Building the Shift–Reduce Table

We show how to construct the table for bottom-up parsing provided the LR(1) grammar has the right form.

Construction of SLR(1) Table

We now describe how to construct the table used in bottom-up parsing of LR(1) grammars.

This process has two main phases:

(1) determining the states and the shifts, and

(2) determining the reductions.

Illustrative Grammar

We use the arithmetic grammar as an example.

1: $E \rightarrow E + T$
2: $E \rightarrow T$
3: $T \rightarrow T \times F$
4: $T \rightarrow F$
5: $F \rightarrow (E)$
6: $F \rightarrow \mathsf{n}$

Items

The key construct is an ***item***: a production with a certain point on RHS marked. We use a dot as the ***marker*** and enclose whole item in square brackets.

For example, item $[E \to E\texttt{+} \cdot T]$ is production $E \to E\texttt{+}T$ with a marker before the T.

If production has RHS of length n, then there are $n + 1$ corresponding items.

An ***initial*** item has marker at the left end; a ***completed*** item has marker at the right end.

Item-Sets

The intended meaning of an item is that the symbols to the left of the marker have already been read, or reduced to, and they are the top of the stack.

For our parser,

> *each state is a set of items, called an* ***item-set****.*

(Note that if construction goes wrong at any point, that is a proof that the grammar is not in right form.)

The Next Item

To determine the item that comes after an item:

1. *Advance the marker.* So if item is $[E \to \cdot\,(\,E\,)]$ and next symbol is $($, we go to item $[E \to (\,\cdot\,E\,)]$.
2. *Take the closure.* For an item, the ***closure*** is set of items as follows: If marker precedes a variable, then closure adds all initial items starting with that variable. If any new item has marker preceding a variable, then add all initial items with that variable. And repeat. (If marker precedes a terminal, there is nothing to add.)

Example Next Item

For the arithmetic grammar, consider the item $[T \to T \cdot \times F]$.

On the symbol $\times$, the result is $[T \to T \times \cdot F]$.

The closure adds all initial items beginning with F: that is, $[F \to \cdot (\, E \,)]$ and $[F \to \cdot \mathsf{n}]$.

The Next Item-Set

Determining the next item-set. *Let Q be an item-set and σ a symbol. The next item-set is constructed by:*

For each item in Q where marker precedes σ:

1. Add item that results from advancing marker past σ.

2. Add closure of resultant item.

The Start Item-Set

We add new start variable to CFG with its only production taking it to the old start variable.

The start item-set is the initial item for the (new) start variable, and its closure.

For the example CFG: The start item is $[E' \rightarrow \cdot E]$. Its closure adds all initial items starting with E, which adds all initial items starting with T, which adds all initial items starting with F:

$\{ [E' \rightarrow \cdot E], [E \rightarrow \cdot E + T], [E \rightarrow \cdot T], [T \rightarrow \cdot T \times F],$
$[T \rightarrow \cdot F], [F \rightarrow \cdot (\ E\ )], [F \rightarrow \cdot \mathsf{n}] \}.$

Determining States and Shifts in Table

We now determine the states and the shifts.

Start with the start item-set. Consider every possible next symbol: terminal and variable. For each symbol, determine the next item-set. Then for each item-set so created, do the same thing. Repeat until no new item-set is created.

Here is the calculation of item-sets for the arithmetic CFG:

State	first occurs	Items found by moving marker	items added
0	start	$[E' \rightarrow \cdot E]$	E, T, F
1	0, (	$[F \rightarrow (\cdot E)]$	E, T, F
2	0, n	$[F \rightarrow \mathsf{n} \cdot]$	
3	0, E	$[E' \rightarrow E \cdot]$, $[E \rightarrow E \cdot + T]$	
4	0, T	$[E \rightarrow T \cdot]$, $[T \rightarrow T \cdot \times F]$	
5	0, F	$[T \rightarrow F \cdot]$	
6	1, E	$[F \rightarrow (E \cdot)]$	
7	3, +	$[E \rightarrow E + \cdot T]$	T, F
8	4, $\times$	$[T \rightarrow T \times \cdot F]$	F
9	6,)	$[F \rightarrow (E) \cdot]$	
10	7, T	$[E \rightarrow E + T \cdot]$, $[T \rightarrow T \cdot \times F]$	
11	8, F	$[T \rightarrow T \times F \cdot]$	

Example: Constructing the Table

Each of the above transitions produces a shift in the table.

For example, suppose in state 3 one considers the transition for +. This defines a new item-set that produces state 7. A shift s7 is added to the table.

Determining the Reductions

Determining the reductions seems obvious: take each completed item and make that a reduction.

But, all our calculations are trying to answer: *when to reduce and when not to.*

FOLLOW *sets*

We need FOLLOW sets. For variable A, FOLLOW(A) is the set of all possible terminals that can follow A in a string derived from S.

One includes eos symbol Δ in FOLLOW(A) if variable A can occur at the end of a derived string.

For example, if $S \stackrel{*}{\Longrightarrow} BC\mathsf{x}BA\mathsf{y}\mathsf{x}C$, then one adds y to FOLLOW(A) and Δ, x to FOLLOW(C).

Example: FOLLOW *sets*

In the arithmetic grammar, $\mathrm{FOLLOW}(E)$ is $\{), \Delta, +\}$.
Both $\mathrm{FOLLOW}(T)$ and $\mathrm{FOLLOW}(F)$ are $\{), \Delta, +, \times\}$.

For instance, $E' \stackrel{*}{\Longrightarrow} E \stackrel{*}{\Longrightarrow} T \stackrel{*}{\Longrightarrow} T \times F \stackrel{*}{\Longrightarrow} F \times F$ shows that $\times \in \mathrm{FOLLOW}(F)$.

Adding the Reductions to the Table

For each completed item, say an A-item, add reduction to table for all symbols that occur in $\mathrm{FOLLOW}(A)$. Except that, the reduction corresponding to artificial production $E' \rightarrow E$ becomes the acceptance point.

Whew!

This process sounds rather daunting. Fortunately there is software to do it. The UNIX utility YACC generates LR tables for certain CFGs.

Practice

For the earlier grammar

1: $S \rightarrow \mathsf{r}\, L$
2: $L \rightarrow L\, \mathsf{,}\, I$
3: $L \rightarrow I$
4: $I \rightarrow \mathsf{v}$

(a) Verify that $\mathrm{FOLLOW}(S) = \{\Delta\}$ and $\mathrm{FOLLOW}(L) = \mathrm{FOLLOW}(I) = \{\mathsf{,}, \Delta\}$.

(b) Determine the item-sets.

(c) Verify the entries in the table earlier (repeated next).

The Table Again

State	*Progress*	r	,	v	eos	S	I	L
0		s1				2		
1	r			s3			4	5
2	S				*acc*			
3	v		R4		R4			
4	I		R3		R3			
5	r L		s6		R1			
6	L ,			s3			7	
7	L , I		R2		R2			

Solution to Practice

(b)

State	Contents of item-set
0	$[S' \rightarrow \cdot\, S], [S \rightarrow \cdot\, \mathsf{r}\, L]$
1	$[S \rightarrow \mathsf{r} \cdot L], [L \rightarrow \cdot\, L\,\mathsf{,}\, I], [L \rightarrow \cdot\, I], [I \rightarrow \cdot\, \mathsf{v}]$
2	$[S' \rightarrow S \cdot]$
3	$[I \rightarrow \mathsf{v} \cdot]$
4	$[L \rightarrow I \cdot]$
5	$[S \rightarrow \mathsf{r}\, L \cdot], [L \rightarrow L \cdot\, \mathsf{,}\, I]$
6	$[L \rightarrow L\,\mathsf{,} \cdot\, I], [I \rightarrow \cdot\, \mathsf{v}]$
7	$[L \rightarrow L\,\mathsf{,}\, I \cdot]$

Summary

The states of the parser correspond to item-sets, generated by trying all possible next terminals. The reductions occur for completed items based on FOLLOW sets.