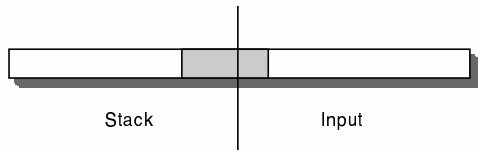
Notes on Canonical LR(1) Table Construction

Computer Science 426

August 30, 2007

Elements of Table Construction

1. *LR(1) items:* Describe configuration at boundary of stack and input.



2. *States:* A state is a collection of LR(1) items that describes *all legal configurations* that may occur at a give point in the parse.
   - If we are in state S_i, configuration I must hold for some item $I \in S_i$. Otherwise, it's a parse error.
   - Which configuration holds will tell us what transition to do.
3. *State graph:* Set of states and transitions (how to get from one state to another).
4. *Parse table:* A representation of the graph.

LR(1) Items

- Anatomy of an LR(1) item: $[A \rightarrow \alpha \cdot \beta, a]$
  - $A \rightarrow \alpha\beta$ is a production
  - A is a nonterminal symbol
  - α, β are strings of symbols (terminal or nonterminal)
  - a is a terminal symbol (the *lookahead item*)
  - $\cdot$ represents the stack-input boundary
- What it means
  - α appears on the top of the stack

- A terminal string derivable from βa appears at the front of the input *unless* $\beta = B\beta'$ for some nonterminal B, in which case we have just recognized B, and a terminal string derivable from $\beta' a$ appears at the front of the input.

- How many LR(1) items are there for $G = (N, \Sigma, P, S)$?

States

- A state is a set of LR(1) items. How many states are there?
- Each state S_i must conform to the following rule. Why?
  - If $[A \to \alpha \cdot B\beta', a] \in S_i$, then $[B \to \cdot\beta'', b] \in S_i$, for all $B \to \beta'' \in P$ and $b \in \Sigma$ appearing first in strings derived from $\beta' a$.
- For a set of LR(1) items I, *closure*(I) is the set formed by adding LR(1) items until the previous rules hold.

Constructing the State Graph

We assume that there is only one production from the start symbol, $S \to A$. Any grammar can be easily modified to conform to this requirement.

1. The initial state S_0 is *closure*$([S \to \cdot A, \text{EOF}])$.
2. For each terminal or nonterminal symbol σ, let J_σ bet the set of LR(1) items in S_0 such that σ appears to the right of the dot ($[A \to \alpha \cdot \sigma\beta, a]$), and let J'_σ be the set of items formed by moving σ in each item in J_σ to the left of the dot ($[A \to \alpha\sigma \cdot \beta, a]$). For each nonempty J_σ, add a transition on σ from S_0 to *closure*(J'_σ).
   - Each transition represents the state change when we push symbol σ onto the stack.
   - If σ is a terminal symbol, the transition is a shift, and if σ is a nonterminal symbol, the transition is a reduce.
   - Each transition may add a new state to the graph
3. Compute the transitions out of any new states, adding further states as necessary
4. Repeat step 3 until there are no more new states

Constructing the Parse Table

For each state S_i in the state graph, do the following in row S_i of the table:

1. If $[A \to \alpha \cdot a\beta, b] \in S_i$, add `shift` S_j to column a of the action table, where S_j is the state to which S_i transitions on a.
2. If $[A \to \alpha\cdot, a] \in S_i$, add `reduce` $A \to \alpha$ to column a of the action table.
3. If $[S \to A\cdot, \text{EOF}] \in S_i$, add `accept` to column `EOF` of the action table.
4. For all transitions S_i to S_j on a nonterminal A, add S_j to column A of the goto table.
5. All remaining table entries are —.