Script generated by TTT

Title: Petter: Compilerbau (07.06.2018)

Date: Thu Jun 07 14:18:55 CEST 2018

Duration: 88:49 min

Pages: 24

LR(0)-Parser

... for example:

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

$$q_2 = \{[E \to T\,\bullet],\ [T \to T\,\bullet * F]\} \qquad q_9 = \{[E \to E + T\,\bullet],\ [T \to T\,\bullet * F]\}$$

$$q_3 = \{[T \to F\,\bullet]\} \qquad q_{10} = \{[T \to T * F\,\bullet]\}$$

$$q_4 = \{[F \to \text{int}\,\bullet]\} \qquad q_{11} = \{[F \to (\,E\,)\,\bullet]\}$$

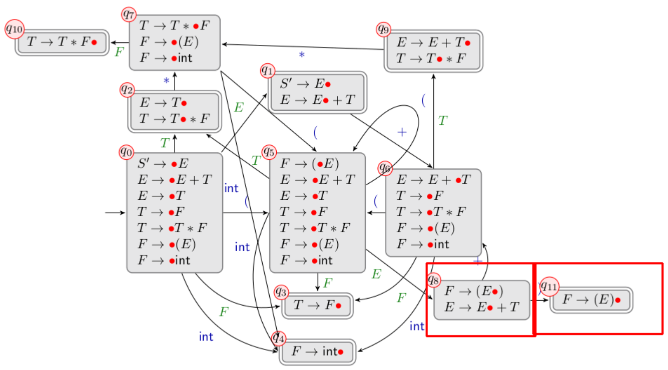
The final states q_1, q_2, q_9 contain more than one admissible item

$\Rightarrow$ non deterministic!

Canonical LR(0)-Automaton

For example:

$$\begin{array}{lcll} S' & \to & E & \\ E & \to & E + T & \mid\ T \\ T & \to & T * F & \mid\ F \\ F & \to & (\ E\ ) & \mid\ \text{int} \end{array}$$



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.

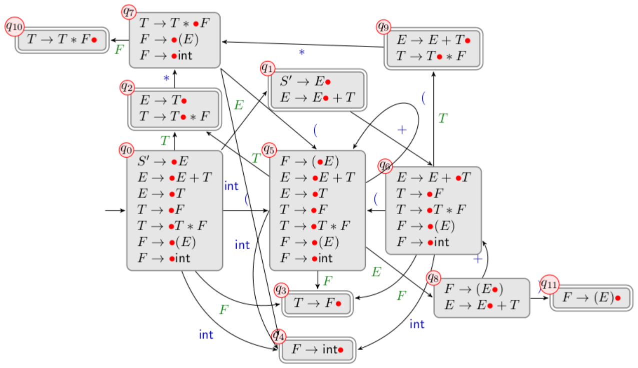
Attention:

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

Canonical LR(0)-Automaton

For example:

$$\begin{array}{llll} S' & \rightarrow & E & \\ E & \rightarrow & E+T & \mid\; T \\ T & \rightarrow & T*F & \mid\; F \\ F & \rightarrow & (\,E\,) & \mid\; \text{int} \end{array}$$



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 $A \rightarrow \gamma$, if $[A \rightarrow \gamma\,\bullet]$ 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 $A \rightarrow \gamma$ 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:

$$\begin{array}{lll} q_1 & = & \{[S' \rightarrow E\,\bullet], \\ & & \;[E \rightarrow E\,\bullet\,+T]\} \\ q_2 & = & \{[E \rightarrow T\,\bullet], \\ & & \;[T \rightarrow T\,\bullet * F]\} \\ q_3 & = & \{[T \rightarrow F\,\bullet]\} \\ q_4 & = & \{[F \rightarrow \text{int}\,\bullet]\} \end{array} \qquad \begin{array}{lll} q_9 & = & \{[E \rightarrow E+T\,\bullet], \\ & & \;[T \rightarrow T\,\bullet * F]\} \\ q_{10} & = & \{[T \rightarrow T * F\,\bullet]\} \\ q_{11} & = & \{[F \rightarrow (\,E\,)\,\bullet]\} \end{array}$$

The final states q_1, q_2, q_9 contain more than 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	$[A \rightarrow X_1 \ldots X_m\,\bullet] \in q_m$, $q = \delta(p, A)$
Finish:	$(q_0\,p, \epsilon, f)$	if	$[S' \rightarrow S\bullet] \in p$

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

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

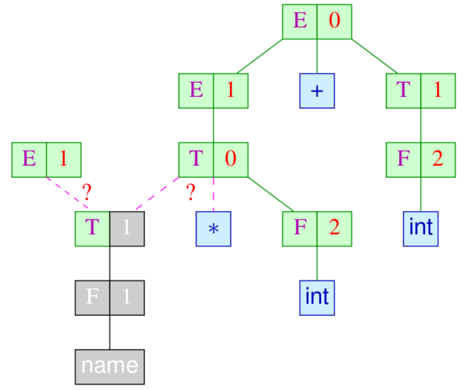
What differenciates the particular Reductions and Shifts?

Input:

$* \; 2 + 40$

Pushdown:

$(\; q_0 \; T \;)$



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

LR(k)-Grammars

Idea: Consider k-lookahead in conflict situations.

Definition:

The reduced contextfree grammar G is called $LR(k)$-grammar, if for $\text{First}_{|\alpha\beta|+k}(\alpha\,\beta\,w) = \text{First}_{|\alpha\beta|+k}(\alpha'\,\beta'\,w')$ with:

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

LR(k)-Grammars

for example:

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

LR(1)-Parsing

Idea: Let's equip items with 1-lookahead

Definition LR(1)-Item

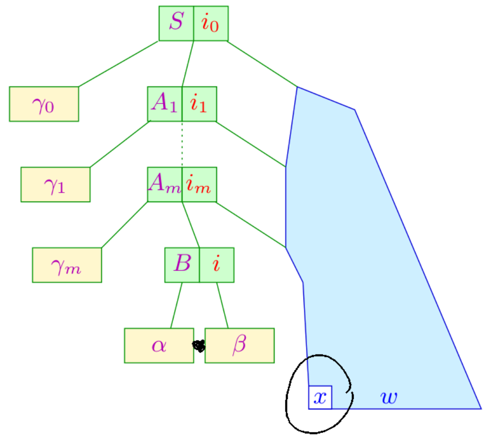
An $LR(1)$-item is a pair $[B \to \alpha \bullet \beta, x]$ with

$$x \in \text{Follow}_1(B) = \bigcup\{\text{First}_1(\nu) \mid S \to^* \mu\, B\, \nu\}$$

Admissible LR(1)-Items

The item $[B \to \alpha \bullet \beta, x]$ is *admissable* for $\gamma\,\alpha$ if:

$S \to_R^* \gamma\, B\, w$ with $\{x\} = \text{First}_1(w)$



... with $\gamma_0 \dots \gamma_m = \gamma$

The Characteristic LR(1)-Automaton

The set of admissible $LR(1)$-items for viable prefixes is again computed with the help of the finite automaton $c(G, 1)$.

The automaton $c(G, 1)$:

States: $LR(1)$-items

Start state: $[S' \to \bullet\, S,\ \epsilon]$

Final states: $\{[B \to \gamma\bullet,\ x] \mid B \to \gamma \in P, x \in \text{Follow}_1(B)\}$

Transitions:

(1) $([A \to \alpha \bullet X\,\beta,\ x], X, [A \to \alpha\, X \bullet \beta,\ x]), \quad X \in (N \cup T)$

(2) $([A \to \alpha \bullet B\,\beta,\ x], \epsilon,\ [B \to \bullet\,\gamma,\ x']),$
$A \to \alpha\, B\, \beta,\ B \to \gamma \in P,$
$x' \in \text{First}_1(\beta) \odot_1 \{x\}$

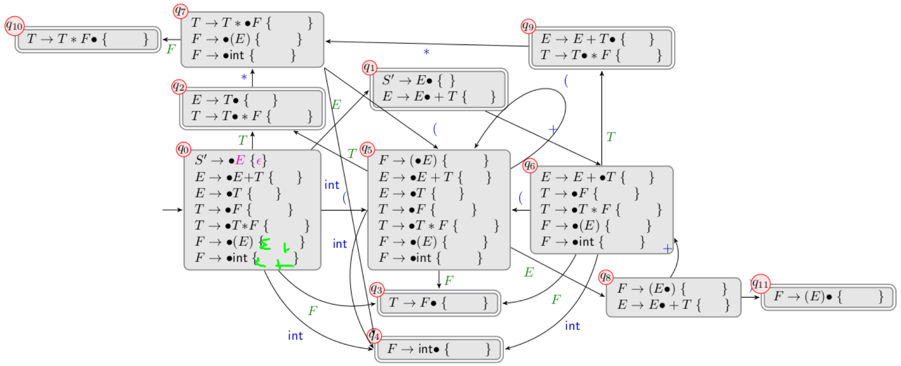
The Canonical LR(1)-Automaton

The canonical $LR(1)$-automaton $LR(G, 1)$ is created from $c(G, 1)$, by performing arbitrarily many ϵ-transitions and then making the resulting automaton deterministic ...

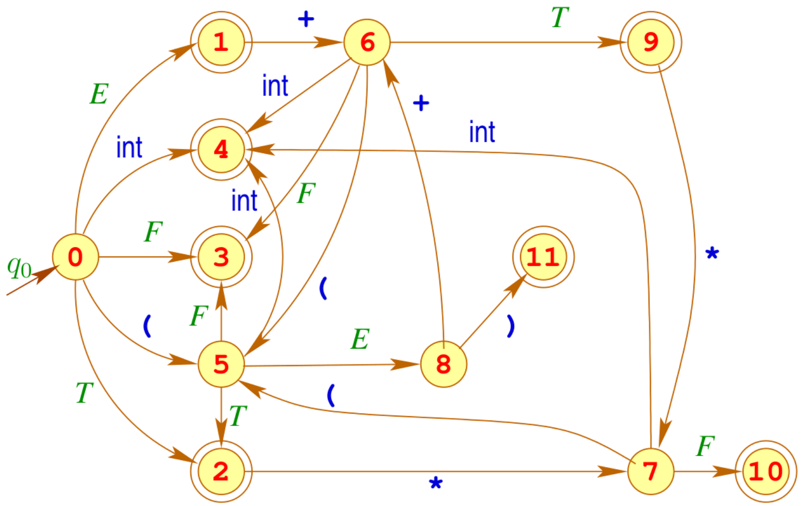
Canonical LR(1)-Automaton

For example:

$$
\begin{array}{lcll}
S' & \to & E & \\
E & \to & E + T & \mid \; T \\
T & \to & T * F & \mid \; F \\
F & \to & (\,E\,) & \mid \; \texttt{int}
\end{array}
$$



The Canonical LR(1)-Automaton



The Canonical LR(1)-Automaton

Discussion:

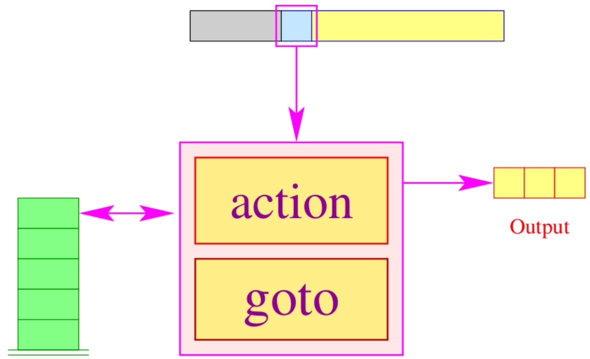
- In the example, the number of states was almost doubled
  ... and it can become even worse
- The conflicts in states q_1, q_2, q_9 are now resolved !
  e.g. we have:

$$
q_9: \quad
\begin{array}{l}
E \to E+T\bullet\ \{\epsilon, +\} \\
T \to T\bullet *F\ \{\epsilon, +, *\}
\end{array}
$$

with:

$$\{\epsilon, +\} \cap (\mathsf{First}_1(*F) \odot_1 \{\epsilon, +, *\}) = \{\epsilon, +\} \cap \{*\} = \emptyset$$

The LR(1)-Parser:



- The goto-table encodes the transitions:

$$\mathsf{goto}[q, X] = \delta(q, X) \quad \in \quad Q$$

- The action-table describes for every state q and possible lookahead w the necessary action.

The LR(1)-Parser:

The construction of the $LR(1)$-parser:

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

Transitions:

Shift: $(p, a, p\,q)$ if $q = \text{goto}[q, a]$, $s = \text{action}[p, w]$

Reduce: $(p\,q_1 \ldots q_{|\beta|}, \epsilon, p\,q)$ if $[A \to \beta\,\bullet] \in q_{|\beta|}$, $q = \text{goto}(p, \boxed{A})$, $[A \to \beta\,\bullet] = \text{action}[q_{|\beta|}, w]$

Finish: $(q_0\,p, \epsilon, f)$ if $[S' \to S\bullet] \in p$

with $LR(G,1) = (Q, T, \delta, q_0, F)$.

The LR(1)-Parser:

Possible actions are:

shift	//	Shift-operation
reduce $(A \to \gamma)$	//	Reduction with callback/output
error	//	Error

... for example:

$$S' \to E$$
$$E \to E + T^{\,0} \mid T^{\,1}$$
$$T \to T * F^{\,0} \mid F^{\,1}$$
$$F \to (\ E\ )^{\,0} \mid \texttt{int}^{\,1}$$

action	\$	int	(	)	+	*
q_1	$S', 0$				s	
q_2	$E, 1$				$E, 1$	s
q_2'				$E, 1$	$E, 1$	s
q_3	$T, 1$				$T, 1$	$T, 1$
q_3'				$T, 1$	$T, 1$	$T, 1$
q_4	$F, 1$				$F, 1$	$F, 1$
q_4'				$F, 1$	$F, 1$	$F, 1$
q_9	$E, 0$				$E, 0$	s
q_9'				$E, 0$	$E, 0$	s
q_{10}	$T, 0$				$T, 0$	$T, 0$
q_{10}'				$T, 0$	$T, 0$	$T, 0$
q_{11}	$F, 0$				$F, 0$	$F, 0$
q_{11}'				$F, 0$	$F, 0$	$F, 0$

The Canonical LR(1)-Automaton

In general:

We identify two conflicts:

Reduce-Reduce-Conflict:

$[A \to \gamma\,\bullet, x]$, $[A' \to \gamma'\,\bullet, x]$ $\in q$ with $A \neq A' \vee \gamma \neq \gamma'$

Shift-Reduce-Conflict:

$[A \to \gamma\,\bullet, x]$, $[A' \to \alpha\,\bullet\,a\,\beta, y]$ $\in q$
with $a \in T$ und $x \in \{a\}$.

for a state $q \in Q$.

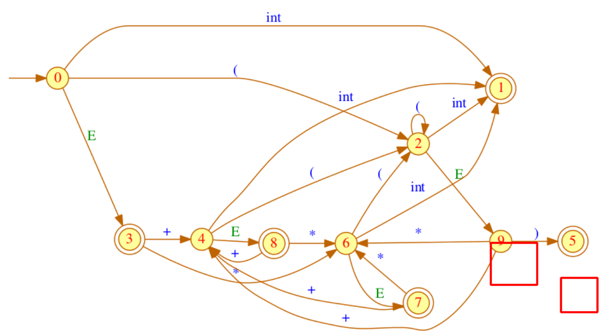
Such states are now called $LR(1)$-unsuited

Precedences

Many parser generators give the chance to fix Shift-/Reduce-Conflicts by patching the action table either by hand or with *token precedences*.

... for example:

$$S' \to E^{\,0}$$
$$E \to E + E^{\,0}$$
$$\mid\ E * E^{\,1}$$
$$\mid\ (\ E\ )^{\,2}$$
$$\mid\ \texttt{int}^{\,3}$$



What if precedences are not enough?

Example (very simplified lambda expressions):

$$
\begin{array}{lcl}
E & \rightarrow & (\ E\ )^{0} \mid \mathsf{ident}^{1} \mid L^{2} \\
L & \rightarrow & \langle \mathsf{args} \rangle \Rightarrow E^{0} \\
\langle \mathsf{args} \rangle & \rightarrow & (\ \langle \mathsf{idlist} \rangle\ )^{0} \mid \mathsf{ident}^{1} \\
\langle \mathsf{idlist} \rangle & \rightarrow & \langle \mathsf{idlist} \rangle\ \mathsf{ident}^{0} \mid \mathsf{ident}^{1}
\end{array}
$$