CS 241 – Week 7 Tutorial

Bottum-up Parsing

Spring 2018

1 Definitions

- A *bottom-up* parser begins with the input string and repeatedly replaces substrings of that string with nonterminals which derive them until the input string has been entirely replaced by the start symbol.
- LR(0) stands for *Left-to-right* scan of input, *Rightmost* derivations, 0 tokens of lookahead.
- An *item* is a production with a dot (•) somewhere on the right-hand side, and epresents a partially-completed rule.
- A *shift-reduce conflict* occurs when a state in the LR machine has both completed and partially completed rules and we cannot determine whether to reduce or shift.
- A *reduce-reduce conflict* occurs when a state in the LR machine has two different completed rules and we cannot determine which rule to reduce by.
- A grammar is LR(0) if its corresponding machine has no conflicts.
- An *SLR(1) parser* (short for Simplified LR(1)) is an LR(0) parser where every finished rule is also given the follow set of its left-hand side as lookaheads. This helps drastically reduce the number of shift-reduce or reduce-reduce conflicts, since there is now only a conflict if the two possibilities (either shifting and reducing or reducing with two different rules) both occur in the same state and also share a common lookahead symbol. A grammar is SLR(1) if it can be parsed by an SLR(1) parser.

There are other LR(1)-family algorithms, notably LR(1) and LALR(1), which we do not cover in this course. The only difference between these algorithms is how the machine is constructed: the algorithm that uses the machine is the same in every case.

2 Problems

2.1 Machine construction

2.1.1 Non-LR(0) grammars

Show each of the following grammars are not LR(0) by constructing enough of the automaton to identify an issue:

1.

$$S \to aSb \tag{0}$$
$$S \to a \tag{1}$$

2.

$$S \to Xab \quad (0)$$
$$S \to Ycd \quad (1)$$
$$X \to e \quad (2)$$
$$Y \to e \quad (3)$$

2.1.2 SLR(1) machine construction

Construct a SLR(1) machine for the following grammar:

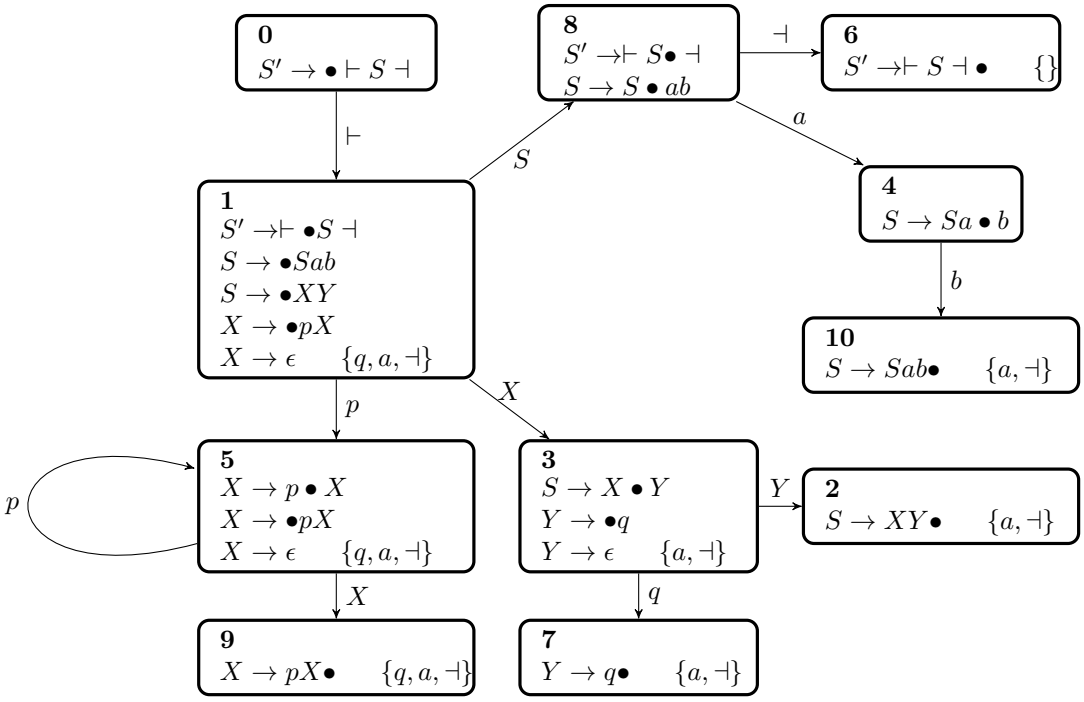
$$S' \to \vdash X \dashv \quad (0)$$
$$X \to XbAb \quad (1)$$
$$X \to \epsilon \quad (2)$$
$$A \to Aa \quad (3)$$
$$A \to \epsilon \quad (4)$$

2.2 Parsing

Consider the following context-free grammar and corresponding SLR(1) shift/reduce machine, given in both bubble diagram and table form:

$$S' \to \vdash S \dashv \quad (0)$$
$$S \to Sab \quad (1)$$
$$S \to XY \quad (2)$$
$$X \to pX \quad (3)$$
$$X \to \epsilon \quad (4)$$
$$Y \to q \quad (5)$$
$$Y \to \epsilon \quad (6)$$

2.2.1 SLR(1) Machine Bubble Diagram



Note: the state numbers have nothing to do with the rule numbers

2.2.2 SLR(1) Machine Table

For each set of three columns, the left represents the state number, the middle the next input symbol, the last the action to take. If the action is "shift x", x denotes the state to move to. If the action is "reduce y", y denotes the rule to reduce by.

0	$\vdash$	shift 1	2	a	reduce 2	5	a	reduce 4	8	a	shift 4
1	a	reduce 4	2	$\dashv$	reduce 2	5	p	shift 5	8	$\dashv$	shift 6
1	p	shift 5	3	a	reduce 6	5	q	reduce 4	9	a	reduce 3
1	q	reduce 4	3	q	shift 7	5	$\dashv$	reduce 4	9	q	reduce 3
1	$\dashv$	reduce 4	3	$\dashv$	reduce 6	5	X	shift 9	9	$\dashv$	reduce 3
1	S	shift 8	3	Y	shift 2	7	a	reduce 5	10	a	reduce 1
1	X	shift 3	4	b	shift 10	7	$\dashv$	reduce 5	10	$\dashv$	reduce 1

2.2.3 Exercises

1. Is the grammar LR(0)?
2. Use the shift/reduce table and grammar given above to parse the string $\vdash pqab \dashv$.
3. Write the reversed rightmost derivation for the string, as well as the parse tree.