SLR – "Simple LR"

An *LR(0) item* (*item* for short) of a grammar G is a production of G with a dot at some position of the right side. Thus, production $A \rightarrow XYZ$ yields the four items

$A \rightarrow \bullet XYZ$
$A \rightarrow X\bullet YZ$
$A \rightarrow XY\bullet Z$
$A \rightarrow XYZ\bullet$

The production $A \rightarrow \in$ generates only one item, $A \rightarrow \bullet$.

The central idea in the SLR method is first to construct from the grammar a deterministic finite automaton to recognize viable prefixes. We group items together in sets, which give rise to the states of the SLR parser. The items can be viewed as the states of an NFA recognizing viable prefixes, and the "grouping together" is really the subset construction discussed earlier.

To construct the canonical LR(0) collection for a grammar, we define an augmented grammar for and two functions, *closure* and *goto*.

If $G$is a grammar with start symbol S, then $G'$the augmented grammar for G is $G$with a new start symbol $S'$and production $S' \rightarrow S$. The purpose of this new starting production is to indicate to the parser when it should stop parsing and announce acceptance of the input. Acceptance occurs only when the parser is about to reduce by $S' \rightarrow S$

The Closure Operation

If $I$is a set of items for a grammar G then $closure(I)$ is the set of items constructed from I by the two rules:

1. Initially, every item in I is added to $closure(I)$
2. If $A \rightarrow \alpha\bullet B\beta$is in $closure(I)$ and $B \rightarrow \gamma$is a production, then add the item $B \rightarrow \bullet\gamma$ to I if it is not already there. We apply this rule until no more new items can be added to $closure(I)$.

Example 4.34

E'	→	*E*
E	→	*E* **+** *T*
E	→	*T*
T	→	*T* ***** *F*
T	→	*F*
F	→	**(** *E* **)**
F	→	**id**

If I is the set of one item $\{E' \rightarrow \bullet E\}$ then $closure(I)$ contains the items

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E + T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T * F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet \mathbf{id}$

Here, $E' \rightarrow \bullet E$ is put in $closure(I)$ by rule (1). Since there is an E immediately to the right of a dot, by rule (2) we add the *E*-productions with dots at the left end, that is,$E \rightarrow \bullet E + T$and $E \rightarrow \bullet T$ Now there is a T immediately to the right of a dot, so we add$T \rightarrow \bullet T * F$ and $T \rightarrow \bullet F$. and Next, the F to the right of a dot forces $F \rightarrow \bullet(E)$ and $F \rightarrow \bullet\mathbf{id}$ to be added. No other items are put into in $closure(I)$by rule (2).

The Goto Operation

The second useful function is *goto*(*I*,*X*) where *I* is a set of items and *X* is a grammar symbol. *goto*(*I*,*X*) is defined to be the closure of the set of all items $[A \rightarrow \alpha X \bullet B]$such that$[A \rightarrow \alpha \bullet XB]$ is in I. Intuitively, if I is the set of items that are valid for some viable prefix γ then *goto*(*I*,*X*) is the set of items that are valid for the viable prefix γX.

Example 4.35. If $I$is the set of two items $\{[E' \rightarrow E\bullet], [E \rightarrow E\bullet + T]\}$, then *goto*(*I*,+) consists of

$E \rightarrow E + \bullet T$
$T \rightarrow \bullet T * F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet \mathbf{id}$

We computed *goto*(*I*,+) by examining *I* for items with + immediately to the right of the dot. $E' \rightarrow E\bullet$ is not such an item, but $E \rightarrow E\bullet + T$ is. We moved the dot over the + to get $\{E \rightarrow E\bullet + T\}$ and then took the closure of this set.

Example 4.36. The canonical collection of sets of LR(0) items for the grammar of example 4.34 is shown in figure 1 below. The *goto* function is given in the transition diagram given in figure 2.

I_0: $E' \rightarrow \bullet E$
$E \rightarrow \bullet E + T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T * F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet \mathbf{id}$

I_1: $E' \rightarrow E \bullet$
$E \rightarrow E \bullet + T$

I_2: $E \rightarrow T \bullet$
$T \rightarrow T \bullet * F$

I_3: $T \rightarrow F \bullet$

I_4: $F \rightarrow (\bullet E)$
$E \rightarrow \bullet E + T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T * F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet \mathbf{id}$

I_5: $F \rightarrow \mathbf{id} \bullet$

I_6: $E \rightarrow E + \bullet T$
$T \rightarrow \bullet T * F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet \mathbf{id}$

I_7: $T \rightarrow T * \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet \mathbf{id}$

I_8: $F \rightarrow (E \bullet)$
$E \rightarrow E \bullet + T$

I_9: $E \rightarrow E + T \bullet$
$T \rightarrow T \bullet * F$

I_{10}: $T \rightarrow T * F \bullet$

I_{11}: $F \rightarrow (E) \bullet$

Figure 1. Canonical LR(0) collection for the grammar of Example 4.34

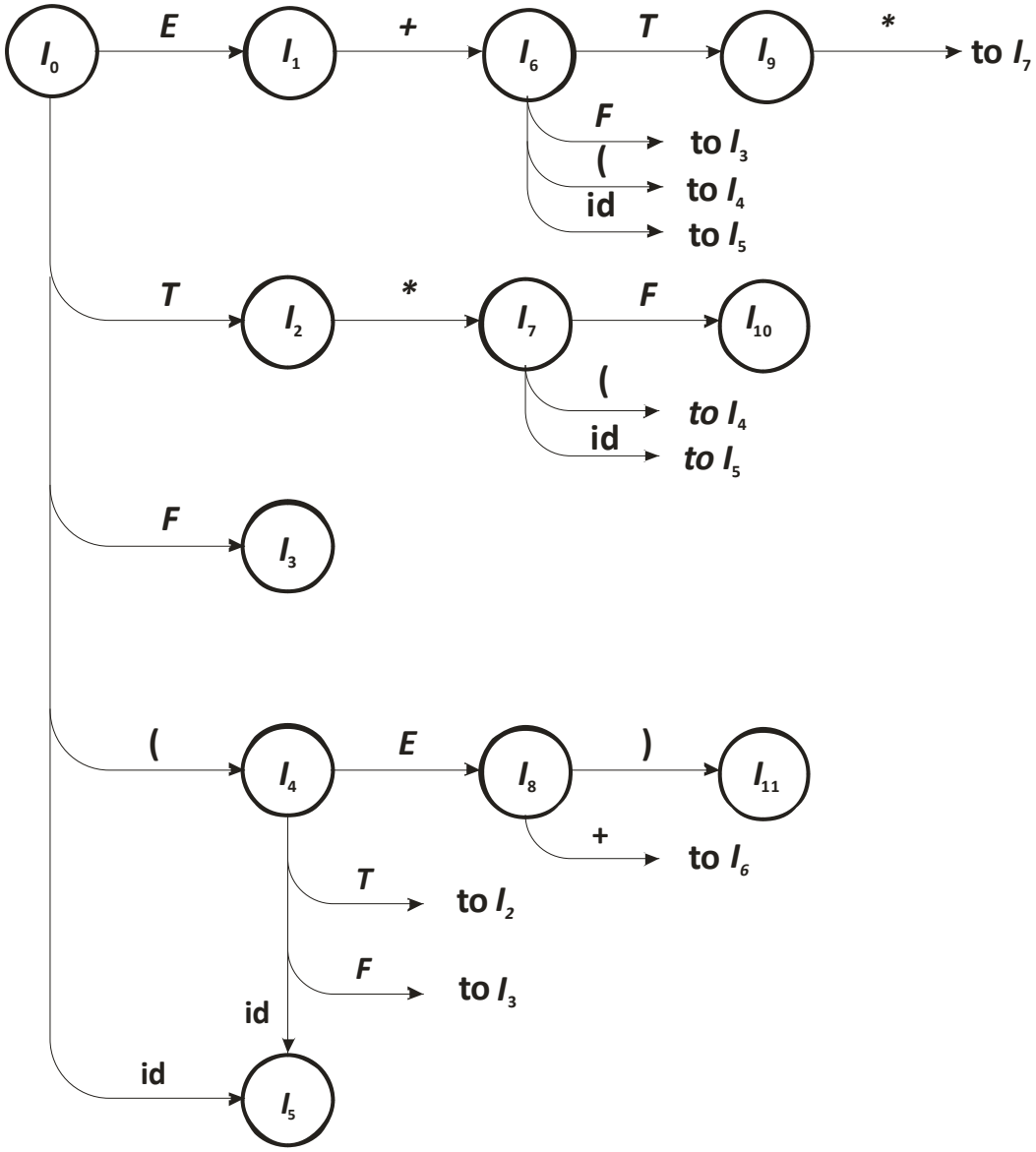

Figure 2. Transition diagram of DFA *D* for viable prefixes

Algorithm 4.8. Constructing an SLR parsing table.

Input: An augmented grammar G'.

Output. The SLR parsing table functions *action* and *goto* for G'.

Method.

1. Construct $C = \{I_0, I_1, \cdots, I_n\}$ the collection of sets of LR(0) items for G'.
2. State i Is constructed from I_i . The parsing actions for state i are determined as follows:
   a) If $[A \to \alpha \bullet a B]$ is in I_i and $goto(I_i, a) = I_j$ then set $action(i, a)$ to "shift j." Here a must be terminal.
   b) If$[A \to \alpha \bullet]$is in I_i then set $action(i, a)$ to "*reduce* $A \to \alpha$" for all a in $FOLLOW(A)$. Here Amay not be S'.
   c) If $[S' \to S \bullet]$ is in I_i then set $action(i, \$)$ to "*accept*"

If any conflicting actions are generated by the above rules, we say the grammar is not SLR(1). The algorithm fails to produce a parser in this case.

3. The $goto$ transitions for state i are constructed for all nonterminals A using the rule: if $goto(I_i, A) = I_j$, then $goto(I_i, A) = j$.
4. All entries not defined by rules (2) and (3) are made "error."
5. The initial state of the parser is the one constructed from the set of items containing$[S' \to \bullet S]$.

Example:

	left side		*right side*
1	*E*	→	*E* **+** *T*
2	*E*	→	*T*
3	*T*	→	*T* ***** *F*
4	*T*	→	*F*
5	*F*	→	**(** *E* **)**
6	*F*	→	**id**

Table 1. Set of productions expressions

STATE	ACTION						GOTO		
	id	**+**	*****	**(**	**)**	**$**	*E*	*T*	*F*
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			