Script generated by TTT

Title: Petter: Compilerbau (08.06.2017)

Date: Thu Jun 08 14:15:09 CEST 2017

Duration: 97:53 min

Pages: 26

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

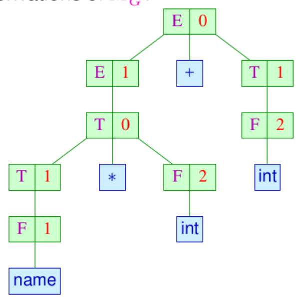
$\Longrightarrow$ LR-Parsing

Reverse Rightmost Derivations in Shift-Reduce-Parsers

Idea: Observe *reverse rightmost*-derivations of M_G^R!

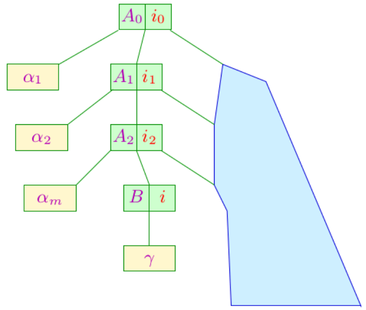
Input:
counter $* \; 2 + 40$

Pushdown:
$(\; q_0 \;)$



Bottom-up Analysis: Viable Prefix

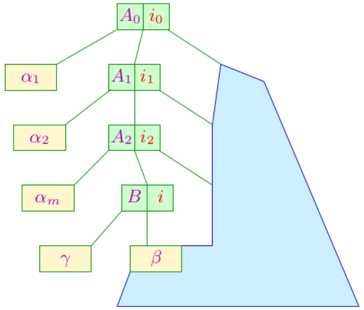
$\alpha\,\gamma$ is viable for $[B \to \gamma \bullet]$ iff $S \to_R^* \alpha\, B\, v$



... with $\alpha = \alpha_1 \ldots \alpha_m$

Bottom-up Analysis: Admissible Items

The item $[B\to\gamma\bullet\beta]$ is called admissible for α' iff $S\to^*_R\alpha\,B\,\omega$ with $\alpha' = \alpha\,\gamma$:



... with $\alpha = \alpha_1 \ldots \alpha_m$

Characteristic Automaton

Observation:
The set of viable prefixes from $(N\cup T)^*$ for (admissible) items can be computed from the content of the shift-reduce parser's pushdown with the help of a finite automaton:

States: Items
Start state: $[S'\to\bullet\,S]$
Final states: $\{[B\to\gamma\bullet]\mid B\to\gamma\in P\}$
Transitions:

(1) $([A\to\alpha\bullet X\,\beta],X,[A\to\alpha\,X\bullet\beta])$, $X\in(N\cup T), A\to\alpha\,X\,\beta\in P$;
(2) $([A\to\alpha\bullet B\,\beta],\epsilon,[B\to\bullet\,\gamma])$, $A\to\alpha\,B\,\beta,\ B\to\gamma\in P$;

The automaton $c(G)$ is called characteristic automaton for G.

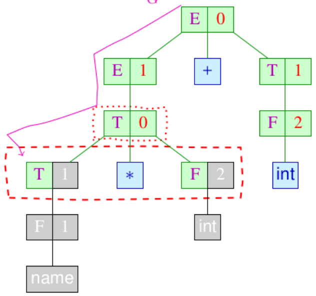
Reverse Rightmost Derivations in Shift-Reduce-Parsers

Idea: Observe *reverse rightmost*-derivations of M_G^R!

Input:

$+\ 40$

Pushdown:
$(\ q_0\ T * F\ )$



Characteristic Automaton

Observation:
The set of viable prefixes from $(N\cup T)^*$ for (admissible) items can be computed from the content of the shift-reduce parser's pushdown with the help of a finite automaton:

States: Items
Start state: $[S'\to\bullet\,S]$
Final states: $\{[B\to\gamma\bullet]\mid B\to\gamma\in P\}$
Transitions:

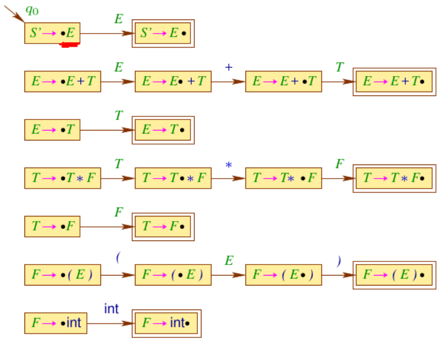
(1) $([A\to\alpha\bullet X\,\beta],X,[A\to\alpha\,X\bullet\beta])$, $X\in(N\cup T), A\to\alpha\,X\,\beta\in P$;
(2) $([A\to\alpha\bullet B\,\beta],\epsilon,[B\to\bullet\,\gamma])$, $A\to\alpha\,B\,\beta,\ B\to\gamma\in P$;

The automaton $c(G)$ is called characteristic automaton for G.

Characteristic Automaton

For example:

$$\begin{array}{lcll} E & \rightarrow & E+T & \mid \; T \\ T & \rightarrow & T*F & \mid \; F \\ F & \rightarrow & (\,E\,) & \mid \; \text{int} \end{array}$$



Characteristic Automaton

Observation:
The set of viable prefixes from $(N \cup T)^*$ for (admissible) items can be computed from the content of the shift-reduce parser's pushdown with the help of a finite automaton:

States: Items
Start state: $[S' \rightarrow \bullet\, S]$
Final states: $\{[B \rightarrow \gamma \bullet] \mid B \rightarrow \gamma \in P\}$
Transitions:

(1) $([A \rightarrow \alpha \bullet X\, \beta], X, [A \rightarrow \alpha\, X \bullet \beta]), \quad X \in (N \cup T), A \rightarrow \alpha\, X\, \beta \in P;$

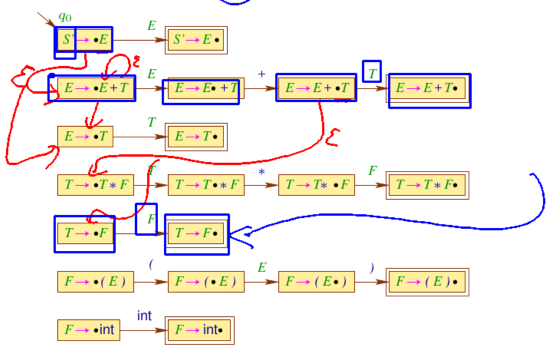
(2) $([A \rightarrow \alpha \bullet B\, \beta], \epsilon, [B \rightarrow \bullet\, \gamma]), \quad A \rightarrow \alpha\, B\, \beta\,,\; B \rightarrow \gamma \in P;$

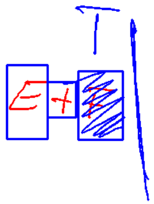
The automaton $c(G)$ is called characteristic automaton for G.

Characteristic Automaton

For example:

$$\begin{array}{lcll} E & \rightarrow & E+T & \mid \; T \\ T & \rightarrow & T*F & \mid \; F \\ F & \rightarrow & (\,E\,) & \mid \; \text{int} \end{array}$$

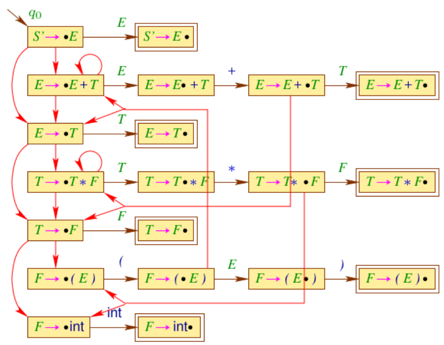




Characteristic Automaton

For example:

$$\begin{array}{lcll} E & \rightarrow & E+T & \mid \; T \\ T & \rightarrow & T*F & \mid \; F \\ F & \rightarrow & (\,E\,) & \mid \; \text{int} \end{array}$$

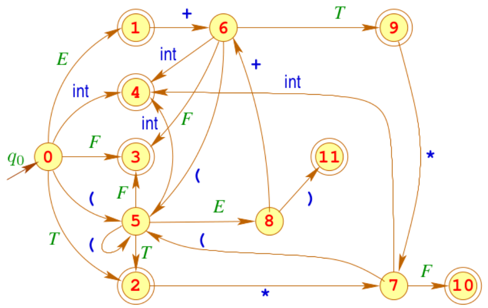


Canonical LR(0)-Automaton

The canonical $LR(0)$-automaton $LR(G)$ is created from $c(G)$ by:

1. performing arbitrarily many ϵ-transitions after every consuming transition
2. performing the powerset construction

... for example:



Canonical LR(0)-Automaton

Example:

$$\begin{array}{lcll} E & \to & E+T & \mid \ T \\ T & \to & T*F & \mid \ F \\ F & \to & (\,E\,) & \mid \ \text{int} \end{array}$$

Therefore we determine:

$q_0 = \{[S' \to \bullet E], [E \to \bullet E+T], [E \to \bullet T], [T \to \bullet T*F], [T \to \bullet F], [F \to \bullet(E)], [F \to \bullet \text{int}]\}$

$\delta(q_0, E) = q_1 = \{[S' \to E\bullet], [E \to E\bullet+T]\}$

$\delta(q_1, +) = q_2 = \{[E \to E+\bullet T], [T \to \bullet T*F], [T \to \bullet F]$

Canonical LR(0)-Automaton

$$\begin{array}{lclcl}
q_5 & = & \delta(q_0, () & = & \{[F \to (\bullet E)], \\
 & & & & [E \to \bullet E+T], \\
 & & & & [E \to \bullet T], \\
 & & & & [T \to \bullet T*F], \\
 & & & & [T \to \bullet F], \\
 & & & & [F \to \bullet (E)], \\
 & & & & [F \to \bullet \text{int}]\} \\
q_6 & = & \delta(q_1, +) & = & \{[E \to E+\bullet T], \\
 & & & & [T \to \bullet T*F], \\
 & & & & [T \to \bullet F], \\
 & & & & [F \to \bullet (E)], \\
 & & & & [F \to \bullet \text{int}]\}
\end{array}$$

$$\begin{array}{lclcl}
q_7 & = & \delta(q_2, *) & = & \{[T \to T*\bullet F], \\
 & & & & [F \to \bullet (E)], \\
 & & & & [F \to \bullet \text{int}]\} \\
q_8 & = & \delta(q_5, E) & = & \{[F \to (E\bullet)]\} \\
 & & & & [E \to E\bullet+T]\} \\
q_9 & = & \delta(q_6, T) & = & \{[E \to E+T\bullet], \\
 & & & & [T \to T\bullet*F]\} \\
q_{10} & = & \delta(q_7, F) & = & \{[T \to T*F\bullet]\} \\
q_{11} & = & \delta(q_8,)) & = & \{[F \to (E)\bullet]\}
\end{array}$$

Canonical LR(0)-Automaton

$$\begin{array}{lclcl}
q_5 & = & \delta(q_0, () & = & \{[F \to (\bullet E)], \\
 & & & & [E \to \bullet E+T], \\
 & & & & [E \to \bullet T], \\
 & & & & [T \to \bullet T*F], \\
 & & & & [T \to \bullet F], \\
 & & & & [F \to \bullet (E)], \\
 & & & & [F \to \bullet \text{int}]\} \\
q_6 & = & \delta(q_1, +) & = & \{[E \to E+\bullet T], \\
 & & & & [T \to \bullet T*F], \\
 & & & & [T \to \bullet F], \\
 & & & & [F \to \bullet (E)], \\
 & & & & [F \to \bullet \text{int}]\}
\end{array}$$

$$\begin{array}{lclcl}
q_7 & = & \delta(q_2, *) & = & \{[T \to T*\bullet F], \\
 & & & & [F \to \bullet (E)], \\
 & & & & [F \to \bullet \text{int}]\} \\
q_8 & = & \delta(q_5, E) & = & \{[F \to (E\bullet)]\} \\
 & & & & [E \to E\bullet+T]\} \\
q_9 & = & \delta(q_6, T) & = & \{[E \to E+T\bullet], \\
 & & & & [T \to T\bullet*F]\} \\
q_{10} & = & \delta(q_7, F) & = & \{[T \to T*F\bullet]\} \\
q_{11} & = & \delta(q_8,)) & = & \{[F \to (E)\bullet]\}
\end{array}$$

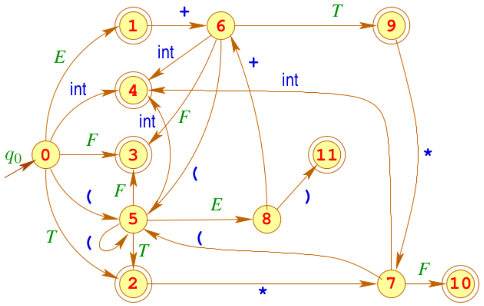
127 / 292

Canonical LR(0)-Automaton

The canonical $LR(0)$-automaton $LR(G)$ is created from $c(G)$ by:

1. performing arbitrarily many ϵ-transitions after every consuming transition
2. performing the powerset construction

... for example:



Canonical LR(0)-Automaton

Observation:

The canonical $LR(0)$-automaton can be created directly from the grammar.
Therefore we need a helper function δ^*_ϵ (ϵ-closure)

$$\delta^*_\epsilon(q) = q \cup \{[B \to \bullet\, \gamma] \mid \exists\, [A \to \alpha \bullet B'\, \beta'] \in q, \; \beta \in (N \cup T)^* : \; B' \to^* B\, \beta\}$$

We define:

- **States:** Sets of items;
- **Start state:** $\delta^*_\epsilon \{[S' \to \bullet\, S]\}$
- **Final states:** $\{q \mid \exists\, A \to \alpha \in P : \; [A \to \alpha \bullet] \in q\}$
- **Transitions:** $\delta(q, X) = \delta^*_\epsilon \{[A \to \alpha\, X \bullet \beta] \mid [A \to \alpha \bullet X\, \beta] \in q\}$

LR(0)-Parser

Idea for a parser:

- The parser manages a viable prefix $\alpha = X_1 \ldots X_m$ on the pushdown and uses $LR(G)$, to identify reduction spots.
- It can reduce with $A \to \gamma$, if $[A \to \gamma \bullet]$ is admissible for α

Optimization:

We push the states instead of the X_i in order not to process the pushdown's content with the automaton anew all the time.
Reduction with $A \to \gamma$ leads to popping the uppermost $|\gamma|$ states and continue with the state on top of the stack and input A.

Attention:

This parser is only deterministic, if each final state of the canonical $LR(0)$-automaton is conflict free.

LR(0)-Parser

... for example:

$$q_1 = \{[S' \to E \bullet], \; [E \to E \bullet + T]\}$$
$$q_2 = \{[E \to T \bullet], \; [T \to T \bullet * F]\}$$
$$q_3 = \{[T \to F \bullet]\}$$
$$q_4 = \{[F \to \text{int} \bullet]\}$$
$$q_9 = \{[E \to E + T \bullet], \; [T \to T \bullet * F]\}$$
$$q_{10} = \{[T \to T * F \bullet]\}$$
$$q_{11} = \{[F \to (\, E\,) \bullet]\}$$

The final states q_1, q_2, q_9 contain more then one admissible item
$\Rightarrow$ non deterministic!

LR(0)-Parser

The construction of the $LR(0)$-parser:

States: $Q \cup \{f\}$ (f fresh)
Start state: q_0
Final state: f

Transitions:

Shift: $(p, a, p\,q)$ if $q = \delta(p, a) \neq \emptyset$
Reduce: $(p\,q_1 \ldots q_m, \epsilon, p\,q)$ if $[A \to X_1 \ldots X_m \bullet] \in q_m,$ $q = \delta(p, A)$
Finish: $(q_0\,p, \epsilon, f)$ if $[S' \to S\bullet] \in p$

with $LR(G) = (Q, T, \delta, q_0, F)$.

LR(0)-Parser

Correctness:

we show:

The accepting computations of an $LR(0)$-parser are one-to-one related to those of a shift-reduce parser M_G^R.

we conclude:

- The accepted language is exactly $\mathcal{L}(G)$
- The sequence of reductions of an accepting computation for a word $w \in T$ yields a reverse rightmost derivation of G for w

LR(0)-Parser

Attention:
Unfortunately, the $LR(0)$-parser is in general non-deterministic.

We identify two reasons:

Reduce-Reduce-Conflict:
$[A \to \gamma\bullet]\,,\ [A' \to \gamma'\bullet] \in q$ with $A \neq A' \vee \gamma \neq \gamma'$

Shift-Reduce-Conflict:
$[A \to \gamma\bullet]\,,\ [A' \to \alpha \bullet a\,\beta] \in q$ with $a \in T$

for a state $q \in Q$.

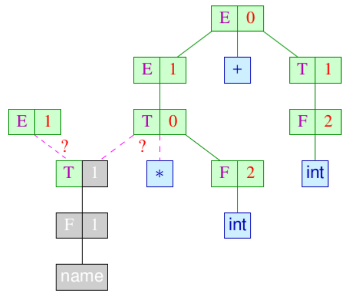
Those states are called $LR(0)$-unsuited.

Revisiting the Conflicts of the LR(0)-Automaton

What differenciates the particular Reductions and Shifts?

Input:
$*\ 2 + 40$

Pushdown:
$(\ q_0\ T\ )$



$$
\begin{array}{lcl}
E & \to & E + T \quad | \quad T \\
T & \to & T * F \quad | \quad F \\
F & \to & (\ E\ ) \quad | \quad \texttt{int}
\end{array}
$$

LR(k)-Grammars

Idea: Consider k-lookahead in conflict situations.

Definition:

The reduced contextfree grammar G is called $LR(k)$-grammar, if for $\text{First}_k(w) = \text{First}_k(x)$ with:

$$\left.\begin{array}{lllll} S & \to_R^* & \alpha\, A\, w & \to & \alpha\, \beta\, w \\ S & \to_R^* & \alpha'\, A'\, w' & \to & \alpha\, \beta\, x \end{array}\right\} \text{follows: } \alpha = \alpha' \wedge A = A' \wedge w' = x$$

LR(k)-Grammars

for example:

(1) $S \to A \mid B \qquad A \to a\, A\, b \mid 0 \qquad B \to a\, B\, b\, b \mid 1$

$a\, b^n\, b\, A\, b\, b^n\, c$

$b\, b\, b\, b$

LR(k)-Grammars

for example:

(3) $S \to a\, A\, c \qquad A \to b\, b\, A \mid b$ ... is not $LR(0)$, but $LR(1)$:

Let $S \to_R^* \alpha\, X\, w \to \alpha\, \beta\, w$ with $\{y\} = \text{First}_k(w)$ then $\alpha\, \underline{\beta}\, y$ is of one of these forms:

$$a\, b^{2n}\, \underline{b}\, c\ ,\ a\, b^{2n}\, \underline{b\, b\, A}\, c\ ,\ \underline{a\, A\, c}$$

(4) $S \to a\, A\, c \qquad A \to b\, A\, b \mid b$ ... is not $LR(k)$ for any $k \geq 0$:

Consider the rightmost derivations:

$$S \to_R^* a\, b^n\, A\, b^n\, c \to a\, b^n\, \underline{b}\, b^n\, c$$