Canonical LR(1) Parsers

Def: An *LR(1) item* is a two-component element of the form

$$[\, A \to \alpha \bullet \beta,\ \mu \,]$$

where the first component is a marked production, $A \to \alpha \bullet \beta$, called the *core* of the item and μ is a *lookahead* character that belongs to the set $V_t \cup \{ \lambda \}$.

An LR(1) item $[\, A \to \alpha \bullet \beta,\ \mu \,]$ is said to be valid for viable prefix γ if there exists a rightmost derivation

$$S \Rightarrow^{*}_{R} \phi A \tau \Rightarrow_{R} \phi \alpha \beta \tau$$

where $\gamma = \phi \alpha$ is the viable prefix and μ is the first symbol of τ or λ if $\tau = \lambda$.

Aside:

Consider the grammar

1. S → A d
2. | B e
3. A → a A b
4. | c
5. B → a B b
6. | c

Is this grammar LR(k) for some fixed k?

What about LL(k) for some fixed k?

Consider the grammar

1. S → A d
2. | A e
3. A → a A b
4. | c

Is this grammar LR(k) for some fixed k?

What about LL(k) for some fixed k?

Consider the grammar

1. $S \rightarrow A\ B$
2. $A \rightarrow a\ A\ b$
3. $\quad\ \ \mid c$
4. $B \rightarrow d$
5. $\quad\ \ \mid e$

Is this grammar LR(k) for some fixed k?

What about LL(k) for some fixed k?

LR(1) Item Construction

1. Generate the start state item set, C_0.

    1.1. Basis Set: $[\ S \rightarrow \bullet\ \alpha,\ \lambda\ ]$

    1.2. Closure Set: if $[\ A \rightarrow \bullet\ X\ \beta,\ \mu\ ]$ where $X \rightarrow \gamma \in P$, then add $[\ X \rightarrow \bullet\ \gamma,\ \nu\ ]$ to C_0 where $\nu \in FIRST_1(\beta\mu)$.

2. Do until no new states or item sets can be created

    2.1 *{Perform a read operation}*
    For each item $[\ A \rightarrow \alpha \bullet X\ \beta,\ c\ ]$ in some state U include the item $[\ A \rightarrow \alpha\ X \bullet \beta,\ c\ ]$ in a new state V. If the basis set already exists, the merge the two states.

    2.2 *{Close the new state}*
    For each item $[\ A \rightarrow \alpha \bullet X\ \beta,\ c\ ]$ in state V and every production $X \rightarrow \gamma \in P$, add $[\ X \rightarrow \bullet\ \gamma,\ d\ ]$ to V where $d \in FIRST_1(\beta c)$.

Consider the following grammar:

0. $G \rightarrow S$
1. $S \rightarrow E = E$
2. $\quad | \ f$
3. $E \rightarrow T$
4. $\quad | \ E + T$
5. $T \rightarrow f$
6. $\quad | \ T * f$

The LR(1) item sets are

C_0

Click here for the answer.

LR(1) Constructor

Given: LR(1) item sets, C_0, C_1, …, C_m, where C_0 is the start state and the states of the parser, 0, 1, …, m, we have the following algorithm for constructing the LR(1) parse tables F and G.

1. Repeat for each state i in the LR(1) item sets

   1.1 {Compute F}

   a. if $[A \rightarrow \alpha \bullet u \beta, c] \in C_i$ where $u \in V_t$ and there is a transition from C_i to C_j on u, then $F(i, u) \leftarrow$ **S**hift.

   b. if $[A \rightarrow \alpha \bullet, u] \in C_i$ where $A \rightarrow \alpha$ is the j^{th} production and $u \in V_t \cup \{ \lambda \}$, then $F(i, u) \leftarrow$ **R**educe **j**.

   c. if $[S \rightarrow \alpha \bullet, \lambda] \in C_i$, then $F(i, \lambda) \leftarrow$ **A**ccept

   d. Otherwise Error

   1.2 {Compute G}

   a. If there is a transition from C_i to C_j on A, then $G(i, A) \leftarrow j$.

   b. Otherwise Error

Note: State 0 is the start state of the parser.

From the partial LR(1) item sets below, we obtain the following partial LR(1) action table.

State	Basis Set	Closure Set	Next State or Reduce
0	[G → • S, λ]		17
		[S → • E = E, λ]	1
		[S → • f, λ]	3
		[E → • T, =:+]	2
		[T → • T * f, =:+:*]	2
		[T → • f, =:+:*]	3
		[E → • E + T, =:+]	1
1	[S → E • = E, λ]		4
	[E → E • + T, =:+]		5
2	[E → T •, =:+]		R3
	[T → T • * f, =:+:*]		11
3	[S → f •, λ]		R2
	[T → f •, =:+:*]		R5
4	[S → E = • E, λ]		6
		[E → • T, λ:+]	7
		[T → • f, λ:+:*]	8
		[T → • T * f, λ:+:*]	7
		[E → • E + T, λ:+]	6

Action F

	f	=	+	*	λ
0					
1					
2					
3					
4					

Click here for the answer.

From the partial LR(1) item sets below, we obtain the following partial next state table.

State	Basis Set	Closure Set	Next State or Reduce
0	[G → • S, λ]		17
		[S → • E = E, λ]	1
		[S → • f, λ]	3
		[E → • T, =:+]	2
		[T → • T * f, =:+:*]	2
		[T → • f, =:+:*]	3
		[E → • E + T, =:+]	1
1	[S → E • = E, λ]		4
	[E → E • + T, =:+]		5
2	[E → T •, =:+]		R3
	[T → T • * f, =:+:*]		11
3	[S → f •, λ]		R2
	[T → f •, =:+:*]		R5
4	[S → E = • E, λ]		6
		[E → • T, λ:+]	7
		[T → • f, λ:+:*]	8
		[T → • T * f, λ:+:*]	7
		[E → • E + T, λ:+]	6

Next state G

State	S	E	T	f	=	+	*
0							
1							
2							
3							
4							

Click here for the answer.

Using the LR(1) parse tables above, the LR parse configuration 5-tuple for f + f = f * f yields

N	S	α	A/π
-	0	f + f = f * f	-

Action F

	f	=	+	*	λ
0	S	.	.	.	.
1	.	S	S	.	.
2	.	R3	R3	S	.
3	.	R5	R5	R5	R2
4	S	.	.	.	.
5	S	.	.	.	.
6	.	.	S	.	R1
7	.	.	R3	S	R3
8	.	.	R5	R5	R5
9	.	R5	R5	R5	.
10	.	R4	R4	S	.
11	S	.	.	.	.
12	.	R6	R6	R6	.
13	S	.	.	.	.
14	.	.	R4	S	R4
15	S	.	.	.	.
16	.	.	R6	R6	R6
17	.	.	.	.	A

Next State G

	S	E	T	f	=	+	*
0	17	1	2	3	.	.	.
1	.	.	.	.	4	5	.
2	.	.	.	.	.	.	11
3	.	.	.	.	.	.	.
4	.	6	7	8	.	.	.
5	.	.	10	9	.	.	.
6	.	.	.	.	.	13	.
7	.	.	.	.	.	.	15
8	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.
10	.	.	.	.	.	.	11
11	.	.	.	12	.	.	.
12	.	.	.	.	.	.	.
13	.	.	14	8	.	.	.
14	.	.	.	.	.	.	15
15	.	.	.	16	.	.	.
16	.	.	.	.	.	.	.
17	.	.	.	.	.	.	.

Grammar

0. G → S
1. S → E = E
2. | f
3. E → T
4. | E + T
5. T → f
6. | T * f

Click here for the answer.

ANSWERS

The complete LR(1) item sets for the above grammar are

State	Basis Set	Closure Set	Next State or Reduce
0	[G → • S, λ]		17
		[S → • E = E, λ]	1
		[S → • f, λ]	3
		[E → • T, =:+]	2
		[T → • T * f, =:+:*]	2
		[T → • f, =:+:*]	3
		[E → • E + T, =:+]	1
1	[S → E • = E, λ]		4
	[E → E • + T, =:+]		5
2	[E → T •, =:+]		R3
	[T → T • * f, =:+:*]		11
3	[S → f •, λ]		R2
	[T → f •, =:+:*]		R5
4	[S → E = • E, λ]		6
		[E → • T, λ:+]	7
		[T → • f, λ:+:*]	8
		[T → • T * f, λ:+:*]	7
		[E → • E + T, λ:+]	6
5	[E → E + • T, =:+]		10
		[T → • f, =:+:*]	9
		[T → • T * F, =:+:*]	10
6	[S → E = E •, λ]		R1
	[E → E • + T, λ:+]		13
7	[E → T •, λ:+]		R3
	[T → T • * f, λ:+:*]		15
8	[T → f •, =:+:*]		R5
9	[T → f •, =:+:*]		R5
10	[E → E + T •, =:+]		R4
	[T → T • * f, =:+:*]		11
11	[T → T * • f, =:+:*]		12
12	[T → T * f •, =:+:*]		R6
13	[E → E + • T, λ:+]		14
		[T → • f, λ:+:*]	8

		[T → • T * f, λ:+:*]	14
14	[E → E + T •, λ:+] [T → T • * f, λ:+:*]		R4 15
15	[T → T * • f, λ:+:*]		16
16	[T → T * f •, λ:+:*]		R6
17	[G → S •, λ]		A

Return

Action F

State	f	=	+	*	λ
0	S	.	.	.	.
1	.	S	S	.	.
2	.	R3	R3	S	.
3	.	R5	R5	R5	R2
4	S	.	.	.	.
5	S	.	.	.	.
6	.	.	S	.	R1
7	.	.	R3	S	R3
8	.	.	R5	R5	R5
9	.	R5	R5	R5	.
10	.	R4	R4	S	.
11	S	.	.	.	.
12	.	R6	R6	R6	.
13	S	.	.	.	.
14	.	.	R4	S	R4
15	S	.	.	.	.
16	.	.	R6	R6	R6
17	.	.	.	.	A

Return

Next State G

State	S	E	T	f	=	+	*
0	17	1	2	3	.	.	.
1	.	.	.	.	4	5	.
2	.	.	.	.	.	.	11
3	.	.	.	.	.	.	.
4	.	6	7	8	.	.	.
5	.	.	10	9	.	.	.
6	.	.	.	.	.	13	.
7	.	.	.	.	.	.	15
8	.	.	.	.	.	.	.
9	.	.	.	.	.	.	.
10	.	.	.	.	.	.	11
11	.	.	.	12	.	.	.
12	.	.	.	.	.	.	.
13	.	.	14	8	.	.	.
14	.	.	.	.	.	.	15
15	.	.	.	16	.	.	.
16	.	.	.	.	.	.	.
17 .	.	.	.	.	.	.	.

Return

N	**S**	α	**A**	π
-	0	f + f = f * f	-	-
3	~~3~~ 0	+ f = f * f	S	-
2	~~2~~ 0	+ f = f * f	R	5
1	1 0	+ f = f * f	R	3
5	5 1 0	f = f * f	S	-
9	~~9~~ 5 1 0	= f * f	S	-
10	~~10 5 1~~ 0	= f * f	R	5
1	1 0	= f * f	R	4
4	4 1 0	f * f	S	-
8	~~8~~ 4 1 0	* f	S	-
7	7 4 1 0	* f	R	5
15	15 7 4 1 0	f	S	-
16	~~16 15 7~~ 4 1 0	λ	S	-
7	~~7~~ 4 1 0	λ	R	6
6	~~6 4 1~~ 0	λ	R	3
17	17 0	λ	R	1
			Accept	