CS 421 — Generating the LR Parsing Tables

Mattox Beckman

Introduction

This is a summary of how to build the LR item sets and tables. It is a bit terse, written on the assumption that you've seen this before but would like a summary.

An LR parser works by collecting input and building parse trees from the "bottom up." As input is consumed, the parser uses the tokens as leaves of the tree, combining them to build larger trees until the tree is complete. The purpose of this guide is to show you how to build the LR parser tables.

We will start by giving some definitions.

A grammar $G = (N, \Sigma, P)$ is a set N of terminal symbols, a set Σ of non-terminal symbols, and a set P of productions. A production has the form $\Sigma \to (\Sigma \cup N)*$, that is, a nonterminal followed by an arrow, followed by zero or more symbols that are either terminal or nonterminal. In this guide we will use capital letters to indicate nonterminals, and lower-case letters for terminals. The special terminal symbol ϵ indicates the absence of input, and the special terminal symbol $ indicates end of input. (Note that $ never occurs in a production. It is used by the parser.)

We will call a production with a symbol E on the left hand side an *E-production*.

$$\begin{aligned} S &\to E \\ E &\to ExE \\ &\mid z \end{aligned}$$

Figure 1: An example grammar with three productions. E and S are nonterminals, and x and z are terminals.

Building the Item Sets

An *item* is a production with an extra annotation. A single symbol $\bullet$ indicating a "cursor" is added to the symbols on the right hand side, and can appear in any position.

The production $E \to ExE$ has four possible items:

$E \to \bullet ExE$
$E \to E \bullet xE$
$E \to Ex \bullet E$
$E \to ExE\bullet$

The item $E \to \bullet ExE$ is the *initial item* for the production $E \to ExE$. The *initial items* for a nonterminal E is the set of initial items for all E-productions in the grammar. Similarly, $E \to ExE\bullet$ is a *final item* for that production.

Intuitively, the $\bullet$ indicates what part of a the production has been read or produced from the input.

An LR parser is a push-down automata that is driven by its input. It uses its stack to keep track of input already consumed as well as subtrees already built, as well as the path the machine took to get to the current state.

There are three actions the machine can take.

- shift — consume input and push it onto the stack

- reduce — pop off zero or more elements from the stack, combining them into a single tree. Each element popped off the stack causes the machine to unwind one step. The newly constructed tree is pushed onto the stack. The reduce action specifies which production is reduced.
- accept — signal a successful parse

These actions are kept in an action table. There is one row for each state, and one column for each non-terminal, including $\$$.

After each action, the machine will transition to another state. These are kept in the goto table, which has one row for each state, and one column for each symbol, both terminals and nonterminals.

The states are represented using *item sets*, one or more items.

We will often need to take the *closure* of an item set. If the item set contains an item with the cursor just to the left of some nonterminal N, we add to the item set all initial items for the N-productions of the grammar, recursively. (I.e., if adding new items introduces more items with a cursor in front of a nonterminal, we add those initial items as well.)

Here, now, is the algorithm for generating the item sets.

1. Augment the grammar with a production $S' \to S$, where S is the start symbol of your grammar.
2. To generate the first item set I_0, take the closure of the item $S' \to \bullet S$. In class, we have tended to drop this item from the item set.
3. Stating with $n = 0$: For each item i in state n:
   (a) For each terminal symbol α of the grammar, take *all* the items in state n containing a cursor in front of α and create a new item set where the cursor has moved to the right of α. Number this new state n', and take its closure. If an identical state already exists, use that state number for n' instead of creating a duplicate.

   Enter a shift action in row n and column α, and enter n' in the corresponding entry of the goto table.

   (b) Similarly, for each nonterminal A of the grammar, take *all* the items in state n containing a cursor in front of A and create a new item set where the cursor has moved to the right of A. Number this new state n', and take its closure. If an identical state already exists, use that state number for n' instead of creating a duplicate.

   Enter n' in the goto table for row n and nonterminal A.

   (c) For every final A-item for production P in this state, enter a reduce final item for this production in row n of your action table, for each column in the follow set of A.

Shift/Reduce Conflicts

If your table contains an entry with both a shift and a reduce, then you have a shift-reduce conflict. This indicates that your grammar is ambiguous. You can see this in your item sets: you will have a final item for some nonterminal A (which causes reduce actions to be added to your table), and another item where the cursor precedes a nonterminal α that is in the follow set of A (which adds a shift action on top of one of the reduce actions.)

1 Example

Consider the grammar from figure 1.

State 0 We start with the item $S' \rightarrow \ \bullet S$

Taking its closure, we get

$S' \rightarrow \ \bullet S$

$S \rightarrow \ \bullet E$

$E \rightarrow \ \bullet ExE$

$\quad\quad | \ \bullet z$

Taking the $S' \rightarrow \bullet S$ item, we create state 1. We enter a 1 in the goto table.

We use $S \rightarrow \bullet E$ and $E \rightarrow \bullet ExE$ to create state 2, and enter a 2 for the E column in the goto table.

We use $E \rightarrow \bullet z$ to create state 3, and enter a shift in the action table, and a 2 in the goto table.

Our tables now look like this:

	Action			Goto				
State	x	z	\$	x	z	\$	S	E
0		s			3		1	2

State 1 State 1 looks like this.

$S' \rightarrow \ S\bullet$

This is a final item, so we enter a reduce action in the table. We annotate is as $r1$, since $S' \rightarrow S$ is the first production in our grammar. The follow set of S' is $\{\$\}$, so we enter it in that column.

Our table looks like this now.

	Action			Goto				
State	x	z	\$	x	z	\$	S	E
0		s			3		1	2
1			r1					

State 2 Here is state 2.

$S \rightarrow \ E\bullet$

$E \rightarrow \ E \bullet xE$

The S item is a final one, so we need a reduce action in column \$. The E item will create a new state 4. Here is our table now.

	Action			Goto				
State	x	z	\$	x	z	\$	S	E
0		s			3		1	2
1			r1					
2	s		r2	4				

State 3 Here is state 3.

$E \rightarrow \ z\bullet$

This is a final item for $E \rightarrow z$, so we enter a reduce 4 action for columns $\{x, \$\}$, the follow set of E.

Here is our table now.

	Action			Goto				
State	x	z	\$	x	z	\$	S	E
0		s			3		1	2
1			r1					
2	s		r2	4				
3	r4		r4					

State 4 Here is state 4. Notice we had to add the initial E items.

$E \rightarrow \ Ex \bullet E$

$E \rightarrow \ \bullet ExE$

$\quad\quad | \ \bullet z$

The $E \to Ex \bullet E$ and $E \to \bullet ExE$ items will create a new state 5 with corresponding entry in the goto table.
The $E \to \bullet z$ recreates state 3, so we recycle it.
Here is our table.

	Action			Goto				
State	x	z	\$	x	z	\$	S	E
0		s			3		1	2
1			r1					
2	s		r2	4				
3	r4		r4					
4		s			3			5

State 5 Here is state 5.

$E \to ExE\bullet$
$E \to E \bullet xE$

Notice we have a final item for E, and another item with the $\bullet$ in front of x, which is part of E's follow set. Here is a shift/reduce conflict!

We put a reduce in the appropriate columns. Shifting x gives us a state identical to state 4, so we reuse it.
Our table will look like this.

	Action			Goto				
State	x	z	\$	x	z	\$	S	E
0		s			3		1	2
1			r1					
2	s		r2	4				
3	r4		r4					
4		s			3			5
5	s/r3		r3	4				

This is the whole table.
Here is the full listing of the item sets for this grammar.

I_0 $S' \to \bullet S$
$S \to \bullet E$
$E \to \bullet ExE$
$\mid \bullet z$
I_1 $S' \to S\bullet$
I_2 $S \to E\bullet$
$E \to E \bullet xE$
I_3 $E \to z\bullet$
I_4 $E \to Ex \bullet E$
$E \to \bullet ExE$
$\mid \bullet z$
I_5 $E \to ExE\bullet$
$E \to E \bullet xE$