CS 241 – Week 9 Tutorial

Bottom-Up Parsing: LR Parsing

Spring 2015

Definitions

- LR(1) stands for *Left-to-right* scan of input, *Right-canonical* derivation, *1* symbol of lookahead.
- An LR(1) parser is a *bottom-up* parser; it begins with the input string and finds a *reversed* derivation of the input, ending at the start symbol.
- An LR(1) grammar is a grammar that can be parsed by an LR(1) parser.
- An *item* is a production with a dot ($\bullet$) somewhere on the RHS (indicating a partially completed rule)
- A *shift-reduce* conflict occurs when we have a state in our LR transducer that has partially completed and completed rules
- A *reduce-reduce* conflict occurs when we have a state in our LR transducer that has two completed rules in an accepting state (e.g. we could reduce by two different rules).

SLR(1) Algorithm

```
push q_0
for each a in ⊢ input ⊣
  while (Reduce[stack.top, a] = A → γ)
    pop 2 * |γ| times
    state ← stack.top
    push A
    push Trans[state, A]
  state ← stack.top
  if (Trans[state, a] = ERROR) reject
  push a
  push Trans[state, a]
accept
```

Bottom-up Parsing (SLR(1))

Consider the following context-free grammar and SLR(1) shift/reduce table:

0. $S' \to \vdash S \dashv$
1. $S \to Sab$
2. $S \to XY$

3. $X \to pX$
4. $X \to \epsilon$
5. $Y \to q$
6. $Y \to \epsilon$

- Create an LR(0) machine for this grammar. Is the grammar LR(0)?
- Using the shift/reduce table, parse the string $\vdash$pqab$\dashv$. Write the reversed right-canonical derivation for the string, as well as the parse tree.

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