Script **generated by TTT**

Title: Petter: Compiler Construction (28.05.2020) - 18: Shift Reduce Parser

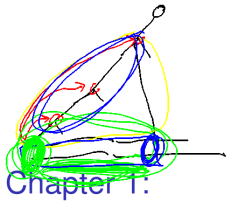
Date: Wed May 06 14:27:00 CEST 2020

Duration: 26:07 min

Pages: 6

Topic:

Syntactic Analysis - Part II



Chapter 1:

Bottom-up Analysis

Shift-Reduce Parser

Example:

$$\begin{array}{lcl} S & \rightarrow & A\,B \\ A & \rightarrow & a \\ B & \rightarrow & b \end{array}$$

The pushdown automaton:

States: q_0, f, a, b, A, B, S;
Start state: q_0
End state: f

q_0	a	$q_0\,a$
a	ϵ	A
A	b	$A\,b$
b	ϵ	B
$A\,B$	ϵ	S
$q_0\,S$	ϵ	f

Shift-Reduce Parser

Construction:

In general, we create an automaton $M_G^R = (Q, T, \delta, q_0, F)$ with:

- $Q = T \cup N \cup \{q_0, f\}$ (q_0, f fresh);
- $F = \{f\}$;
- Transitions:

$$\begin{array}{rcll} \delta & = & \{(q, x, q\,x) \mid q \in Q, x \in T\} \cup & // \text{ Shift-transitions} \\ & & \{(\alpha, \epsilon, A) \mid A \to \alpha \in P\} \cup & // \text{ Reduce-transitions} \\ & & \{(q_0\,S, \epsilon, f)\} & // \text{ finish} \end{array}$$

Shift-Reduce Parser

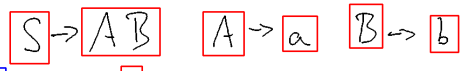
Construction:

In general, we create an automaton $M_G^R = (Q, T, \delta, q_0, F)$ with:

- $Q = T \cup N \cup \{q_0, f\}$ (q_0, f fresh);
- $F = \{f\}$;
- Transitions:

$$\begin{array}{rcll} \delta & = & \{(q, x, q\,x) \mid q \in Q, x \in T\} \cup & // \text{ Shift-transitions} \\ & & \{(\alpha, \epsilon, A) \mid A \to \alpha \in P\} \cup & // \text{ Reduce-transitions} \\ & & \{(q_0\,S, \epsilon, f)\} & // \text{ finish} \end{array}$$

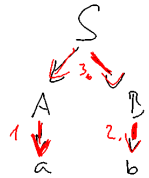
Example-computation:



$$\begin{array}{rcl} (q_0, & a\,b) & \vdash (q_0\,a, \quad b) \\ & \vdash (q_0\,A, & b) \\ & \vdash (q_0\,A\;b, & \epsilon) \\ & \vdash (q_0\,A\,B, & \epsilon) \\ & \vdash (q_0\,S, & \epsilon) \\ & \vdash (f, & \epsilon) \end{array}$$

Shift-Reduce Parser

Observation:



- The sequence of reductions corresponds to a reverse rightmost-derivation for the input
- To prove correctnes, we have to prove:

$$(\epsilon, w) \vdash^* (A, \epsilon) \quad \text{iff} \quad A \to^* w$$

- The shift-reduce pushdown automaton M_G^R is in general also non-deterministic
- For a deterministic parsing-algorithm, we have to identify computation-states for reduction

$\Longrightarrow$ LR-Parsing