$ koji su uzajamno prosti sa $(p-1)(q-1)$ čini grupu u odnosu na množenje po modulu $(p-1)(q-1)$. U ovom primeru izaberimo $e = 79$ i $d = 79^{-1}(\mathrm{m}od\ 3220) = 1019$. Brojevi e i d će biti javni ključ za šifrovanje, odnosno tajni ključ za dešifrovanje. Preciznije, javni ključ je uređeni par (pq, e), dok je tajni ključ par (pq, d). Brojevi p i q treba da ostanu tajni. Funkcija kodiranja je

$$f_{RSA}(x) = x^e(\mathrm{m}od\ pq)$$

a funkcija dekodiranja

$$(f_{RSA}(x))^d(\mathrm{m}od\ pq) = x^{ed}(\mathrm{m}od\ pq) = x^{1+k(p-1)(q-1)}(\mathrm{m}od\ pq) = x(\mathrm{m}od\ pq) = x.$$

Ako da su p i q poznati, broj d se efikasno izračunava, pa se nalaženje inverzne funkcije od f_{RSA} svodi na problem faktorisanja brojeva. Pod pretpostavkom da je množenje prostih brojeva jednosmerna funkcija ovo se ne može izvesti efikasno i f_{RSA} jeste *tp*-funkcija. U praksi je poruka koja se šifruje binarni niz. Taj niz se deli na blokove koji svaki za sebe predstavlja ceo broj veći do jednak od 0, a manji od $n = pq$. Zato je svaki blok binarni niz nešto kraći od dužine binarnog zapisa broja n. Na primer, neka je 688232 poruka koju treba šifrovati prethodno određenim ključem. poruka se deli u dva bloka $x_1 = 688$ i $x_2 = 232$ koji se kodiraju sa $c_1 = x_1^e(\mathrm{m}od\ pq) = 688^{79}(\mathrm{m}od\ 3337) = 1570$ i $c_2 = 232^{79}(\mathrm{m}od\ 3337) = 2756$. Dekodiranje se obavlja sa $c_1^d(\mathrm{m}od\ pq) = 1570^{1019}(\mathrm{m}od\ 3337) = 688$ i $2756^{1019}(\mathrm{m}od\ 3337) = 232$. Ξ

Algoritam RSA se može koristiti i za takozvani digitalni potpis kojim se osoba B uverava da je osoba A zaista poslala poruku x. Neka A i B imaju redom javne ključeve e_A i e_B i tajne ključeve d_A i d_B i koriste RSA algoritam. Potpisana poruka koju šalje A je oblika $S_A(x) = (x, D(d_A, x))$ i sadrži originalnu poruku x na koju je nadovezan potpis, tj. vrednost $D(d_A, x)$. Pošto važi $x^{ed}\mathrm{m}od\ pq = x^{de}\mathrm{m}od\ pq$ postupci kodiranja i dekodiranja za RSA komutiraju, pa B proverava potpis računajući $E(e_A, D(d_A, x)) = D(d_A, E(e_A, x)) = x$. Ako je zbog sigurnosti potrebno i kodirati celu potpisanu poruku koristi se uobičajeni postupak nad ključevima e_B i d_B.

10.8.2 Protokoli

Razmena poruka između dve osobe, kao na slici 10.1, može biti izazvana različitim razlozima, recimo željom osobe A da u nešto ubedi osobu B, čak i ako to nije tačno. Ovakve situacije se u kriptologiji formalizuju *protokolima* koji se shvataju kao skupovi izračunavanja dvaju osoba koje u dijalogu sarađuju deleći ulazne i izlazne podatke.

Sistemi interaktivnih dokaza

Interaktivni dokaz[42] (A,B) je protokol u kome učestvuju osobe A i B tako da:

[42] Interactive proof.

- A može da vrši eksponencijalno složena izračunavanja, pri čemu osobi B nešto dokazuje,
- B raspolaže samo sredstvima čija složenost pripada klasi BPP,
- i A i B znaju ulazni podatak x,
- A i B razmenjuju poruke m_1, m_2, ..., $m_{2|x|^k}$, čije su dužine polinomijalno ograničene u odnosu na $|x|$, pri čemu A šalje neparne, a B parne poruke,
- poruke koje šalje A su funkcije ulaznog podatka x i svih prethodnih poruka, $m_{2l-1} = f(x, m_1, \ldots, m_{2l-2})$,
- svaka poruka koju šalje B je funkcija ulaznog podatka x, svih prethodnih poruka i jednog slučajnog broja izabranog za tu poruku koji A ne poznaje i predstavljenog dugim nizom bitova, $m_{2l} = f(x, m_1, \ldots, m_{2l-1}, r_l)$ i
- nakon $2|x|^k$ koraka poslednja poruka koju šalje B je 'da' ili 'ne' što redom znači da je B ubeđen, odnosno da nije, u vezi tvrđenja vezanog za ulazni podatak.

Sistemi interaktivnih dokaza su verovatnosni pandan klase složenosti NP, kao što su verovatnosne klase RP i BPP analogani klase složenosti P.

Primer 10.8.7 Primer za interaktivni dokaz je pokušaj osobe A da ubedi osobu B da je neka klasična iskazna formula α zadovoljiva. Koristeći svoje snažne resurse A može da pronađe interpetaciju koja zadovoljava formulu, dok B to može lako da proveri. Ali ako formula nije zadovoljiva, B može zahtevati da se pošalje dovoljan broj interpretacija da bi utvrdio (ne)zadovoljivost. Ξ

Definicija 10.8.8 Klasa složenosti izračunavanja IP sadrži sve probleme L za koje:

- za svaki $x \in L$, verovatnoća da interaktivni dokaz (A,B) prihvata x je barem $1 - \frac{1}{2^{|x|}}$ i
- za svaki $x \notin L$, verovatnoća da interaktivni dokaz (A',B), u kome je algoritam A zamenjen proizvoljnim algoritmom A' eksponencijalne složenosti, prihvati x je najviše $\frac{1}{2^{|x|}}$.

Primer 10.8.9 Problem neizomorfnosti grafova GNI je ilustracija interaktivnog dokaza. Pretpostavimo da su data dva grafa $G = (V, E)$ i $G' = (V, E')$ sa jednakim skupom čvorova. Grafovi G i G' su izomorfni ako postoji permutacija π skupa čvorova V tako da je $E' = \{(\pi(u), \pi(v)) : (u, v) \in E\}$. Za problem GNI još nije poznato da li je u klasi P ili je NP-kompletan, ali se pokazuje da pripada klasi IP.

Pretpostavimo da osoba A želi da osobi B dokaže da ulazni grafovi G i G' nisu izomorfni. Protokol (A,B) se opisuje na sledeći način. Neka je ulaz $x = (G, G')$. Osoba B šalje poruke tako što u i-tom koraku:

- definiše graf G_i tako što bira slučajan bit b_i i ako je $b_i = 1$, $G_i = G$, inače $G_i = G'$,

- generiše slučajnu permutaciju π_i i
- šalje poruku $m_{2i-1} = (G, \pi(G_i))$.

Osoba A proverava da li su grafovi izomorfni, pa ako jesu, šalje odgovor $m_{2i} = 1$, inače šalje odgovor $m_{2i} = 0$. Nakon $|x|$ poslatih poruka B prihvata da grafovi nisu izomorfni ako su jednaki vektori $(b_1, \ldots, b_{|x|})$ slučajnih bitova koje je B generisao i $(m_2, \ldots, m_{2|x|})$ poruka koje je slao A.

Ako grafovi G i G' nisu izomorfni, osoba B nekada šalje izomorfne, a nekada neizomorfne grafove, zavisno od vrednosti slučajnog bita. Osoba A zna da grafovi nisu izomorfni, jer poseduje dovoljno računarske snage da to utvrdi, pa će uvek odgovarati korektno, jer u ovom slučaju joj nije u interesu da pređe osobu B. Time dolazimo u situaciju da ako je bit $b_i = 1$, B šalje izomorfne grafove i A to potvrđuje odgovarajući sa 1 i slično, ako je bit $b_i = 0$, B šalje neizomorfne grafove, a A to potvrđuje odgovarajući sa 0. Prema tome, B prihvata ulaz ako se spomenuti vektori poklapaju.

Ako G i G' jesu izomorfni, bez obzira na izbor bita b_i, B uvek šalje poruku oblika $(G, \pi(G))$. Osoba A tako uvek dobija izomorfne grafove. Podsetimo se da A želi da osobu B ubedi da grafovi nisu izomorfni. To znači da A mora odgovarati tako da se ni jednom bit odgovora ne razlikuje od odgovarajućeg bita koji je izabrao B. Verovatnoća da A, bez obzira na izbor algoritma kojim se služi, $|x|$ puta pogodi vrednost slučajnih bitova je najviše $\frac{1}{2^{|x|}}$, kao što se i zahteva. Ξ

Lako se vidi da je klasa složenosti NP sadržana u klasi IP jer se dobija od nje pod pretpostavkom da IP ne koristi slučajnost u algoritmu za B. Slično, ako B ignoriše poruke koje šalje A, klasa IP se svodi na klasu BPP. Precizno mesto klase IP u hijerarhiji klasa složenosti određeno je sledećom teoremom.

Teorema 10.8.10 $IP = PSPACE$.

Na osnovu ove teoreme, za svaki problem $L \in PSPACE$ i primerak x osoba A može ubediti verovatnosni proveravač polinomijalne složenosti oličen u osobi B da je zaista $x \in L$ iako je konvencionalni dokaz eksponencijalno dugačak.

Sistemi sa nultim znanjem

Sistemi sa nultim znanjem[43] se primenjuju u situacijama kada osoba A želi da osobu B ubedi u nešto, recimo da poseduje rešenje nekog teškog problema, a pri tom ne želi da oda samo rešenje, jer predstavlja tajnu.

Primer 10.8.11 Pretpostavimo da A zna kako da sa 3 boje oboji čvorove velikog grafa $G = (V, E)$ pri čemu susedni čvorovi nisu iste boje. Problem 3 boje je NP-kompletan problem (i kao takav težak) tako da A želi da B sazna za postojanje rešenja. Istovremeno, A ne želi da saopšti samo rešenje jer se plaši da će neko drugi moći da ga zloupotrebi, tj. koristi a da osoba A za to ne dobija protivverednost.

Protokol (A,B) se opisuje na sledeći način. Tri boje kodirajmo dvobitnim rečima. Tada je rešenje koje poseduje A preslikavanje $h : V \to \{00, 11, 01\}$. Jedan korak komunikacije obuhvata tri faze. Najpre A izvršava:

[43] Zero knowledge systems.

- generiše slučajnu permutaciju skupa boja i $|V|$ četvorki parametara (p_i, q_i, d_i, e_i) koji određuju RSA algoritam, po jedan za svaki $i \in V$,
- za svaki čvor i, boju čvora nakon permutacije $b_i b'_i = \pi(h(i))$ i slučajno izabrane brojeve x_i i x'_i ne veće od $\frac{p_i q_i}{2}$, A izračunava verovatnosni kod

$$(y_i, y'_i) = ((2x_i + b_i)^{e_i} \mathrm{m}od\ p_i q_i, (2x'_i + b'_i)^{e_i} \mathrm{m}od\ p_i q_i)$$

- poruka koju A šalje sadrži javni deo RSA i kodirani deo $(e_i, p_i q_i, y_i, y'_i)$ za svaki čvor $i \in V$.

Zatim, osoba B slučajno bira i i j, čvorove povezane ivicom u E, i proverava da li su različite boje. Osoba A na to odgovara tajnim ključevima d_i i d_j kako bi B izračunao

$$(b_i, b'_i) = ((y_i^{d_i} \mathrm{m}od\ p_i q_i) \mathrm{m}od\ 2, (y_i'^{d_i} \mathrm{m}od\ p_i q_i) \mathrm{m}od\ 2)$$

i analogno za (b_j, b'_j). Konačno, B proverava da li su boje različite. Ovakvi koraci se ponavljaju $k|E|$ puta gde je k parametar sigurnosti protokola.

Ako A ima rešenje problema, boje susednih čvorova koje B analizira će uvek biti različite. Pri tome B ne saznaje ništa o samom rešenje jer jedino što vidi jesu neke slučajne vrednosti. Ako A nema rešenje postojaće bar dva susedna čvora obojena u istu boju, pa je verovatnoća daće ih B izabrati bar $\frac{1}{|E|}$. Nakon $k|E|$ koraka komunikacije verovatnoća da će B pronaći kontraprimer je bar $1 - (1 - \frac{1}{|E|})^{k|E|}$. Ξ

Primetimo da, pošto je problem bojenja grafa u 3 boje NP-kompletan, svaki problem koji je u NP ima dokaz sa nultim znanjem.

10.9 Drugi pristupi složenosti

10.9.1 Opisna složenost

U ovom poglavlju smo za procenu složenosti problema koristili mašinski zavistan pristup u kome smo se oslanjali na različite varijante Tjuringove mašine. *Opisna*, odnosno *deskriptivna složenost* je pristup u kome se klase složenosti opisuju varijantama klasične logike prvog i drugog reda, na način nezavistan od konkretnog mašinskog modela izračunavanja. Na primer, pokazuje se da se klasa NP sastoji upravo od onih problema koji se opisuju formulama logike drugog reda u kojoj su svi kvantifikatori drugog reda egzistencijalni.

10.9.2 Složenost konačnih objekata

Pristup složenosti koji je opisan u prethodnim odeljcima se bavi potencijalno beskonačnim jezicima, tako da se od njega ne može očekivati odgovor na pitanja oblika koja je od dve binarne reči 011010110111001 i 010101010101010 složenija, tj. sadrži više informacija. Pod količinom informacija koju neka reč sadrži podrazumeva se veličina opisa reči iz koga se ta reč može restaurirati. Opisi koji su značajno kraći od odgovarajućih reči ukazuju na to da se te reči mogu sabiti bez gubljenja informacija, pa da ni količina informacija koje sadrže nije velika.

U ovakvoj situaciji je pogodno koristiti teoriju složenosti koja se takođe naziva *opisna*, mada nema nikakve veze sa pristupom iz odeljka 10.9.1, a koju je zasnovao Kolmogorov (Андрей Николаевич Колмогоров 1903 – 1987) i u kojoj se analizira složenost pojedinačnih objekata, poput reči, gramatika, automata, zapisa programa itd. Zapravo, pošto se svi ovi objekti mogu prikazati kao reči binarnog alfabeta, dovoljno je razviti teoriju opisne složenosti binarnih reči. Složenost reči x na binarnom alfabetu, u oznaci $K(x)$, se definiše kao dužina najkraćeg opisa Tjuringove mašine M i njenog ulaza w za koje je $M(w) \downarrow x$. Pokazuje se da izbor kodiranja Tjuringovih mašina nije bitan, jer za neko fiksirano kodiranje μ_0 i bilo koje drugo kodiranje μ' postoji konstanta c takva da za svaku reč x važi $K_{\mu_0}(x) \leq K_{\mu'}(x) + c$.

Neke od osobina opisne složenosti su:

- Postoji konstanta c takva da je za svaku reč x, $K(x) \leq |x| + c$, tako da složenost neke reči ne može biti puno veća od dužine same te reči.
- Postoji konstanta c takva da je za svaku reč x, $K(xx) \leq K(x) + c$, tako da složenost reči koja sadrži pravilna ponavljanja podreči ne može biti puno veća od složenosti same te podreči.

Pokazuje se da postoje reči proizvoljne dužine za koje složenost $K(x)$ nije manja od dužine reči. Takve reči se ne mogu opisati kompaktnije nego što je sam zapis reči, pa se nazivaju i *reči koje se ne mogu sabiti*[44]. Reči koje se ne mogu sabiti poseduju mnoge osobine koje se očekuju od slučajnih reči. Recimo reči ove vrste imaju približno isti broj 0 i 1.

Realni problemi sa opisnom složenošću su to što funkcija $K(x)$ nije izračunljiva, a problem da li se reč može sabiti nije odlučiv.

Primetimo na kraju da se po prirodi stvari, složenost nekog izračunavanja izražava funkcijom, dok se opisna složenost nekog objekta izražava konstantom.

10.10 Zaključak

Razdvajanje problema na praktično izračunljive i izračunljive u principu, zavisno od toga jesu li, ili ne, u klasi P nije uvek opravdano. Recimo, algoritam sa eksponencijalnom vremenskom granicom složenosti u kojoj je eksponent mali je u nekim praktičnim slučajevima, u kojima ulaz relativno nije veliki, pogodniji od algoritma sa polinomijalnom vremenskom granicom složenosti. U tabeli 10.1 su prikazana dva slučaja. U svakom od redova je prikazana dužina rada računara koji u sekundi obavlja 10^9 koraka izračunavanja za po dva algoritma sa polinomijalnom, odnosno eksponencijalnom, vremenskom granicom složenosti. Razlika između redova je u stepenima polinoma, odnosno eksponetima i ilustruje relativnost ove vrste podele na praktično i samo u principu izračunljive probleme.

Slična situacija se javlja kod linearnog programiranja i simpleks algoritma koji je eksponencijalan, ali dobrih performansi u praksi, odnosno nekih polinomijalnih algoritama za ovaj problem koji su u praksi veoma spori. Teorijski gledano, broj slučajeva u kojima bi neki eksponencijalni algoritam mogao biti bolji od polinomijalnog je obavezano konačan, ali sa stanovišta praktičnog programiranja, primerci

[44] Incompressible.

Vremenska granica	Vreme izvršavanja	Vremenska granica	Vreme izvršavanja
n^2	$3.6 \cdot 10^{-6} sec$	2^n	36.3 godine
n^{10}	19.4 godine	$2^{\sqrt[4]{n}}$	$8 \cdot 10^{-9} sec$

Tabela 10.1. Poređenje vremena izvršavanja algoritama za ulazne veličine $n = 60$.

problema koji su interesantni mogu biti upravo u tom skupu. Ipak treba reći da polinomijalnih algoritama sa ogromnim stepenima nema puno, kao i eksponencijalnih algoritama sa jako malim eksponentom, pa spomenute sitacije nisu pravilo.

Druga primedba u vezi teorije složenosti izračunavanja se odnosi na to što se analiziraju slučajevi u kojima se algoritmi najgore ponašaju. Moguće je da se algoritam sa lošim najgorim slučajem prihvatljivo, pa čak i superiorno u odnosu na ostale, ponaša u proseku. Primer za to je *quick-sort* algoritam sortiranja koji za slučajan niz ima složenost $O(n \log_2 n)$, dok je složenost za najgori slučaj $O(n^2)$. Analiza očekivanog, a ne najgoreg, slučaja je u takvim situacijama mnogo informativnija. Međutim, da bi se ovakva analiza sprovela potrebno je poznavanje distibucije ulaznih problema, što je često teško ostvarljivo.

Polinomijalne funkcije su pogodne za analizu: njihova klasa je zatvorena za sabiranje i množenje, logartitmi svih polinomijalnih funkcija se razlikuju u konstantnom faktoru, pa su svi u $\theta(\log_2 n)$ itd. Zbog svega ovoga je izbor pristupu prihvatanja statusa praktične izračunljivosti za probleme koji su u najgorem slučaju polinomijalni nužno pojednostavljenje koje dovodi do primenljive i elegantne teorije koja govori o stvarnim izračunavanjima.

10.11 Za dalje proučavanje

Teorija složenosti izračunavanja je relativno mlada disciplina. Referentne tekstove za nju predstavljaju [48, 105, 118]. Poslednja rečenica u odeljku 10.10 je zapravo teza izneta u [105], gde se mogu pronaći i dokazi za teoreme koje se u ovom poglavlju samo formulisane. Jedan od novijih rezultata teorije složenosti pokazuje da je problem $PRIMES \in P$ [4]. Opisna složenost, kao logički pristup problemu data je u [58], dok se opisna složenost konačnih reči uvodi u [67]. Preplitanje slučajnosti, složenosti i izračunavanja kao osnov moderne kriptologije analizira se u [39]. Praktičan pristup kriptologiji sa prikazom odgovarajućih tehnika i algoritama u programskom jeziku C izlaže se u [115]. Na našem jeziku se u [130] analiziraju različiti algoritmi, njihove primene u kriptologiji i razmatraju NP-kompletni problemi.

11

Zadaci

11.1 Izračunljivost. Odlučivost

Zadatak 11.1. Napisati program za Tjuringovu mašinu u kome se za neko fiksirano $n > 0$ ispisuje n jedinica počev od tekuće pozicije na desno, posle čega se glava vraća na polaznu poziciju i završava rad.

Zadatak 11.2. Napisati program za Tjuringovu mašinu u kome se podrazumeva da traka sadrži samo jednu nepraznu reč sastavljenu od $n > 0$ jedinica, a glava se nalazi iznad ćelije u kojoj je najlevlji znak 1. Program briše tu reč i ispisuje reč koja sadrži $2n$ jedinica. *Uputstvo.* Najpre obrisati najlevlji znak 1 i ići na desni kraj reči. Zatim preći jedan blanko znak (0), pa ako se tada naiđe na blanko znak upisati dva znaka 1, a ako se naiđe na znak 1 ići do kraja te nove reči i tada upisati dva znaka 1. Potom vratitit glavu na najlevlji znak prve reči, ako takav postoji, posle čega se postupak ponavlja. Ako je prva reč već obrisana glava se pomera na početak druge (i sada jedine na traci) reči, nakon čega se prekida izvršavanje.

Zadatak 11.3. Napisati program za Tjuringovu mašinu koji pomera ulaznu reč za jedno mesto udesno.

Zadatak 11.4. Napisati program za Tjuringovu mašinu koji kopira ulaznu reč w na deo trake odvojen jednim blanko znakom od krajne desne ćelije koja sadrži znak 1 iz reči w.

Zadatak 11.5. Napisati program za Tjuringovu mašinu koji izračunava funkciju prethodnik definisanu sa

$$f(x) = \begin{cases} x - 1 & \text{za } x \in \mathbb{N} \setminus 0 \\ \text{nedefinisano} & \text{za } x = 0 \end{cases}$$

Uputstvo. Kako se glava nalazi nad najlevljom ćelijom koja sadrži znak 1 iz unarnog zapisa argumenta, najpre se sadržaj te ćelije obriše, a zatim se glava pomera u desno do prve ćelije koja sadrži znak 1. Ako je argument programa nula, onda se ulazi u beskonačnu petlju u kojoj se mašina stalno pomera u desno. Prema tome, za svaki $x > 0$, $P(x) \downarrow x - 1$, dok je $P(0) \uparrow$.

Zadatak 11.6. Napisati program za Tjuringovu mašinu koji izračunava funkciju $f(x, y) = x + y$. *Uputstvo.* Na početku rada unarne reprezentacije brojeva x i y su na traci razdvojene jednim blanko znakom. Glava se najpre pomera do kraja

zapisa broja x, zatim se umesto blanko znaka koji razdvaja zapise brojeva x i y upisuje znak 1. Kako se reprezentacija brojeva x i y sastoji od $x+1$ odnosno $y+1$ jedinica, sada je na traci $x+1+1+y+1$ jedinica. Zato se odlazi na kraj zapisa broja y i brišu dva znaka 1. Na traci se sada nalazi neprekidan niz od $x+y+1$ jedinica, što je unarna reprezentacija broja $x+y$, pa ostaje da se samo još glava vrati na najlevlju ćeliju koja sadrži znak 1. Dakle, $P(x,y) \downarrow x+y$.

Zadatak 11.7. Napisati program za Tjuringovu mašinu koji izračunava funkciju $f(x,y) = x \cdot y$. *Uputstvo.* Ulazni podaci se na traci smeštaju kao $1^{x+1}01^{y+1}$. Reč 1^{x+1} služi kao brojač koliko puta treba iskopirati drugu reč. Voditi računa da se unarna reprezentacija broja $x \cdot y$ sastoji od $x \cdot y + 1$ jedinice.

Zadatak 11.8. Napisati program za Tjuringovu mašinu koji izračunava funkciju

$$f(x,y) = \begin{cases} \text{najveći broj } \leq \frac{x}{y} & \text{za } y > 0 \\ \text{nedefinisano} & \text{za } y = 0 \end{cases}$$

Zadatak 11.9. Ispitati da li je funkcija

$$jednako(x,y) = \begin{cases} 1 & \text{za } x = y \\ 0 & \text{za } x \neq y \end{cases}$$

Tjuring-izračunljiva.

Zadatak 11.10. Napisati program za Tjuringovu mašinu koji izračunava funkciju $f(x,y) = x^y$.

Zadatak 11.11. Napisati program za Tjuringovu mašinu koji za svako k i i za koje je $k \geq i \geq 1$ izračunava funkciju $f(k,i,x_1,x_2,\ldots,x_k) = x_i$, tj. projekciju i-te koordinate.

Zadatak 11.12. Napisati program koji kao ulazne podatke dobija opis ulaza i program za Tjuringovu mašinu i simulira izvršavanje programa na različitim verzijama mašine.

Zadatak 11.13. Dokazati da je funkcija množenja $f(n,x) = n \cdot x$ primitivno rekurzivna. *Uputstvo.* Funkcija je definisana sa $f(0,x) = 0$ i $f(n+1,x) = h(f(n,x),n,x)$, gde je $h(x,y,z) = +(P_3^1, P_3^3)$, a $+$ funkcija sabiranja.

Zadatak 11.14. Dokazati da je funkcija stepenovanja $f(n,x) = x^n$ primitivno rekurzivna. *Uputstvo.* Funkcija je definisana sa $f(0,x) = 1$ i $f(n+1,x) = h(f(n,x),n,x)$, gde je $h(x,y,z) = *(P_3^1(x,y,z), P_3^3(x,y,z))$, a $*$ funkcija množenja.

Zadatak 11.15. Dokazati da je funkcija *signum* definisana sa:

$$sgn(n) = \begin{cases} 0 & \text{za } n = 0 \\ 1 & \text{za } n > 0 \end{cases}$$

primitivno rekurzivna. *Uputstvo.* Funkcija je definisana sa $sgn(0) = 0$ i $sgn(n+1) = K_2^1(sgn(n), n)$, gde je $K_2^1(x,y) = 1$ konstantna funkcija za koju je u primeru 2.4.2 pokazano da je primitivno rekurzivna.

Zadatak 11.16. Dokazati da je funkcija definisana sa:

$$\overline{sgn}(n) = \begin{cases} 1 & \text{za } n = 0 \\ 0 & \text{za } n > 0 \end{cases}$$

primitivno rekurzivna. *Uputstvo.* Funkcija je definisana sa $\overline{sgn}(0) = 1$ i $\overline{sgn}(n+1) = K_2^0(\overline{sgn}(n), n)$, gde je $K_2^0(x, y) = 0$ konstantna funkcija za koju je u primeru 2.4.2 pokazano da je primitivno rekurzivna.

Zadatak 11.17. Dokazati da je funkcija definisana sa

$$\text{n}eparno(n) = \begin{cases} 1 & \text{za } n \text{ neparno} \\ 0 & \text{za } n \text{ parno} \end{cases}$$

primitivno rekurzivna. *Uputstvo.* Funkcija je definisana sa n$eparno(0) = 0$ i n$eparno(n + 1) = h(\text{n}eparno(n), n)$, gde je $h(x, y) = 1 - P_2^1(x, y)$ primitivno rekurzivna funkcija. Drugi način je definisati funkciju n*eparno* kao karakterističnu funkciju istoimenog predikata iz primera 2.4.21.

Zadatak 11.18. Dokazati da je funkcija definisana sa

$$\text{p}arno(n) = \begin{cases} 0 & \text{za } n \text{ neparno} \\ 1 & \text{za } n \text{ parno} \end{cases}$$

primitivno rekurzivna. *Uputstvo.* Funkcija se dobija kompozicijom primitivno rekurzivnih funkcija, jer je p$arno(n) = \overline{sgn}(\text{n}eparno(n))$. Drugi način je definisati funkciju p*arno* kao karakterističnu funkciju istoimenog predikata iz primera 2.4.21.

Zadatak 11.19. Dokazati da je funkcija definisana sa

$$\text{p}olovina(n) = \begin{cases} \frac{n}{2} & \text{za } n \text{ parno} \\ \frac{n-1}{2} & \text{za } n \text{ neparno} \end{cases}$$

primitivno rekurzivna. *Uputstvo.* Funkcija je definisana sa p$olovina(0) = 0$ i p$olovina(n + 1) = h(\text{p}olovina(n), n)$, gde je $h(x, y) = +(P_2^1(x, y), \text{n}eparno(P_2^2(x, y)))$ primitivno rekurzivna funkcija.

Zadatak 11.20. Dokazati da je funkcija faktorijel definisana sa

$$!(n) = \begin{cases} 1 & \text{za } n = 0 \\ n \cdot (n-1) \cdots 1 & \text{za } n \neq 0 \end{cases}$$

primitivno rekurzivna. *Uputstvo.* Funkcija je definisana primitivnom rekurzijom: $!(0) = 1$ i $!(n + 1) = h(!(n), n)$, gde je $h(x, y) = *(P_2^1(x, y), +(1, P_2^2(x, y)))$ primitivno rekurzivna funkcija.

Zadatak 11.21. Dokazati da je funkcija $f(n) = 0 + 1 + \cdots + n$ primitivno rekurzivna. *Uputstvo.* Funkcija je definisana primitivnom rekurzijom: $f(0) = 0$ i $f(n + 1) = h(f(n), n)$, gde je $h(x, y) = +(P_2^1(x, y), S(P_2^2(x, y)))$, primitivno rekurzivna funkcija.

Zadatak 11.22. Dokazati da su predikati $\neq$, $\leq$, $>$ i $\geq$ primitivno rekurzivni. *Uputstvo.* Predikati $=$ i $<$ su primitivno rekurzivni, a za karakteristične funkcije važi $C_{\neq} = C_{\neg =}$, $C_{\leq} = C_{=} \vee C_{<}$, $C_{>} = C_{\neg \leq}$ i $C_{\geq} = C_{=\vee >}$.

Zadatak 11.23. Dokazati da je funkcija $\max\{x_1, \ldots, x_n\}$ koja daje maksimum konačnog skupa prirodnih brojeva primitivno rekurzivna. *Uputstvo.* Za $n = 2$ funkcija je definisana sa $\max\{x_1, x_2\} = x_1 \cdot C_{\geq}(x_1, x_2) + x_2 \cdot C_{<}(x_1, x_2)$ i predstavlja kompoziciju primitivno rekurzivnih funkcija. Ako je definisana funkcija maksimuma za $n = k$ brojeva, maksimum $k+1$ brojeva se definiše sa $\max\{x_1, \ldots, x_{k+1}\} =$

$\max\{x_1,\ldots,x_k\}\cdot C_{\geq}(\max\{x_1,\ldots,x_k\},x_{k+1})+x_{k+1}\cdot C_{\geq}(\max\{x_1,\ldots,x_k\},x_{k+1})$.

Zadatak 11.24. Dokazati da je funkcija $\min\{x_1,\ldots,x_n\}$ koja daje minimum konačnog skupa prirodnih brojeva primitivno rekurzivna. *Uputstvo.* Primeniti analogan postupak kao u zadatku 11.23 sa funkcijom max.

Zadatak 11.25. Dokazati da je funkcija $f(x,y)$ koja izračunava najmanji zajednički sadržalac brojeva x i y primitivno rekurzivna. *Uputstvo.* $f(x,y)=(\mu z\leq x\cdot y)(\div(x,z)\wedge\div(y,z))$.

Zadatak 11.26. Dokazati da je funkcija $f(x)$ koja izračunava celi deo kvadratnog korena broja x primitivno rekurzivna. *Uputstvo.* $f(x)=(\mu y\leq x)((y+1)^2>x)$.

Zadatak 11.27. Dokazati da je funkcija $f(n)$ koja izračunava n-ti član Fibonačijevog niza ($f(0)=1$, $f(1)=1$, $f(n+2)=f(n)+f(n+1)$) primitivno rekurzivna. *Uputstvo.* Posmatrajmo funkciju $g(n)=2^{f(n)}\cdot 3^{f(n+1)}$ za koju je $(g(n))_0=f(n)$, $(g(n))_1=f(n+1)$. Za funkciju važi da je $g(0)=2^1\cdot 3^1=6$ i $g(n+1)=2^{f(n+1)}\cdot 3^{f(n+2)}=2^{f(n+1)}\cdot 3^{f(n)+f(n+1)}=6^{f(n+1)}\cdot 3^{f(n)}=6^{(g(n))_1}\cdot 3^{(g(n))_0}$, pa je g primitivno rekurzivna, a time i f.

Zadatak 11.28. Dokazati da su primitivno rekurzivne funkcije f i g uzajamno definisane sa: $f(0)=a$, $g(0)=b$, $f(n+1)=h(n,f(n),g(n))$ i $g(n+1)=t(n,f(n),g(n))$, gde su h i t primitivno rekurzivne funkcije. *Uputstvo.* Posmatrajmo funkciju $i(n)=2^{f(n)}\cdot 3^{g(n)}$. Razmotriti i situaciju u kojoj su funkcije f i g arnosti $n+1$: $f(0,x_1,\ldots,x_n)=k(x_1,\ldots,x_n)$, $g(0,x_1,\ldots,x_n)=q(x_1,\ldots,x_n)$, $f(n+1,x_1,\ldots,x_n)=h(n,x_1,\ldots,x_n,f(n,x_1,\ldots,x_n),g(n,x_1,\ldots,x_n))$ i $g(n+1,x_1,\ldots,x_n)=t(n,x_1,\ldots,x_n,f(n,x_1,\ldots,x_n),g(n,x_1,\ldots,x_n))$.

Zadatak 11.29. Dokazati da je Akermanova funkcija dobro definisana, tj. da postoji funkcija $f:\mathbb{N}^3\to\mathbb{N}$ koja zadovoljava uslove zadate u odeljku 2.4.6. *Uputstvo.* Videti [82, str. 63].

Zadatak 11.30. Dokazati da postoji parcijalno rekurzivna funkcija $g(x,y)$ koja je definisana ako i samo ako je $y\in Im(f_x)$. *Uputstvo.* Definišimo funkciju $g(x,y)$ tako da bude jednaka $((\mu z)UP(x,(z)_0,y,(z)_1))_1$. Funkcija $g(x,y)$ je definisana ako i samo ako funkcija f_x za neki argument staje, a izračunata je vrednost y.

Zadatak 11.31. Dokazati da je inverzna funkcija f^{-1} parcijalno rekurzivne $1-1$ funkcije f parcijalno rekurzivna. *Uputstvo.* Za neki e je $f\simeq f_e$. Definišimo $f^{-1}(y)=((\mu z)UP(e,(z)_0,y,(z)_1))_1$.

Zadatak 11.32. Neka je $f(x,y)=y^x$. Pokazati da postoji rekurzivna funkcija $K(x)$ za koju je $f_{K(x)}(y)\simeq f(x,y)$. *Uputstvo.* Pošto je y^x parcijalno rekurzivna funkcija postoji indeks e tako da je $f(x,y)=f_e(x,y)$, odnosno po s-m-n-teoremi $f(x,y)\simeq f_{S_1^1(e,x)}(y)$. Indeks e je konstantan, pa je funkcija S_1^1 u stvari unarna funkcija koju nazivamo $K(x)$.

Zadatak 11.33. Pokazati da postoji rekurzivna funkcija $K(x)$ takva da je za svaki $n\in\mathbb{N}$ $K(n)$ jedan indeks funkcije $f(x,y)=\left[\sqrt[x]{y}\right]$ koja izračunava celi deo n-tog korena broja. *Uputstvo.* Pošto je $\left[\sqrt[x]{y}\right]$ parcijalno rekurzivna funkcija postoji indeks e tako da je $f(x,y)=f_e(x,y)$, odnosno po s-m-n-teoremi $f(x,y)\simeq f_{S_1^1(e,x)}(y)$. Indeks e je konstantan, pa je funkcija S_1^1 u stvari unarna funkcija koju nazivamo $K(x)$.

Zadatak 11.34. Dokazati da postoji rekurzivna funkcija $K(x,y)$ takva da je $f_x(z)\cdot f_y(z)\simeq f_{K(x,y)}(z)$. *Uputstvo.* Najpre je $f_x(z)\simeq f_U(x,z)$ i $f_y(z)\simeq f_U(y,z)$. Funkcija $f_U(x,z)\cdot f_U(y,z)=f_e(x,y,z)$ za neki indeks e, pa je po s-m-n-teoremi

$f_x(z) \cdot f_y(z) \simeq f_{S_1^2(e,x,y)}(z)$. Indeks e je konstantan, pa je funkcija S_1^2 u stvari binarna funkcija koju nazivamo $K(x,y)$.

Zadatak 11.35. Dokazati da postoji unarna rekurzivna funkcija $K(x)$ takva da $Dom(f_{K(x)}) = Im(f_x)$. *Uputstvo.* Neka je $g(x,y) = ((\mu z)UP(x,(z)_0,y,(z)_1)$. Kada je funkcija definisana važi $f_x(g(x,y)) = y$. Pošto je g parcijalno rekurzivna funkcija postoji indeks e tako da je $g(x,y) = f_e(x,y)$, odnosno po s-m-n-teoremi $g(x,y) \simeq f_{S_1^1(e,x)}(y)$. Indeks e je konstantan, pa je funkcija S_1^1 u stvari unarna funkcija koju nazivamo $K(x)$. Dobija se $Im(f_x) = Dom(f_{K(x)})$.

Zadatak 11.36. Dokazati da je klasa odlučivih skupova zatvorena za operacije komplement, unije, preseka i razlike. *Uputstvo.* Za skupove A i B karakteristične funkcije su redom $C_{\complement A} = 1 - C_A$, $C_{A\cup B} = sgn(C_A + C_B)$, $C_{A\cap B} = C_A \cdot C_B$, $C_{A\setminus B} = \dot{-}(C_A, C_B)$.

Zadatak 11.37. Neka je $c \in \mathbb{N}$ konstanta. Dokazati da predikat $P(x)$ koji važi ako je $c \in Dom(f_x)$ nije odlučiv. *Uputstvo.* Skup $\mathbb{B} = \{f_x : c \in Dom(f_x)\}$ nije ni prazan, a ne sadrži ni sve parcijalno rekurzivne funkcije, pa po teoremi 2.8.5 problem da li je $f_x \in \mathbb{B}$ nije odlučiv zbog čega ni $P(x)$ nije odlučiv.

Zadatak 11.38. Dokazati da predikat $P(x)$ koji važi ako je $x \in Im(f_x)$ nije odlučiv. *Uputstvo.* Neka je karakteristična funkcija C_P rekurzivna. Tada je i funkcija

$$h(x) = \begin{cases} x & \text{za } C_P(x) = 0 \\ \uparrow & \text{za } C_P(x) = 1 \end{cases}$$

parcijalno rekurzivna, pa postoji indeks m tako da je $h = f_m$. Da li je $m \in Im(f_m)$? Pokazati da se dobija kontradikcija. Zato C_P nije rekurzivna, pa ni P odlučiv.

Zadatak 11.39. Dokazati da predikat $P(x,y)$ koji važi ako je $x \in Im(f_y)$ nije odlučiv. *Uputstvo.* Redukovati na prethodni zadatak.

Zadatak 11.40. Dokazati da predikat $P(x,y)$ koji važi ako je $Dom(f_x) = Dom(f_y)$ nije odlučiv. *Uputstvo.* Redukovati na problem f_x je totalna funkcija koji nije odlučiv, a bio bi da jeste $P(x,y)$.

Zadatak 11.41. Dokazati da predikat $P(x)$ koji važi ako je $Dom(f_x) = \emptyset$ nije odlučiv. *Uputstvo.* Skup $\mathbb{B}$ koji sadrži sve funkcije praznog domena nije ni prazan niti sadrži sve parcijalno rekurzivne funkcije, pa po teoremi 2.8.5 problem da li je $f_x \in \mathbb{B}$ nije odlučiv zbog čega ni $P(x)$ nije odlučiv.

Zadatak 11.42. Dokazati da predikat $P(x)$ koji važi ako je $Im(f_x)$ beskonačan nije odlučiv. *Uputstvo.* Skup $\mathbb{B}$ koji sadrži sve funkcije sa osobinom da im je kodomen beskonačan nije ni prazan niti sadrži sve parcijalno rekurzivne funkcije, pa po teoremi 2.8.5 problem da li je $f_x \in \mathbb{B}$ nije odlučiv zbog čega ni $P(x)$ nije odlučiv.

Zadatak 11.43. Dokazati da ne postoji rekurzivna funkcija binarna funkcija f koja konvergira ako funkcija $f_x(y) \downarrow$ a u računanju se koriste sam brojevi manji do jednaki $f(x,y)$.[1] *Uputstvo.* Predikat $T(x,y,z)$ definisan analogno predikatu UP sem što ne vodi brigu o rezultatu funkcije $f_x(y)$ je primitivno rekurzivan. Kako je $y \in Dom(f_x)$ ako i samo ako važi $T(x,y,f(x,y))$, a $y \in Dom(f_x)$ nije odlučivo,

[1] U terminologiji programa ovaj zadatak se može formulisati i ovako: Dokazati da ne postoji rekurzivna funkcija binarna funkcija f koja konvergira ako predikat $P_x(y) \downarrow$ u manje do jednako $f(x,y)$ koraka. (U oznaci P_x x je indeks programa u nekom nabrajanju.)

funkcija f nije rekurzivna.

Zadatak 11.44. Dokazati da je klasa parcijalno odlučivih predikata zatvorena za konjunkciju, disjunkciju, egzistencijalnu kvantifikaciju (ako je $P(x,y)$ parcijalno odlučiv, to je i $(\exists y)P(x,y)$) i ograničene univerzalnu i egzistencijalnu kvantifikaciju (ako je $P(x,y)$ parcijalno odlučiv, to su i $(\forall y < z)P(x,y)$ i $(\exists y < z)P(x,y)$), a nije za negaciju i univerzalnu kvantifikaciju. *Uputstvo.* Zatvorenost za konjunkciju i disjunkciju se pokazuje konstrukcijom karakterističnih funkcija. Predikat $P(x,y)$ je parcijalno odlučiv ako postoji odlučiv predikat $R(x,y,z)$ takav da je $M(x,y)$ ako i samo ako je $(\exists z)R(x,y,z)$. Posmatramo odlučivi predikat $S(x,y)$ koji važi ako i samo ako važi $R(x,(y)_0,(y)_1)$. Sada $(\exists y)M(x,y)$ važi ako i samo ako važi $(\exists y)S(x,y)$, pa je $(\exists y)M(x,y)$ parcijalno odlučiv. Karakteristična funkcija za predikat $(\forall y < z)P(x,y)$ je $C_{(\forall y<z)P(x,y)} = \Pi_{y<z}C_P(x,y)$. Predikat $Halt(x,y)$ je parcijalno odlučiv, a nije odlučiv, pa njegov komplement nije parcijalno odlučiv. Zbog toga ova klasa nije zatvorena za negaciju. Pošto je klasa zatvorena za egzistencijalnu kvantifikaciju, a nije za negaciju, nije zatvorena ni za univerzalnu kvantifikaciju jer je $\forall = \neg\exists\neg$.

Zadatak 11.45. Dokazati da je predikat $P(x,y)$ koji važi ako je $x \in Im(f_y)$ parcijalno odlučiv. *Uputstvo.* Primetimo da $x \in Im(f_y)$ važi ako i samo ako je $(\exists w)UP(y,(w)_0,(w)_1,(w)_2)$, a predikat UP je primitivno rekurzivan.

Zadatak 11.46. Dokazati da je predikat $P(x)$ koji važi ako je $Im(f_x) \neq \emptyset$ parcijalno odlučiv. *Uputstvo.* Skup $Im(f_x)$ je neprazan ako i samo ako važi $(\exists w)UP(x,(w)_0,(w)_1,(w)_2)$, a predikat UP je primitivno rekurzivan.

Zadatak 11.47. Dokazati da je predikat $P(x)$ koji važi ako je $Dom(f_x) \neq \emptyset$ parcijalno odlučiv. *Uputstvo.* Skup $Dom(f_x)$ je neprazan ako i samo ako važi $(\exists w)UP(x,(w)_0,(w)_1,(w)_2)$, a predikat UP je primitivno rekurzivan.

11.2 Matematička logika i Bulove algebre

Zadatak 11.48. Da li je zadovoljiv skup formula $\{A_1 \vee A_2, \neg A_2 \vee \neg A_3, A_3 \vee A_4, \neg A_4 \vee \neg A_5, \ldots\}$?

Zadatak 11.49. Metodom istinitosnih tablica ispitati da li su formule date u tabeli 3.4 tautologije.

Zadatak 11.50. Ispitati da li su formule $(A \to (B \to A))$, $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ i $(\neg B \to \neg A) \to (A \to B)$ zadovoljive, tautologije ili kontradikcije.

Zadatak 11.51. Izračunati $KNF(A)$ i $DNF(A)$ za formule: $(A \to B) \vee (\neg A \wedge C)$, $A \leftrightarrow (B \wedge \neg A)$, $((A \to B) \wedge C) \vee (\neg A \leftrightarrow C)$.

Zadatak 11.52. Napisati program koji polazeći od iskazne formule A konstruiše $KNF(A)$, $DNF(A)$ i definicionu formu formule A..

Zadatak 11.53. Dokazati da potpune forme dobijene upotrebom istinitosne tablice formule, kao što je opisano u odeljku 3.3.2, predstavljaju konjunktivnu, odnosno disjunktivnu, normalnu formu, tj. da su ekvivalentne polaznoj formuli. Dokazati da, ako polazna formula sadrži n iskaznih slova, važi da je formula tautologija (kontradikcija) ako i samo ako njena potpuna disjunktivna (konjunktivna) normalna forma sadrži 2^n disjunkata (konjunkata). *Uputstvo.* Razmatrati vrednost

formula i normalnih formi pri proizvoljnoj interpretaciji.

Zadatak 11.54. Dokazati jedinstvenost potpunih formi. *Uputstvo.* Posmatrajmo potpunu disjunktivnu normalnu formu formule A. Svaka interpretacija I za koju je $I \models A$, zadovaljava tačno jedan disjunkt u disjunktivnoj normalnoj formi i obrnuto svaka interpretacija koja zadovoljava potpunu disjunktivnu formu zadovoljava tačno jedan njen disjunkt i formulu A. Ovo bi dovelo do kontradikcije ako bi postojale dve potpune disjunktivne forme koje ne bi imale iste disjunkte.

Zadatak 11.55. Pokazati da se svaka konjunktivna, odnosno disjunktivna, normalna forma može ekvivalentno proširiti do potpune forme. *Uputstvo.* Neka formula sadrži iskazna slova p i q i neka je jedan disjunkt u njenoj konjunktivnoj formuli p. Kako je $\models p \leftrightarrow (p \vee (q \wedge \neg q))$ primenom distibutivnog zakona se vrši traženo kompletiranje $\models p \leftrightarrow ((p \vee q) \wedge (p \vee \neg q))$.

Zadatak 11.56. Neka su p, q i r iskazna slova i neka je formula A definisana sa $A = B \wedge r \wedge C$. Formule B i C su redom definisane sa $B = p \vee \neg D$ i $C = E \vee F$, dok su $D = p \vee q \vee \neg r$, $E = r \wedge D$ i $F = q \wedge r$. Optimizovati formulu A. *Uputstvo.* $A = (p \vee \neg(p \vee q \vee \neg r)) \wedge r \wedge ((q \wedge (p \vee q \vee \neg r)) \vee (q \wedge r)) = (p \vee (\neg p \wedge \neg q \wedge r)) \wedge r \wedge (q \wedge (p \vee q \vee r \vee \neg r)) = (p \vee (\neg p \wedge \neg q \wedge r)) \wedge r \wedge (q \wedge (p \vee q \vee \top)) = (p \vee (\neg p \wedge \neg q \wedge r)) \wedge r \wedge q = (p \wedge r \wedge q) \vee (\neg p \wedge \neg q \wedge r \wedge r \wedge q) = (p \wedge r \wedge q) \vee \bot = (p \wedge r \wedge q)$.

Zadatak 11.57. Pronaći iskaznu formulu A takvu da je svaka $KNF(A)$ eksponencijalno duža od A.

Zadatak 11.58. Konstruisati normalne forme za elektronsko kolo polusabirača koji ima dva ulaza x i y, gde x i y odgovaraju bitovima koji se sabiraju, i dva izlaza - r, bit rezultata, i s, bit prenosa. *Uputstvo.* Koristiti primer 3.3.13.

Zadatak 11.59. Dokazati da za svaku formulu A postoji formula B u konjunktivnoj normalnoj formi takva da svaki disjunkt sadrži tačno tri literala pri čemu je formula A zadovoljiva ako i samo ako je zadovoljiva formula B. Takođe, veličina formule B je ograničena linearnom funkcijom dužine formule A. *Uputstvo.* Pretpostavimo da je A u konjunktivnoj normalnoj formi, jer se A uvek može prevesti u definicionu formu koja je zadovoljiva ako i samo ako je zadovoljiva i A. Kao što je spomenuto u uputstvu za zadatak 11.55, svaki konjunkt iz A koji sadrži manje od tri iskazna literala se može proširiti tako da sadrži tačno tri literala. Neka je $D = B_1 \vee B_2 \vee \ldots \vee B_k$, $k \geq 4$, konjunkt iz A koji ima bar četiri literala i neka su $C_1, \ldots, C_{k-3}$ iskazna slova koja se ne javljaju u A. Konjunkt D se transformiše u formulu D' oblika

$$(B_1 \vee B_2 \vee C_1) \wedge (B_3 \vee \neg C_1 \vee C_2) \wedge \ldots \wedge (B_{k-2} \vee \neg C_{k-4} \vee C_{k-3}) \wedge (B_{k-1} \vee B_k \vee \neg C_{k-3})$$

koja je u konjunktivnoj normalnoj formi. Pokazuje se da je konjunkt D zadovoljiv ako i samo ako je zadovoljiva i formula D'. Neka je, recimo, $k = 4$ i neka je konjunkt $D = B_1 \vee B_2 \vee B_3 \vee B_4$ zadovoljiv. To znači da postoji interpretacija I koja zadovoljava neki literal. Neka to bude literal B_1. Tada posmatramo interpretaciju I' koja se poklapa sa I na literalima B_i i za koju je $I'(C_1) = \bot$. Lako se proverava da interpretacija I' zadovoljiva formulu $D' = (B_1 \vee B_2 \vee C_1) \wedge (B_3 \vee B_4 \vee \neg C_1)$. Slično, ako neka interpretacija I zadovoljava formulu D', ona zadovoljava ili iskazno slovo C_1 ili njegovu negaciju. Neka je $I(C_1) = \top$. Pošto je $I(\neg C_1) = \bot$, mora biti $I(B_3 \vee B_4) = \top$, pa je i $I(D) = \top$. Ovakvo razmatranje sa lako uopštava za proizvoljno $k > 4$. Postupak se ponavlja dok god u A ima konjunkata sa više od

tri literala.

Zadatak 11.60. Opisati skup teorema formalne teorije čiji jezik je skup $\{0,1\}$, skup formula se sastoji od svih reči, skup aksioma sadrži samo formulu 1, a pravila izvođenja su: 'iz B izvesti $B0$' i 'iz B izvesti $B1$'. *Uputstvo.* Formula je teorema ako i samo ako je čini konačni niz koji počinje znakom 1.

Zadatak 11.61. Opisati formalnu teoriju kod koje važi da je neka formula teorema ako i samo ako je čini konačni niz sastavljen od parnog broja jedinica.

Zadatak 11.62. Dokazati da su formule date u tabeli 3.4 teoreme iskaznog računa L.

Zadatak 11.63. Odrediti $Res^*(F)$ skupa $F = \{\{A_1, \neg A_2, A_5\}, \{\neg A_1, A_2, A_3\}, \{\neg A_1, \neg A_4, \neg A_5\}, \{A_1, A_4\}\}$.

Zadatak 11.64. Metodom rezolucije ispitati da li su formule date u tabeli 3.4 tautologije.

Zadatak 11.65. Pokazati da je realizacija metode rezolucije za iskaznu logiku opisana u odeljku 3.7.1 kompletna.

Zadatak 11.66. Isprogramirati realizaciju metode rezolucije za iskaznu logiku koja je opisana u odeljku 3.7.1.

Zadatak 11.67. Metodom analitičkog tabloa ispitati da li su formule date u tabeli 3.4 tautologije.

Zadatak 11.68. Isprogramirati realizaciju metode tabloa za iskaznu logiku koja je opisana u odeljku 3.8.4.

Zadatak 11.69. Izračunati kanonsku disjunktivnu formu za formulu $x+y$. *Uputstvo.* $x+y = ((0+0)\cdot x'\cdot y') + ((0+1)\cdot x'\cdot y) + ((1+0)\cdot x\cdot y') + ((1+1)\cdot x\cdot y) = (x'\cdot y) + (x\cdot y') + (x\cdot y)$.

Zadatak 11.70. Konstruisati BDD i redukovani BDD za funkcije: $f(x) = x$, $f(x) = x'$, $f(x,y) = x\cdot y$, $f(x,y) = x+y$ i proveriti da li vrednosti funkcija tih BDD-reprezentacija odgovaraju polaznim funkcijama.

Zadatak 11.71. Konstruisati BDD za formulu $f(a,b,c,d,e,f) = a\cdot b + c\cdot d + e\cdot f$ u odnosu na redoslede promenljivih: $a < b < c < d < e < f$, odnosno $a < c < e < b < d < f$.

Zadatak 11.72. Dokazati da postupak opisan u odeljku 4.3.2 zaista daje redukovani BDD.

Zadatak 11.73. Napisati program koji za zadati redosled promenljivih konstruiše BDD za ulaznu fomulu.

Zadatak 11.74. Na predikatskom jeziku prvog reda zapisati iskaz 'Svaki broj je paran ili neparan'.

Zadatak 11.75. Opisati predikatski jezik, interpretaciju i formule kojima se modelira struktura podataka red[2] koja se koristi po principu FIFO[3], tj. 'prvi stavljen, prvi uzet'. *Uputstvo.* Slično kao u primeru 5.3.13 definisati formalne simbole za prazan red, početak i ostatak reda i stavljanje novog objekta u red, interpretaciju, formulu sa jednom slobodnom promenljivom koja važi samo za reči dužine jedan i formulu koja modelira postupak FIFO. U poslednjoj formuli razlikovati situacije stavljanja objekta u prazan i neprazan red.

Zadatak 11.76. Dokazati da je formula $(\forall x)(A \to B) \to (A \to (\forall x)B)$ valjana ako promenljiva x nije slobodna u A, a da u suprotnom to nije slučaj. *Uput-*

[2] *Queue.*

[3] Skraćenicu čine prva slova fraze *First in first out.*

stvo. Ako formula nije valjana, postoje interpretacija I i valuacija v takve da je $I((\forall x)(A \to B))_v = \top$, $I(A)_v = \top$ i $I((\forall x)B))_v = \bot$. Znači da postoji neki c, element domena interpretacije I, takav da za neku valuaciju $v' \equiv_x v$ koja je jednaka sa v, sem što $v'(x) = c$, važi $I(B)_{v'} = \bot$. Pošto A ne sadrži slobodnu promenljivu x, $I(A)_v = I(A)_{v'} = \top$, pa bi suprotno pretpostavci bilo $I(A \to B)_{v'} = \bot$, odnosno $I((\forall x)(A \to B))_v = \bot$.

Zadatak 11.77. Da li je unifikabilan skup literala $\{P(x, y), P(f(a), g(x)), P(f(x), g(f(z)))\}$?

Zadatak 11.78. Koliko dugo radi procedura Unifikacija iz odeljka 5.11.1 u slučaju skupa literala $\{P(x_1, \ldots, x_n), P(f(x_0, x_0), f(x_1, x_1), \ldots, f(x_{n-1}, x_{n-1}))\}$?

Zadatak 11.79. Naći sve rezolvente klauza $C_1 = \{\neg P(x, y), \neg P(f(a), g(z, b)), Q(x, z)\}$ i $C_2 = \{P(f(x), g(a, b)), \neg Q(f(a), b), \neg Q(a, b)\}$.

Zadatak 11.80. Ispitati valjanost formula $(\exists x)(\forall y)P(x, y) \to (\forall y)(\exists x)P(x, y)$ i $(\exists x)(\forall y)P(x, y) \to (\forall x)(\exists y)P(x, y)$. *Uputstvo.* Prva formula jeste, a druga nije valjana. Primeniti različite procedure dokazivanja (rezolucija, tablo) u dokazu.

Zadatak 11.81. Isprogramirati prevodilac koji proizvoljnu predikatsku formulu zapisanu u tekstualnoj datoteci učitava i prevodi u drvoidnu strukturu pogodnu za računarsku manipulaciju. *Uputstvo.* Koristiti programska oruđa Lex i Yacc.

Zadatak 11.82. Isprogramirati metodu analitičkih tabloa za ispitivanje valjanosti predikatskih formula.

Zadatak 11.83. Isprogramirati metodu rezolucije za ispitivanje valjanosti predikatskih formula.

Zadatak 11.84. Isprogramirati postupak konstrukcije definicione forme.

Zadatak 11.85. Pokazati da se klase K i D-valjanih formula razlikuju. *Uputstvo.* Koristiti formulu $\Box p \to \Diamond p$ i svet K-modela koji nema ni jedan dostižan svet.

Zadatak 11.86. Pokazati da se klase K i T-valjanih formula razlikuju. *Uputstvo.* Koristiti formulu $\Box p \to p$.

Zadatak 11.87. Pokazati da se klase T i B-valjanih formula razlikuju. *Uputstvo.* Koristiti formulu $p \to \Box\Diamond p$.

Zadatak 11.88. Dokazati preostale slučajeve u teoremi 7.1.9.

Zadatak 11.89. Konstruisati modele u kojima nisu valjane formule $\Box\alpha$, $\alpha \to \Box\alpha$, $\Diamond\alpha$, $\Box\alpha \to \Diamond\alpha$ i $\alpha \to \Diamond\alpha$ i one u kojima to jeste slučaj.

Zadatak 11.90. Dokazati teoreme potpunosti za normalne modalne logike sledeći i razrađujući postupak opisan u teoremi 7.1.10.

Zadatak 11.91. Isprogramirati metodu tabloa za različite modalne logike.

Zadatak 11.92. Koristeći navedene reference, kao i literaturu na koju se u njima upućuje opisati različite pristupe u nemonotonom zaključivanju i isprogamirati odgovarajuće procedure izvođenja.

11.3 Teorija formalnih jezika

Zadatak 11.93. Opisati jezik generisan gramatikom $G = \langle\{S, S_1\}, \{0, 1\}, \{S \to \varepsilon, S \to 01, S \to 0S_11, S_1 \to 01, S_1 \to 0S_11\}, S\rangle$. *Uputstvo.* $L(G) = \{0^n1^n\}$.

Zadatak 11.94. Opisati jezik generisan gramatikom $G = \langle\{S, B, C\}, \{a, b, c\}, \{S \to aSBC, S \to aBC, CB \to BC, aB \to ab, bB \to bb, bC \to bc, cC \to cc\}, S\rangle$.

Uputstvo. $L(G) = \{a^n b^n c^n\}$.

Zadatak 11.95. Proveriti da li reči *abac* i *aabcc* pripadaju jeziku $L(G)$ generisanom gramatikom iz zadatka 11.94.

Zadatak 11.96. Opisati jezik kome odgovara regularni izraz

$$(0^* \cup (((0^*(1 \cup (11)))((00^*)(1 \cup (11)))^*)0^*)).$$

Uputstvo. Jezik sadrži sve binarne reči koje ne sadrže podstring 111.

Zadatak 11.97. Opisati jezik kome odgovara regularni izraz $((0 \cup 1) \cdot (1 \cup 0))^*$

Uputstvo. Jezik sadrži sve binarne reči parnih dužina.

Zadatak 11.98. Ispitati da li je jezik koji se sastoji od prirodnih brojeva deljivih sa 2 regularan. *Uputstvo.* $A = \{0, 1, \ldots, 9\}$. $L_1 = L(0 \cup (1 \cup 2 \cup \ldots \cup 9)(0 \cup 1 \cup \ldots \cup 9)^*)$ je regularan jezik koji sadrži sve prirodne brojeve. $L_2 = L_1 \cap L((0 \cup 1 \cup \ldots \cup 9)^*(0 \cup 2 \cup 4 \cup 6 \cup 8))$ je regularni jezik koji se sastoji od prirodnih brojeva koji se završavaju sa 0, 2, 4, 6 ili 8, pa su deljivi sa 2.

Zadatak 11.99. Ispitati da li je jezik $\{a^p : p$ je prost broj$\}$ regularan. *Uputstvo.* Primenom teoreme 9.3.8 pokazati da bi u jeziku, ako je regularan, bila i reči a^q, gde q nije prost broj.

Zadatak 11.100. Izabrati jedan nedeterministički konačan automat i konstruisati njemu ekvivalentan deterministički automat.

Zadatak 11.101. Napisati progam koji na ulazu prihvata opis proizvoljnog nedeterminističkog konačnog automata i tranformiše ga u njemu ekvivalentan deterministički konačni automat.

Zadatak 11.102. Definisati regularnu gramatiku, konstruisati odgovarajući deterministički konačni automat koji prihvata reč *if* i identifikatore (niz znakova koji počinje slovom iza koga sledi niz slova, cifara i/ili donjih crta) i napisati deo programskog koda kojim se simulira rad automata.

Zadatak 11.103. Definisati regularnu gramatiku i konstruisati odgovarajući deterministički konačni automat koji prihvata službene reči, brojeve i identifikatore programskog jezika Pascal.

Zadatak 11.104. Pokazati da jezik $\{wcw^{-1} : w \in (L(a) \cup L(b))^n\}$ nije regularan, a jeste kontekstno slobodan. *Uputstvo.* Za kontekstno slobodnu gramatiku $G = \langle\{S\}, \{a, b, c\}, \{S \to c, S \to aSa, S \to bSb\}, S\rangle$ je $L(G) = \{wcw^{-1} : w \in (L(a) \cup L(b))^n\}$.

Zadatak 11.105. Napisati izvođenja za reč $(id * id + id) * (id + id)$ u gramatici opisanoj u primeru 9.4.1. Nacrtati odgovarajuće drvo izvođenja.

Zadatak 11.106. Ispitati da li su jezici $\{a^i b^i c^l\}$, $\{a^i b^j c^j\}$ i $\{a^i b^i c^i\}$ kontekstno slobodni.

Zadatak 11.107. Koristeći programe Lex i Yacc napisati program koji za korektno unetu formulu predikatskog računa prvog reda konstruiše i štampa njenu reprezentaciju u formi drveta, a ako formula nije sintaksno korektna ispisuje poruku o grešci.

Zadatak 11.108. Opisati jezik generisan gramatikom $G = \langle\{S, A\}, \{a, b\}, \{S \to AA, A \to AAA, A \to a, A \to bA, A \to Ab\}, S\rangle$. *Uputstvo.* Jezik sadrži sve reči nad alfabetom $\{a, b\}$ u kojima se znak a javlja dva ili bilo koji veći paran broj puta. Indukcijom po dužini izvođenja se pokazuje da sve reči izvedene iz S imaju ukupno paran broj pojava znakova a i A. Za dužinu 1, jedino izvođenje je $S \to AA$, pa tvrđenje važi. Pod pretpostavkom da tvrđenje važi za sva izvođenja dužine k, u

$k+1$ koraku broj pojava znaka A se uvećava za 2 ako se primeni pravilo $A \to AAA$, ne menja ako se primeni pravilo $A \to bA$ ili pravilo $A \to Ab$, a umanjuje se za jedan, dok se broj pojava znaka a uvećava za 1, pa ukupan broj ostaje isti, ako se primeni pravilo $A \to a$. Sa druge strane, pokazuje se da sve svaka reč koja sadrži paran broj pojava znaka a može izvesti iz S. Reč koja sadrži $2n$ pojava znaka a je oblika $b^{m_1}ab^{m_2}a \dots b^{m_{2n}}ab^{m_{2n+1}}$, gde su $m_i \geq 0$. Ovakva reč se izvodi najpre primenom pravila $S \to AA$ i $A \to AAA$ kojima se dobija reč oblika A^{2n}. Primenom pravila $A \to bA$ se dobija reč $b^{m_1}A^{2n}$, a potom se primenom pravila $A \to a$ dobija $b^{m_1}aA^{2n-1}$. Zatim se poslednji deo postupka ponavlja.

11.4 Teorija složenosti izračunavanja

Zadatak 11.109. Dokazati da je $GAP \leq_{ef} CV$.

Zadatak 11.110. Dokazati da problem $RELPRIM = \{\langle x, y\rangle : x \text{ i } y \text{ su uzajamno (relativno) prosti brojevi}\} \in P$. *Uputstvo.* Koristiti Euklidov algoritam za nalaženje najvećeg zajedničkog delioca. Ponavljati dok god $y \neq 0$: $x := x \bmod y$ i zameniti vrednosti x i y; nakon ove petlje u x je najveći zajednički delilac ulaznih brojeva. Ako je $x = 1$, brojevi su uzajamno prosti. Broj prolaza kroz petlju je reda $\log_2(\min\{x, y\})$, što je proporcionalno veličini ulaza koji čine dva binarna broja.

Zadatak 11.111. Dokazati da problem $MODEXP = \{\langle a, b, c, d\rangle : a, b, c, d \text{ su binarni brojevi za koje je: } a^b = c(\bmod\ d)\}$ pripada klasi složenosti P. *Uputstvo.* Stepenovanje svesti na izračunavanje a^0, a^1, a^2, a^4, ...

Zadatak 11.112. Dokazati da je svaki kontekstno slobodan jezik u klasi složenosti P.

Zadatak 11.113. Dokazati da problem $COMPOSITES = \{x : x = pq \text{ za cele brojeve veće od } 1\} \in NP$.

Zadatak 11.114. Dokazati da problem $CLIQUE = \{\langle G, k\rangle : G \text{ je neorijentisani graf koji ima } k\text{-clique}\} \in NP$. *Uputstvo.* Nedeterministički izabrati podskup c od k čvorova i proveriti da li su svaka dva čvora povezana ivicom.

Zadatak 11.115. Dokazati da problem $SUBSETSUM = \{\langle S, t\rangle : S \text{ je skup brojeva, a suma elemenata jednog njegovog podskupa je } t\} \in NP$. *Uputstvo.* Nedeterministički izabrati podskup c i proveriti da li je suma elemenata jednaka t.

Zadatak 11.116. Dokazati da problem $PRIMES = \{n : n \text{ je prost broj}\} \in NP$. *Uputstvo.* Za $p > 1$, multiplikativna grupa $Z_p^* = \{x : x \text{ i } p \text{ su uzajamno prosti i } 1 \leq x < p\}$ je ciklična i reda $p - 1$ ako i samo ako je p prost broj. Grupa je ciklična ako $(\exists x)(\forall a \in G)(\exists n)a = x^n$.

Zadatak 11.117. Dokazati da su problemi $SUBSETSUM$ i bojenja čvorova grafa sa 3 boje NP-kompletni.

Literatura

[1] M. Abadi, J. Halpern, Decidability and expressivness for first-order logics of probability, Information and computation, 112, 1 – 36, 1994.

[2] W. Ackermann, Solvable cases of the decision problem, North-Holland, 1954.

[3] D. Acketa, Odabrana poglavlja teorije prepoznavanja oblika, Univerzitet u Novom Sadu, 1986.

[4] M.Agrawal, N. Kayal, N. Saxena, Discovery of a deterministic Polynomial time algorithm for testing whether or not a number is prime, http://www.cse.iitk.ac.in/users/manindra/index.html, 2002.

[5] A. Aho, R. Sethi, J. Ullman, Compilers, principles, techniques, and tools, Addison-Wesley, 1986.

[6] F. Baader, Logic-Based Knowledge Representation, In M.J. Wooldridge and M. Veloso, editors, Artificial Intelligence Today, Recent Trends and Developments, Lecture Notes in Computer Science, 1600, 13–41. Springer Verlag, 1999.

[7] F. Baader, I. Horrocks, U. Sattler, Description Logics for the Semantic Web, KI - Künstliche Intelligenz, 4, 2002.

[8] F. Baader, D. McGuinness, D. Nardi, P. Patel - Schneider, The Description Logic Handbook: Theory, implementation, and application, Cambridge University Press, ISBN: 0521781760, 2003.

[9] J. Barwise, An introduction to first-order logic, u Handbook of mathematical logic, editor Jon Barwise, 5 – 46, North-Holland, 1977.

[10] G. Boole, An investigation of the laws of thought, on which are founded mathematical theories of logic and probability (Laws of thought), Wakton and Maberley, London, 1854.

[11] A. Borgida, On The Relationship between Description Logic and Predicate Logic Queries,Conference on Information and Knowledge Managment, Proceedings of the third international conference on Information and knowledge management Gaithersburg, Maryland, United States, ISBN:0-89791-674-3, ACM Press New York, 219 – 225, 1994.

[12] G. Boolos, R. Jeffrey, Computability and logic, Cambridge University Press, 1974.

[13] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, P. McKenzie, Systems and software verification: Model-checking techniques and tools, Springer, 2001.

[14] J. Burgess, Basic tense logic, u: Handbook of philosophical logic, edts.: Dov M. Gabbay, F. Günthner, 89 – 132, D. Reidel Publishing company, 1984.

[15] S. Buss, A. Kechris, A. Pillay, R. Shore, The prospects for mathematical logic in the twenty-first century, The Bulletin of Symbolic Logic, Volume 7, Issue 2, 169 – 196, 2001.

[16] A. Chagarov, M. Zakharyaschev, Modal logic, Calderon Press, 1997.

[17] C. Chang, H. Keisler, Model theory, North-Holland, 1973.

[18] C. Chang, R. Lee, Symbolic logic and mechanical theorem proving, Academic press, 1973.

[19] N. Chomsky, Three models for the description of languages, IRE Transactions on information theory, 2, no. 3, 113–124, 1956.

[20] N. Cutland, Computability, an introduction to recursive funkction theory, Cambridge university press, 1986.

[21] M.Ćirić, T.Petković, S.Bogdanović, Jezici i automati, Prosveta, Niš, 2000.

[22] V. Damjanović, Semantički Web, ontologije, agenti, Specijalistički rad, Fakultet Organizacionih nauka, Novembar 2003.

[23] M. Davis, E. Weyuker, Computability, complexity and languages, Academic Press, 1983.

[24] F.Donini, M. Lenzerini, D. Nardi, and A. Schaerf In Gerhard Brewka, editor, Foundation of Knowledge Representation, 191–236. CSLI-Publications, 1996.

[25] R. Đorđević, M. Rašković, Z. Ognjanović, Completeness Theorem for Propositional Probabilistic Models whose Measures have only Finite Ranges, Archive for Mathematical Logic, 43, 557 –563, 2004.

[26] E. Eder, Relative complexities of first order calculi, Vieweg, 1992.

[27] A. Emerson, Temporal and modal logic, u: Handbook of theoretical computer science, edt. J. van Leeuwen, 995 – 1072, Elsevier Science Publishers B.V., 1990.

[28] Encyclopedic dictionary of mathematics, editori Iyanaga, Shocichi, Kawada, Yukiyosi, MIT Press, 1968.

[29] R. Epstein, W. Carnielli, Computability, computable functions, logic and the foundations of mathematics, Wadsworth & Brooks/Cole, 1989.

[30] Evolutionary computation 1 (Basic algorithms and operators) and 2 (Advanced algorithms and operators), edt. T. Bäck, D.B. Fogel, T. Michalewicz, Institute of Physics Publishing, 2000.

[31] R. Fagin, J. Halpern, N. Megiddo, A logic for reasoning about probabilities, Information and computation 87, 78 – 128, 1990.

[32] R. Fagin, Y. Halpern, Reasoning about knowledge and probability, Journal of the ACM, vol. 41, no 2, 340 – 367, 1994.

[33] M. Fitting, Intuitionistic logic, model theory and forcing, North-Holland Publishing Company, 1969.

[34] M. Fitting, Proof methods for modal and intuitionistic logics, D. Reidel, 1983.

[35] M. Fitting, Basic modal logic, u: Handbook of logic in artificial intelligence and logic programming, edts.: Dov M. Gabbay, C. J. Hooger, J. A. Robinson, vol. 1, 385 – 448, Clarendon press, 1993.

[36] N. Friedman, J. Halpern, Plausibility measures and default reasoning, Journal of the ACM 48:4, 648–685, 2001.

[37] J. Garson, Quantification in modal logic, u: Handbook of philosophical logic, edts.: Dov M. Gabbay, F. Günthner, 249 – 307, D. Reidel Publishing company, 1984.

[38] R. Garnier, J. Taylor, Discrete mathematics for new technology, Institute of Physics Publishing, 2002.

[39] O.Goldreich, Modern cryptography, probabilistic proofs and pseudorandomness, Springer, 1999.

[40] M. Goldszmidt, J. Pearl, Qualitative Probabilities for Default Reasoning, Belief Revision, and Causal Modeling, Artificial Intelligence, Vol. 84, No. 1-2, 57–112, 1996.

[41] G. Hachtel, F. Somenzi, Logic synthesis and verification algorithms, Kluwer academic publishers, 1996.

[42] T. Hailperin, Boole's logic and probability: a critical exposition from the standpoint of contemporary algebra, logic and probability theory, North-Holland, 1976.

[43] P. Hájek, Methamathematics of fuzzy logic, Kluwer Academic Publishers, 1998.

[44] P. Halmos, Lectures on Boolean Algebras, Springer-Verlag, 1974.

[45] J. Halpern, M. Vardi, Model checking vs. theorem proving: a manifesto, Proceedings of the 2. International conference Principles of knowledge representation and reasoning, edts: J. A. Allen, R. Fikes, E. Sandewall, 325 – 334, 1991.

[46] J. Halpern, Reasoning about knowledge: a survey, u: Handbook of logic in artificial intelligence and logic programming, edts.: Dov M. Gabbay, C. J. Hooger, J. A. Robinson, vol. 4, 1 – 34, Clarendon press, 1995.

[47] J. Halpern, R. Harper, N. Immerman, P. Kolaitis, M. Vardi, V. Vianu, On the unusual effectiveness of logic in computer science, The Bulletin of Symbolic Logic, Volume 7, Issue 2, 213–326, 2001.

[48] Handbook of theoretical computer science, vol. A, Algorithms and complexity, edt. Van Leeuwen, Jan, Elsevier, Amsterdam, 1990.

[49] D. Harel, Dynamic logic, u: Handbook of philosophical logic, edts.: Dov M. Gabbay, F. Günthner, 497 – 604, D. Reidel Publishing companu, 1984.

[50] The Harper encyclopedia of science, editor Wyckoff, Jerome, Harper & Row, 1963.

[51] J.Hopcroft, J.Ullman Formal languages and their relation to automata, Addison-Wesley, 1969.

[52] I. Horrocks, Optimising tableaux decision procedures for description logics, PhD Thesis, University of Manchester, http://www.cs.man.ac.uk/ horrocks/Publications/publications.html, 1997.

[53] P. Hotomski, I. Pevac, Matematički i programski problemi veštačke inteligencije u oblasti automatskog dokazivanja teorema, Naučna knjiga, Beograd, 1988.

[54] G. Hughes, M. Cresswell, An introduction to modal logic, Methuen, 1968.

[55] G. Hughes, M. Cresswell, A companion to modal logic, Methuen, 1984.

[56] N. Ikodinović, Z. Ognjanović, A logic with coherent conditional probabilities, Lecture Notes in Computer Science Vol. 3571, 726–736, Springer-Verlag, 2005.

[57] N. Ikodinović, M. Rašković, Z. Marković, Z. Ognjanović, Measure Logic, Lecture notes in computer science, 2007.

[58] N.Immerman, Descriptive complexity, Springer, 1998.

[59] P.Janičić, Matematička logika u računarstvu, ispit 4. godine R smera Matematičkog fakulteta u Beogradu, http://www.matf.bg.ac.yu/∼janicic, 2003.

[60] M. Jocković, Z. Ognjanović, S. Stankovski, Veštačka inteligencija, inteligentne mašine i sistemi, Krug, Beograd, 1998.

[61] D. Jovanović, N. Mladenović, Z. Ognjanović, Variable Neighborhood Search for the Probabilistic Satisfiability Problem, Proceedings of the 6th Metaheuristics International Conference, August 22-26, 2005, Vienna, Austria, edt. Richard F. Hartl, 557–562, 2005.

[62] M. Kapetanović, A. Krapež, A proof procedure for the first order logic, Publications de L'Institut Mathematique, n.s. 59 (73), 3–5, 1989.

[63] J. Keisler, Hyperfinite model theory, u zborniku radova Logic colloquim 76, editori Gandy, R. O., Hyland, J. M. E., North-Holland, 1977.

[64] J. Keisler, Probability quantifiers, glava XIV u knjizi Model-theoretic logics, editori Barwise, J., Feferman, S., edicija Perspectives in mathematical logic, Springer-Verlag, 1985.

[65] S. Kleene, Izračunljivost, odlučivost i teoreme nepotpunosti, u: Novija filozofija matematike, priredio Zvonimir Šikić, prevod Dean Rosenzweig iz V poglavlja knjige Kleene, S., Mathematical logic, John Wiley & Sons, 1967, Nolit, 1987.

[66] D. Knuth, On the translation of languages from left to right, Information and control, vol. 8, no. 6, 607 – 639, 1965.

[67] A. Kolmogorov, Three approaches to the quantitative definition of information, Probl. of information transmission, vol. 1, no. 1, 1 – 7, 1965.

[68] J. Kratica, Paralelizacija genetskih algoritama za rešavanje nekih NP-kompletnih problema, doktorska disertacija, Univerzitet u Beogradu, Matematički fakultet, 2000.

[69] S. Kraus, D. Lehmann, M. Magidor, Nonmonotonic reasoning, preferential models and cumulative logics,Artificial Intelligence, 44:167 – 207, 1990.

[70] A. Kron, Odnos polivalentnih logika i teorije verovatnoće, Naučna knjiga, Beograd, 1967.

[71] A. Kron, Logika, Univerzitet u Beogradu, 1998.

[72] D. Lehmann, M. Magidor, What does a conditional knowledge base entail?, Artificial Intelligence 55:1 – 60, 1992.

[73] H. Lewis, C. Papadimitriou, Elements of the theory of computation, Prentice-Hall, 1981.

[74] O. Lichtenstein, A. Pnueli, Propositional temporal logics: decidability and completeness, L.J. of the IGPL, vol. 8, no. 1, 55 – 85, 2000.

[75] C. Lutz, The Complexity of Description Logics with Concrete Domains, PhD Thesis, University of Hamburg, http://lat.inf.tu-dresden.de/ clu/papers/, 2002.

[76] J.A. Makowsky, Lecture Notes of Logic for Computer Science 1, part I, II, http://www.cs.technion.ac.il/ janos/COURSES/CD/overview.html, 1995.

[77] Z. Marković, Pregled algoritamskih sistema, Računarstvo u nauci i obrazovanju, br. 1, 7 – 14, 1987.

[78] Z. Marković, Z. Ognjanović, M. Rašković, A Probabilistic Extension of Intuitionistic Logic, Mathematical Logic Quarterly, vol. 49, 415–424, 2003.

[79] Z. Marković, Z. Ognjanović, M. Rašković, An intuitionistic logic with probabilistic operators, Publications de L'Institute Matematique, n.s. 73 (87), 31 – 38, 2003.

[80] Z. Marković, M. Rašković, Z. Ognjanović, Completeness theorem for a logic with imprecise and conditional probabilities, Publications de L'Institute Matematique (Beograd), ns. 78 (92) 35 – 49, 2005.

[81] Ž. Mijajlović, Z. Marković, K. Došen, Hilbertovi problemi i logika, Zavod za udžbenike i nastavna sredstva, Beograd, 1986.

[82] Ž. Mijajlović, Algebra 1, MILGOR, Beograd - Moskva, 1998.

[83] M. Miličić, Description Logics with Concrete Domains and Functional Dependencies, Master's thesis, Dresden University of Technology, http://wwwtcs.inf.tu-dresden.de/ maja/papers/thesis.pdf, 2004.

[84] J. Minker, An overview of nonmonotonic logic and logic programing, http://citeseer.nj.nec.com/minker93overview.html, 1993.

[85] N. Nilsson, Probabilistic logic, Artificial intelligence 28, 71 – 87, 1986.

[86] N. Nilsson, Probabilistic logic revisited, Artificial intelligence 59, 39 – 42, 1993.

[87] P. Odifreddi, Classical Recursion theory. The Theory of Functions and Sets of Natutral Numbers, Studies in Logic and the Foundations of Mathematics, vol. 125, North-Holland, 1989.

[88] Z. Ognjanović, Dokazivač teorema u modalnom racunu S4 zasnovan na metodi dualnih tabloa, magistarski rad, Matematički fakultet u Beogradu, 1993.

[89] Z. Ognjanović, A tableau-like proof procedure for normal modal logics, Theoretical Computer Science 129, 167–186, 1994.

[90] Z. Ognjanović, M. Rašković, A logic with higher order probabilities, Publications de L'Institute Matematique (Beograd), 60 (74), 1–4, 1996.

[91] Z. Ognjanović, A logic for temporal and probabilistic reasoning, Workshop on Probabilistic Logic and Randomised Computation, ESSLLI '98, Saarbruecken, Germany, August, 1998.

[92] Z. Ognjanović, Miodrag Rašković, Some probability logics with new types of probability operators, Journal of Logic and Computation, Volume 9, Issue 2, 181 – 195, 1999.

[93] Z. Ognjanović, Neke verovatnosne logike i njihove primene u računarstvu, doktorska teza, Prirodno-matematički fakultet, Univerzitet u Kragujevcu, 1999.

[94] Z. Ognjanović, M. Rašković, Some first order probability logics, Theoretical Computer Science, Vol. 247, No. 1-2, 191 – 212, 2000.

[95] Z. Ognjanović, A completeness theorem for a first order linear-time logic, Publications de L'Institute Matematique (Beograd), ns. 69 (83), 1–7, 2001.

[96] Z. Ognjanović, J. Kratica, M. Milovanović, A genetic algorithm for satisfiability problem in a probabilistic logic: A first report, 6th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2001, Toulouse. Lecture notes in computer science (LNCS/LNAI), 2143, 805 – 816, Springer-Verlag, 2001.

[97] Z. Ognjanović, J. Kratica, M. Milovanović, Primena genetskih algoritama na SAT problem, Zbornik konferencije XXVIII Jugoslovenski simpozijum o operacionim istraživanjima SYMOPIS 2001, Beograd 2. do 5. 10. 2001, edts. S. Minić, S. Borović, N. Petrović, 289 – 292, Beograd 2001.

[98] Z. Ognjanović, U. Midić, J. Kratica, A Genetic Algorithm for Probabilistic SAT Problem, to appear in Proceedings of the 7th Conference "Artificial Intelligence and Soft Computing", June 7-11, 2004, Zakopane, Poland, Lecture notes in computer science (LNCS/LNAI), Springer-Verlag, 2004.

[99] Z. Ognjanović, N. Ikodinović, Z. Marković, A logic with Kolmogorov style conditional probabilities, Proceedings of the 5th Panhellenic logic symposium, Athens, Greece, July 25-28, 2005, 111–116, 2005.

[100] Z. Ognjanović, U. Midič, N. Mladenović, A Hybrid Genetic and Variable Neighborhood Descent for Probabilistic SAT Problem, Lecture Notes in Computer Science vol. 3636, 42–53, Springer-Verlag, 2005.

[101] Z. Ognjanović, Discrete Linear-time Probabilistic Logics: Completeness, Decidability and Complexity, Journal of Logic Computation, Vol. 16, No. 2, 257–285, 2006.

[102] Z. Ognjanović, A. Perović, M. Rašković, Logics with the Qualitative Probability Operator, Logic Journal of the Interest Group in Pure and Applied Logic, 2007.

[103] N. Styazhkin, Histroy of mathematical logic from Leibniz to Peano, MIT Press, 1969.

[104] D. Tošić, N. Mladenović, J. Kratica, V. Filipović. Genetski algoritmi, Matematičk institut, Beograd, 2004.

[105] C. Papadimitriou, Computational complexity, Addison-Wesley, 1995.

[106] M. Rašković, Classical logic with some probability operators, Publications de L'Institute Matematique, ns. vol. 53 (67), 1 – 3, 1993.

[107] M. Rašković, Z. Ognjanović, A first order probability logic - LP_Q, Publications de L'Institute Matematique (Beograd), ns. 65 (79), 1-7, 1999.

[108] M. Rašković, Z. Ognjanović, Z. Marković, A Logic with Conditional Probabilities, Lecture notes in artificial intelligence (LNCS/LNAI), 3229, 226 – 238, Springer-Verlag, 2004.

[109] M. Rašković, Z. Marković, Z. Ognjanović, A Logic with approximate conditional probabilities that can Model Default Reasoning, International Journal of Approximate Reasoning, 2007.

[110] J. Peyton, L. Simon, The implementation of functional programming languages, Prentice Hall, 1987.

[111] D. Poole, Default logic, u: Handbook of logic in artificial intelligence and logic programming, edts.: Dov M. Gabbay, C. J. Hooger, J. A. Robinson, vol. 3, 189 – 215, Clarendon press, 1994.

[112] S. Prešić, Elementi matematičke logike, Zavod za udžbenike i nstavna sredstva, 1983.

[113] R. Robinson, A machine-oriented logic based on the resolution principle, Journal of the ACM, 12, 23–41, 1965.

[114] T. Schaub, The automation of reasoning with incomplete information, Lecture notes in artificial intelligence 1409, Springer, 1998.

[115] B. Schneier, Applied cryptography. Protocols, algorithms and source code in C, John Wiley& Sons, 1993.

[116] U. Schöning, Logic for computer scientists, Birkhäuser, 1989.

[117] G. Sénizergues, L(A)=L(B)?, decidability results from complete formal systems, Theoretical Compututer Science, 251, no. 1-2, 1–166, 2001.

[118] M. Sipser, Introduction to the theory of computation, PWS publishing company, 1997.

[119] R. Smullyan, First-order logic, Springer-Verlag, 1968.

[120] I. Spasić, P. Janičić, Teorija algoritama, jezika i automata. Zbirka zadataka, Matematički fakultet, Beograd, 2000.

[121] L. Stockmeyer, Classifying the computational complexity on problems, The journal of symbolic logic vol. 52, no. 1, 1 – 43, 1987.

[122] P. Subašić, Fazi logika i neuronske mreže, Tehnička knjiga, Beograd, 1997.

[123] R. Turner, Logics for artificial intelligence, John Wiley & sons, 1984.

[124] A. Urquhart, Many-valued logic, u: Handbook of philosophical logic, vol III, edts.: Dov M. Gabbay, F. Günthner, 71 – 116, D. Reidel Publishing companu, 1986.

[125] M. Vardi, An automata-theoretic approach to linear temporal logic, Proceedings of the 2nd Panhellenic logic symposium, edts. Kolaitis, P., Koletsos, G., 6 – 34, 1999.

[126] J. van Benthem, Temporal logic, u: Handbook of logic in artificial intelligence and logic programming, edts.: Dov M. Gabbay, C. J. Hooger, J. A. Robinson, vol. 4, 241 – 350, Clarendon press, 1995.

[127] S. Vujošević, Matematička logika, o mogućnostima formalnog metoda, CID Podgorica, 1996.

[128] A. Yasuhara, Recursive function theory & logic, Academic Press, 1971.

[129] L. Zadeh, Fuzzy sets, Information and control 8, 338 – 353, 1965.

[130] M. Živković, Algoritmi, Matematički fakultet, Beograd, 2000.

Indeks