Abduction and Logic Programming

Lorenz Leutgeb
e1127842@student.tuwien.ac.at

Abstract

Abduction is a form of reasoning aimed at obtaining explanations for observations under a given theory. We summarize how abduction can be modeled by deduction for the case where the theory is a logic program. A framework that accommodates a wide range of abduction problems in view of different application domains by varying semantics of inference towards the explanation is used and described. Abduction can also be modeled by argumentation, which is closely related to logic programming. A correspondence is explored.

1 Introduction

Abduction, next to deduction and induction, is a form of logical reasoning where, given a theory and some observations, the goal is to find an explanation for the observations under the theory. It is a very intuitive reasoning task, carried out by the human mind in a continuous fashion, which somehow relates to what one might call "common sense".

This work is motivated by the fascination that abduction can be modeled by deduction and an interest in logic programming. It aims to establish links between three works that take different approaches to abduction:

Abduction via Deduction In Section 1.1 we revisit logical reasoning and establish an intuition for the three aforementioned forms of reasoning using short examples in order to explain how Console et al. [4] achieve abduction through deduction in Section 3.

A Framework for Abduction Problems While Eiter et al. [7] were mainly interested in surveying the complexity and discovering mappings between different abduction problems, they defined a versatile framework to talk about abduction problems as logic programs under given semantics. In Section 1.2 logic programming (as a form of declarative programming) is informally described and very briefly contrasted with imperative programming, which is much more widespread in application. A more formal perspective on logic programs is given in Section 2.1, as they are the foundation of Section 4, where said framework generalizing abduction problems modeled by logic programming is described.

Abduction in Argumentation Abstract argumentation is a more recent notion that closely relates to logic programming. We will introduce it in Section 1.3, and give a formal definition of argumentation frameworks in Section 2.2. In Section 5 we will then explore how Booth et al. [2] model abduction in argumentation, and highlight the connection to logic programming.

1.1 Deduction, Induction and Abduction

We use the work of Peirce [15] to highlight the underlying schema and differences between three types of reasoning by means of syllogisms[1]:

[1] The interested reader may refer to [5] resp. [14] in order to find out more about the philosophical roots of abduction or Peirce's Logic in general.

Deduction

	Rule.	All the beans from this bag are white.
	Case.	These beans are from this bag.
∴	*Result.*	These beans are white.

Induction

	Case.	These beans are from this bag.
	Result.	These beans are white.
∴	*Rule.*	All the beans from this bag are white.

Abduction

	Rule.	All the beans from this bag are white.
	Result.	These beans are white.
∴	*Case.*	These beans are from this bag.

The three types of reasoning are obtained by arranging *Rule*, *Case* and *Result* so that two of them act as premises, concluding the third. Three examples: (1) Mathematics heavily depend on deduction to derive more explicit knowledge from basic axioms. (2) Physicists employ experiments and their results to derive the laws of physics, which can be regarded as induction. (3) A physician uses medical knowledge to explain the symptoms of her patient, a diagnosis obtained by abduction.

Abduction is of particular interest in the field of artificial intelligence, when it comes to generating explanations, diagnoses and plans but has also been successfully applied to speech recognition, vision and learning [4, 7, Introductions].

In the above syllogism showing abduction, the *Rule* corresponds with certain knowledge or a theory, while the *Result* stands for an observation and the *Case* maps to an explanation, diagnosis, plan and similar.

It is important to note that abduction might result in multiple explanations and it is in general not sound, as some explanations (or diagnoses) may turn out false [7]. However, this is only at the meta-level, meaning that the theory can be adjusted to lead to better results.

1.2 Logic Programming

Logic programming is regarded a very powerful approach to declarative programming. Rather than an algorithm to solve a problem, the problem itself (and its solution, to some extent) is described. As a logic program does not define any kind of control flow, it is not executed directly, like imperative programs, but instead put into a solver (which in turn is also a program). Similar to how imperative programs are first written, then (in some cases) interpreted by an interpreter, a logic program is written and then solutions are obtained by a solver.

Take the Clique problem for example: Given a graph its solution is the set of all fully connected sub-graphs. Writing an imperative program (be it in Java, C#, Go or a similar language) to solve Clique is not straightforward, though certainly a reasonable task for a second-semester computer science student. The student has to account for the control flow of the program, keep track of the solutions already obtained, and so forth. Using a non-hosted language she might even have to deal with allocating and deallocating memory. With logic programming in turn, the task of the programmer is just to formalize the problem in a way that the solver understands, which normally turns out to be the easier task.

When first looking at the structure of logic programs (see Section 2) the structure of the rules, essentially implications, appears strikingly similar to the deductive nature of mathematical logic. This makes logic programming an interesting vehicle for encoding theories and knowledge in the form of rules. What is not so obvious, is how these programs can be transformed in such a way that abduction is modeled deductively, as we will see in Section 3.

For an introduction to and overview of Answer Set Programming, see [8].

1.3 Abstract Argumentation

Abstract argumentation introduced by Dung [6] is an active field of artificial intelligence research. While the field was inspired by the human capability "to understand new concepts, to perform scientific reasoning, to express, clarify defend their opinions in their daily lives" through argumentation and dialogue, its core notion of an argumentation framework (see Definition 3) is best understood as a set of arguments that might conflict with one another. This makes abstract argumentation interesting for problems where conflict resolution is key.[2]

Exploiting the simplicity of AFs, many semantics were presented that define how to compute extensions from an AF and therefore what constitutes a desirable (sub-)set of arguments. These semantics are competing in some sense and inspire new lines of research. Identifying classes of semantics, i.e. using pairwise agreement on solutions between semantics and other relations, is also active research.

Also, Dung [6] argues that "many major approaches to nonmonotonic reasoning in AI and logic programming are in fact different forms of [abstract argumentation]" [6, Introduction, p. 325]. Through this connection we will look at how abduction from logic programming fits into the paradigm of argumentation frameworks as laid out by Booth et al. [2].

2 Preliminaries

We briefly revisit logic programs, as they are fundamental to both [4] and [7].

2.1 Logic Programs

Definition 1 (see [18, Slide 15])**.** *A* rule *is an ordered pair of the form*

$$a_1 \vee \ldots \vee a_m \leftarrow b_1 \wedge \ldots \wedge b_k \wedge \text{not } b_{k+1} \wedge \ldots \wedge \text{not } b_n$$

where $a_1, \ldots, a_m, b_1, \ldots, b_n$ *are literals,* not *is* negation as failure *(or* default negation*),* $a_1 \vee \ldots \vee a_m$ *is the* head *of* r *and* $b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_n$ *is the* body *of* r*.*

Some more concrete classes of rules are of interest: A rule is a fact *if* $n = 0$ *and* $m \geq 1$*;* basic *if* $n = k$ *and* $m \geq 1$*;* non-disjunctive *if* $m = 1$*;* normal *if it is non-disjunctive and contains no strong negation* $\neg$*;* Horn *if it is normal and basic;* ground *if all its literals are ground.*

Definition 2. *A logic program (usually denoted LP) is a finite set of rules. Analogous to the classification of a rule, a program is basic, normal, Horn etc. if all of its rules are.*

[2] For an introductory lecture refer to [1].

When evaluating logic programs, there is no canonical semantics, especially considering incomplete information and default negation. Two possible approaches are described in Section 4, as the core idea of the framework in [7] is varying semantics for inference.

2.2 Argumentation Frameworks

In Section 1.3 we will build upon the notion of argumentation frameworks (AF).

Definition 3 (originally [6, Definition 2], taken from [2, Definition 1])**.** *Given a countable infinite set $\mathcal{U}$ called the* universe of arguments*, an* argumentation framework (AF) *is a pair $F = (A, \rightsquigarrow)$ where A is a finite subset of $\mathcal{U}$ and $\rightsquigarrow$ a binary relation over A. If $a \rightsquigarrow b$ we say that a* attacks *b. $\mathcal{F}$ denotes the set of all argumentation frameworks.*

3 Abduction via Deduction

From the above introduction, abduction and deduction seem to be distinct and it is not obvious how abduction can be implemented or programmed. It is however possible to derive the conclusions of abduction deductively.

The "fundamental result" of [4] is "highlighting the bridge between abduction and deduction through the completion semantics". We will use this section of the paper to familiarize ourselves with their result. In order to resemble the main theorem, we import Clark completion [3] as well as one meta-level and one object-level definition of abduction problems.

Let us consider the concrete example ([13, p. 371] as cited in [4, p. 663]) of a simple interpretation problem represented by the following domain theory LP_1:

$$
\begin{aligned}
T_1 = \{ \quad & grass\_is\_wet \leftarrow rained\_last\_night, \\
& grass\_is\_wet \leftarrow sprinkler\_was\_on, \\
& grass\_is\_cold\_and\_shiny \leftarrow grass\_is\_wet, \\
& shoes\_are\_wet \leftarrow grass\_is\_wet \quad \}
\end{aligned}
$$

Now, assume that we want to obtain an explanation for our manifestation $M_1 \equiv grass\_is\_cold\_and\_shiny$, that is we ask ourselves: "Why might the grass be shiny?" Intuitively, we will look at the theory and try to to follow implications from consequent to antecedent (resp. left to right), starting at the observation and leading to the two atoms *rained_last_night* and *sprinkler_was_on*. Indeed, in this case, the two atoms coincide with the two minimal explanations of M_1:

$$
\begin{aligned}
S_1 &= \{rained\_last\_night\} \\
S_2 &= \{sprinkler\_was\_on\}
\end{aligned}
$$

More generally, we firstly notice that the only "sensible" explanations (or hypotheses) for our observation are the atoms that are reached last when intuitively following implications from consequent to antecedent: The set of atoms that do not occur in the head of any rule of the theory are called *abducible atoms* (terminology from [9] as cited in [4, p. 664]). Secondly, to reach a formal definition of the intuition, Clark completion [3], a transformation that allows (again, intuitively) following implications of the theory in reverse is applied.

The completion of non-abducible predicates of a theory LP, denoted LP_C is formally defined in [4] as follows: The completion LP_C is a set of equivalences

$\{p_i \leftrightarrow D_i | i = 1, \ldots, n\}$, where $p_1, \ldots p_n$ are all the non-abdicuble atoms in LP and $D_i \equiv Q_{i1} \vee \ldots \vee Q_{im}$ in case $\{Q_{ij} \rightarrow p_i | j = 1, \ldots, m\}$ is the set of clauses in LP having p_i as their head. Applying this transformation to our example leads to:

$$\begin{aligned} LP_{1,C} = \{ \quad & grass\_is\_wet \leftrightarrow \quad rained\_last\_night \vee sprinkler\_was\_on, \\ & grass\_is\_cold\_and\_shiny \leftrightarrow \quad grass\_is\_wet, \\ & shoes\_are\_wet \leftrightarrow \quad grass\_is\_wet \quad \} \end{aligned}$$

Moreover, we require the theory LP to be *hierarchical*, meaning that it is possible to assign a rank to every atom such that for every rule in the theory the rank of the atoms in the head is strictly higher than the rank associated with any atom in the body of the rule. Practically this means that a directed graph where vertices map to atoms and edges map to body/head relationship in a rule is acyclic. The assumption is needed to show the termination of a procedure traversing the rules in a similar manner as above. Notice that abducible atoms will be of the lowest rank, and the strict order implies termination. [4, cf. Section 3, p. 667, 669]

Now that we have an intuition for what an abduction problem consists, a sense of what to expect as an explanation, and a rough idea of a transformation that will help generating the explanation, we go ahead with a definition of an abduction problem. Here, we use the more general definition from the framework postulated in [7].

Definition 4 ([7, Definition 1, p. 140])**.** *Let V be a set of propositional atoms. A logic programming abduction problem (LPAP) $\mathscr{P}$ over V consists of a tuple $\langle H, M, LP, \models \rangle$ where $H \subseteq V$ is a finite set of hypotheses, $M \subseteq V \cup \{\neg v | v \in V\}$ is a finite set of manifestations, LP is a propositional logic program on V and $\models$ is an inference operator.*

Console et al. [4] additionally require LP to be hierarchical.

With this, the next immediate question is how explanations can be characterized. A meta-level definition is presented first, and an object-level definition presented second.

Definition 5. *Let $\mathscr{P} = \langle H, M, LP, \models \rangle$ be a LPAP, and let $S \subseteq H$, then S is a* solution *(or* m-explanation*) to (resp. for) $\mathscr{P}$ iff $LP \cup S \models M$. The set of solutions of $\mathscr{P}$ is denoted* $\mathrm{Sol}(\mathscr{P})$.

Remark 1. *Two comments on the differences between the definition of an m-explanation in [4, Definition 3, p. 671] and a solution in [7, Definition 2, p. 140] for clarity: Console et al. [4] restrict the reference operator to $\vdash_{NF}$, the SLDNF derivation, i.e. the query evaluation procedure from [3], while Eiter et al. [7] were interested in a more general framework for arbitrary inference relations (elaborated in Section 4). Notation also slightly differs, as observations, called manifestations, are denoted Ψ and solutions, called m-explanations, are denoted E in [4].*

Definition 6 (adapted from [4, Definition 2, p. 669])**.** *Let $\mathscr{P} = \langle H, M, LP, \vdash_{NF} \rangle$ be a be a LPAP using SLDNF derivation and LP_C the (Clark) completion of non-abducible predicates in LP. The* explanation formula *for $\mathscr{P}$ is the most specific formula F in the language of abducible atoms such that:*

$$LP_C, M \models F$$

where F is more specific than F' iff $\models F \rightarrow F'$.

With this, a procedure to compute an explanation formula F is described in [4], that produces F from LP_C by substituting non-abducible atoms p_i with their D_i counterpart from the completion until only abducible atoms are present. They argue that this procedure halts on the grounds that LP is assumed to be hierarchical. That the sketched procedure indeed results in F is shown in [4, Theorem 1].

A correspondence between the object-level and meta-level definition of explanations for abduction problems, resembles the connection between deduction on the object level and abduction on the meta-level.

Theorem 1 (adapted from [4, Theorem 2, p. 671]). *Let $\mathscr{P} = \langle H, M, LP, \vdash_{NF} \rangle$ be a be a LPAP using SLDNF derivation having F as the explanation formula. Let S be a set of abducible atoms and I an interpretation such that for every abducible atom α*

$$I \models \alpha \text{ iff } \alpha \in S$$

Then S is a solution (an m-explanation) iff $S \models F$.

Proof of Theorem 1 is omitted for brevity.

Further Console et al. [4] describe how above definitions can be extended to account for dependence between abducible atoms (additional knowledge about abducibles). They account for two types:

- *taxonomic* or *abstraction relationships* between abducible atoms, i.e. implications of the form
$$\alpha \rightarrow \beta$$
(where α and β are abducible atoms);
- *constraints* between abducible atoms in the form of denials, i.e. of the form
$$\neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$$
where $\alpha_i, \ldots, \alpha_n$ are abducible atoms.

Again, transformations on the object level are described [4, Section 4.1], and correspondence with the meta-level [4, Section 4.2] is shown.

Lastly, they provide an extension to the first order case that stems on an "equality theory included in the completion" [4, Section 5.1, p. 684].

4 A Framework for Abduction Problems

Eiter et al. [7] formulated a general framework for abduction problems allowing variation in the semantics of inference and compared the complexity of common semantics.

The flexibility of their framework is evident in Definition 4, which was already used in the previous section, as it allows for an arbitrary inference operator. They justify this by explaining that different semantics may be needed depending on the application domain.

Briefly, they think that $\models_{wf}$, $\models^b_{st}$, $\models^c_{st}$ are of interest, defined as follows:

Definition 7 (see [11, Section 2]). *Given a logic program LP and an interpretation I, the* Gelfond-Lifschitz transform *of LP with respect to I, denoted LP^I is defined as follows:*

$$\begin{aligned} LP^I = \{ & a_1 \vee \ldots \vee a_m \leftarrow b_1, \ldots, b_k \mid \\ & a_1 \vee \ldots \vee a_m \leftarrow b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_n \in LP \\ & \text{and } \{b_{k+1}, \ldots, b_n\} \cap M = \emptyset \} \end{aligned}$$

Note that if LP is non-disjunctive, then LP^I is a Horn program.

Definition 8. *Every Horn program P has a least model* $\mathrm{lm}(P)$. *Given a logic program LP and an interpretation I, I is called a* stable model *iff* $I = \mathrm{lm}(LP^I)$. *The collection of all stable models of LP is denoted by* $\mathrm{STM}(LP) = \{I | I = \mathrm{lm}(LP^I)\}$.

Definition 9 ([7, p. 137])**.** Brave reasoning *(or* credulous reasoning*) infers that a literal Q is true in LP (denoted $LP \models^b_{st} Q$) iff Q is true with respect to M for* ***some*** $M \in \mathrm{STM}(LP)$.

Definition 10 ([7, p. 137])**.** Cautious reasoning *(or* skeptical reasoning*) infers that a literal Q is true in LP (denoted $LP \models^b_{st} Q$) iff (1) Q is true with respect to M for* ***all*** $M \in \mathrm{STM}(LP)$ *and (2)* $\mathrm{STM}(LP) \neq \emptyset$.

The well-founded semantics described by Gelder et al. [10] represent a progression from Clark completion [10, Section 1.1], discussed in Section 3. It involves three-valued models (adding $\perp$ as a "third, unknown truth value" to true and false) and so called *unfounded sets*. A definition is omitted in this work for brevity and to avoid complexity. The most important takeaway is that the well-founded semantics infers that a literal Q is true in LP, denoted $LP \models_{wf} Q$, iff it is true in the well-founded model, and that there is at most one such model (unlike with stable semantics, where there might be multiple stable models).

It is interesting to note here that in the choice of single model ($\vdash_{NF}$ as in Prolog, and $\models_{wf}$) vs. multiple model semantics (stable models in Answer Set Programming, and $\models^b_{st}$, $\models^c_{st}$ based on multiple models) for inference lies the reason for what is sometimes referred to as the *Great Logic Programming Schism* [17, Section 3, p. 13].

In [7] three important decision problems are highlighted: Given an LPAP $\mathscr{P} = \langle H, M, LP, \models \rangle$,

1. does there exist a solution for $\mathscr{P}$? (*consistency*),
2. is a given hypothesis $h \in H$ relevant for $\mathscr{P}$, i.e., does h contribute to some solution of $\mathscr{P}$? (*relevance*),
3. is a given hypothesis $h \in H$ necessary for $\mathscr{P}$, i.e., is h contained in all solutions of $\mathscr{P}$? (*necessity*).

They then continue to prove the complexity of solution verification ($S \in \mathrm{Sol}(\mathscr{P})$), consistency checking ($\mathrm{Sol}(\mathscr{P}) \neq \emptyset$) and $\preceq$-relevance, -necessity of some $h \in H$ for LPAPs and disjunctive LPAPs (LP disjunctive, see Definition 2) where preference $\preceq$ is either minimality with respect to inclusion ($\subseteq$), cardinality ($\leq$) or no preference ($=$) along all three inference operators $\models_{wf}$, $\models^b_{st}$, $\models^c_{st}$. The results are that these problems are complete in the lower end of the polynomial hierarchy ($\Delta^P_2[O(\log(n)]$ to Σ^P_4), and allow for some interesting transformations between abduction problems of different semantics.

5 Abduction in Argumentation

In this section we will take a closer look at the work of Booth et al. [2], which expanding on the definition of an argumentation framework (AF), introduces abductive argumentation frameworks (AAF).

First, given an AF $F = (A, \rightsquigarrow)$ we need to recognize how abduction can be expressed: An observation maps to a set of arguments $X \subseteq A$, "that each individually supports the observation" [2, p. 118] and the goal is finding an explanation, argumentation framework G that supports X.

Remark 2. *In the work of Booth et al. [2] there is no clear distinction of* Rule *(theory),* Result *(observation) and* Case *(explanation) (referring to the syllogisms from Section 1.1). Theory (which seems to be entangled with the observation) and explanation both are AFs while the observation is a set of arguments, but the connection is not clearly explained. Further, it is unclear what "supports the observation" means. They define an AF supporting an observation, but not an argument supporting an observation.*

In analogy to $\models^{c}_{st}$ and $\models^{b}_{st}$, Booth et al. [2] define skeptical and credulous support of an observation, based on the complete semantics.

Definition 11 ([2, Defintion 2])**.** *Let $F = (A, \rightsquigarrow)$. An extension of F is a set $E \subseteq A$. An extension E is conflict-free iff for no $a, b \in E$ it holds that $a \rightsquigarrow b$. An argument $a \in A$ is defended by E iff for all b such that $b \rightsquigarrow a$ there is a $c \in E$ such that $c \rightsquigarrow b$. Given an extension E, we define* $\mathrm{Def}_F(E)$ *by* $\mathrm{Def}_F(E) = \{a \in A \mid E \text{ defends } a\}$. *An extension E is admissible iff E is conflict-free and $E \subseteq \mathrm{Def}_F(E)$, and complete iff E is conflict-free and $E = \mathrm{Def}_F(E)$. The set of complete extensions of F will be denoted by* $\mathrm{Co}(F)$. *Furthermore, the grounded extension (denoted by* $\mathrm{Gr}(F)$*) is the unique $\subseteq$-minimal complete extension of F.*

Definition 12. *Given an AF $F = (A, \rightsquigarrow)$, an observation $X \subseteq A$ is skeptically (resp. credulously) supported iff for all (resp. some) $E \in \mathrm{Co}(F)$ it holds that $x \in E$ for some $x \in X$.*

Now it might be the case that an AF F does not skeptically or credulously support an observation X. In this case, the goal is to find a new AF G that in turn supports X. We arrive at the definition of an abductive argumentation framework.

Definition 13. *An* abductive argumentation framework *is a pair $M = (F, I)$ where F is an argumentation framework and and $I \subseteq \mathcal{F}$ a set of argumentation frameworks called* abducible *such that $F \in I$.*

As expected, an AAF $M = (F, I)$ skeptically (resp. credulously) explains an observation X if any $G \in I$ skeptically (resp. credulously) supports X. We continue to highlight the connection between F and $G \in I$ through logic programming.

For correspondence between AAFs and logic programs Booth et al. [2] rely on results by Wu et al. [19], whereby "a logic program LP over V can be transformed into an AF F such that the consequences of P under the partial stable semantics[3] can be computed by looking at the complete extensions of F. The idea is that an argument consists of a conclusion $C \in V$, a set of rules $R \subseteq LP$ used to derive C and a set $N \subseteq V$ of atoms that must be underivable in order for the argument to be acceptable" (quoting from [2], edited for the same notation as in Sections 3 and 4). Such a triple (C, R, N) represents an *instantiated argument* which is said to be *generated*[4] from LP and the set of arguments generated by LP is denoted A_{LP}. The attack relation generated by LP, denoted $\rightsquigarrow_{LP}$, is defined by $(C, R, N) \rightsquigarrow_{LP} (C', R', N')$. With this we have the established the link between AFs and logic programming: For a logic program LP over V, an atom $C \in V$ is a skeptical (resp. credulous) consequence of LP iff some $(C, R, N) \in A_{LP}$ is skeptically (resp. credulously) accepted in $(A_{LP}, \rightsquigarrow_{LP})$.

In the instantiated setting by Booth et al. [2] the definition of a hypothesis with respect to a logic program LP over V differs from hypotheses as per Definition 4

[3] Note that according to Przymusinski [16] this semantics coincides with the well-founded semantics that is part of the survey in [7] and very briefly mentioned in Section 4.

[4] A formal definition of a logic program *generating* an instantiated argument is omitted for brevity. See [2, Definition 13]

(and therefore from the framework defined by Eiter et al. [7]): Now we look at pairs (Δ^+, Δ^-) where Δ^+ and Δ^- are sets of abducibles (which in turn are in V). Δ^+ means added abducibles and complements the removed abducibles denoted Δ^-. Thus a hypothesis skeptically (resp. credulously) explains a query $Q \in V$ iff Q is a skeptical (resp. credulous) consequence of $(LP \cup \Delta^+) \setminus \Delta^-$. With this we can explain the connection between F and $G \in I$ for an AAF $F = (G, I)$.

Definition 14 ([2, Definition 16] edited for the same notations as in Sections 3 and 4)**.** *Given a logic program LP over V and a hypothesis (Δ^+, Δ^-), we denote by $F_{(\Delta^+,\Delta^-)}$ the AF $(A_{(LP\cup\Delta^+)\setminus\Delta^-}, \rightsquigarrow_{(LP\cup\Delta^+)\setminus\Delta^-})$. The AAF* generated by *LP is denoted by M_{LP} and defined by $M_{LP} = ((A_{LP}, \rightsquigarrow_{LP}), I_{LP})$, where $I_{LP} = \{F_{(\Delta^+,\Delta^-)} \mid \Delta^+, \Delta^- \subseteq V, \Delta^+ \cap \Delta^- = \emptyset\}$.*

Booth et al. [2] conclude that their model of abduction in argumentation is an abstraction of abductive logic programming.

6 Conclusion

While the main motivation of this work was the fascination for abduction, i.e. how a computer might attempt common-sense reasoning, a goal was also to deepen understanding of logic programming and to get behind how it can implement abduction. This fits a bigger picture towards combining symbolic (knowledge representation via logic programming) and non-symbolic perspectives (neural networks) on artificial intelligence, which we regard as an important field of research in the coming years.

6.1 Summary

Concerning the three preceding sections, key take-aways are that (1) abduction from logic programs, while this problem appears so intangible at first, can mapped to deduction in a straightforward manner, (2) complexities of important decision problems arising from abduction reside within the polynomial hierarchy, with interesting mappings between them, and (3) unsurprisingly argumentation can generalize abduction with a strong connection to logic programming.

We made an effort to introduce abduction, induction and deduction by example, in order to juxtapose abduction and deduction in Section 3. The connection between Section 3 and Section 4 is explicit in that we used the framework described in the latter for our definition and reasoning in the former. Further, the connection between Section 4 and Section 5 is twofold: (1) The notion of credulous and skeptical acceptance of an argument, and the truth of an atom under the credulous and skeptical semantics are strikingly familiar. (2) The close link between logic programs and argumentation frameworks (that even was highlighted in the initial work by Dung [6]) invites to reason about AAFs inside the framework defined by Eiter et al. [7].

6.2 Open Questions

The question on how logic programs generating AAFs (the result from Wu et al. [19] as discussed in Section 5) can be integrated into the framework by Eiter et al. [7] remains open: How can hypotheses of the form (Δ^+, Δ^-) (with $\Delta^+, \Delta^- \subseteq V$) on the one side be mapped to hypotheses of the form $H \subseteq V$ on the other side, where V is the set of atoms?

Also, the explanation dialogues detailed in [2, Section 4] were not touched on in this work. They constitute an interesting variant of dialogical proof using hypothetical moves, however the connection to logic programming is not straightforward.

References

[1] P. Baroni. An Introduction to Abstract Argumentation. http://www.epcl-study.eu/content/downloads/slides/baroni_2013_abstract_argumentation.pdf.

[2] R. Booth, D. M. Gabbay, S. Kaci, T. Rienstra, and L. W. N. van der Torre. Abduction and dialogical proof in argumentation and logic programming. In T. Schaub, G. Friedrich, and B. O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 117–122. IOS Press, 2014. ISBN 978-1-61499-418-3. doi: 10.3233/978-1-61499-419-0-117. URL http://dx.doi.org/10.3233/978-1-61499-419-0-117.

[3] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, 1977.*, Advances in Data Base Theory, pages 293–322, New York, 1977. Plemum Press. ISBN 0-306-40060-X.

[4] L. Console, D. T. Dupré, and P. Torasso. On the relationship between abduction and deduction. *J. Log. Comput.*, 1(5):661–690, 1991. doi: 10.1093/logcom/1.5.661. URL http://dx.doi.org/10.1093/logcom/1.5.661.

[5] I. Douven. Abduction. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2016 edition, 2016. https://plato.stanford.edu/archives/win2016/entries/abduction/.

[6] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995. doi: 10.1016/0004-3702(94)00041-X. URL https://doi.org/10.1016/0004-3702(94)00041-X.

[7] T. Eiter, G. Gottlob, and N. Leone. Abduction from logic programs: Semantics and complexity. *Theor. Comput. Sci.*, 189(1-2):129–177, 1997. doi: 10.1016/S0304-3975(96)00179-X. URL http://dx.doi.org/10.1016/S0304-3975(96)00179-X.

[8] T. Eiter, G. Ianni, and T. Krennwallner. Answer set programming: A primer. In Tessaris et al. [17], pages 40–110. ISBN 978-3-642-03753-5. doi: 10.1007/978-3-642-03754-2_2. URL http://dx.doi.org/10.1007/978-3-642-03754-2_2.

[9] K. Eshghi. Abductive planning with event calculus. In Kowalski and Bowen [12], pages 562–579. ISBN 0-262-61056-6.

[10] A. V. Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991. doi: 10.1145/116825.116838. URL http://doi.acm.org/10.1145/116825.116838.

[11] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In Kowalski and Bowen [12], pages 1070–1080. ISBN 0-262-61056-6.

[12] R. A. Kowalski and K. A. Bowen, editors. *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, August 15-19, 1988 (2 Volumes)*, 1988. MIT Press. ISBN 0-262-61056-6.

[13] J. Pearl. Embracing causality in formal reasoning. In K. D. Forbus and H. E. Shrobe, editors, *Proceedings of the 6th National Conference on Artificial Intelligence. Seattle, WA, July 1987.*, pages 369–373. Morgan Kaufmann, 1987. URL `http://www.aaai.org/Library/AAAI/1987/aaai87-066.php`.

[14] C. Peirce and J. Buchler. *Philosophical Writings of Peirce*. Dover books on philosophy. Dover Publications, 1940. ISBN 9780486202174. URL `https://books.google.at/books?id=Uq1kDQAAQBAJ`.

[15] C. S. Peirce. Illustrations of the Logic of Science VI: Deduction, Induction, and Hypothesis. *The Popular Science Monthly*, 13, 1878.

[16] T. C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundam. Inform.*, 13(4):445–463, 1990.

[17] S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh, M. Rousset, and R. A. Schmidt, editors. *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures*, volume 5689 of *Lecture Notes in Computer Science*, 2009. Springer. ISBN 978-3-642-03753-5. doi: 10.1007/978-3-642-03754-2. URL `http://dx.doi.org/10.1007/978-3-642-03754-2`.

[18] H. Tompits. Introduction to Knowledge Based Systems: Answer Set Programming. Lecture at Vienna University of Technology, 2016.

[19] Y. Wu, M. Caminada, and D. M. Gabbay. Complete extensions in argumentation coincide with 3-valued stable models in logic programming. *Studia Logica*, 93(2-3):383–403, 2009. doi: 10.1007/s11225-009-9210-5. URL `https://doi.org/10.1007/s11225-009-9210-5`.