Script **generated by TTT**

Title: Petter: Compilerbau (06.06.2019)

Date: Thu Jun 06 14:30:08 CEST 2019

Duration: 74:25 min

Pages: 29

Shift-Reduce Parser

Construction:

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

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

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

Reverse Rightmost Derivations in Shift-Reduce-Parsers

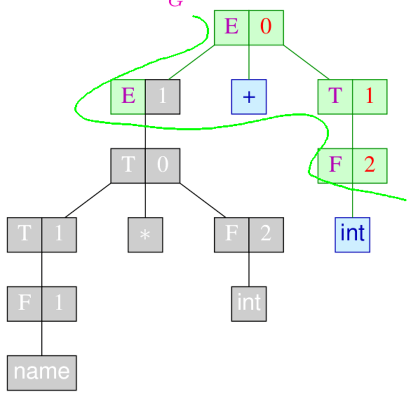
Idea: Observe *reverse rightmost*-derivations of M_G^R!

Input:

$+ \; 40$

Pushdown:

$(\; q_0 \; E \;)$



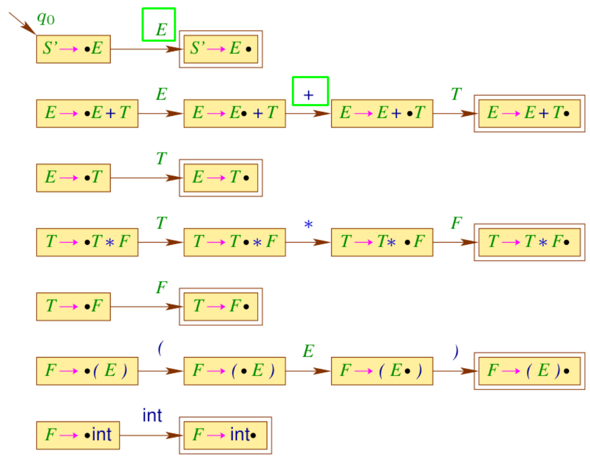
$$
\begin{array}{lll}
E & \to & E{+}T^{\,0} \mid T^{\,1} \\
T & \to & T{*}F^{\,0} \mid F^{\,1} \\
F & \to & (\,E\,)^{\,0} \mid \text{name}^{\,1} \mid \text{int}^{\,2}
\end{array}
$$

Characteristic Automaton

For example:

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

Transitions (1)

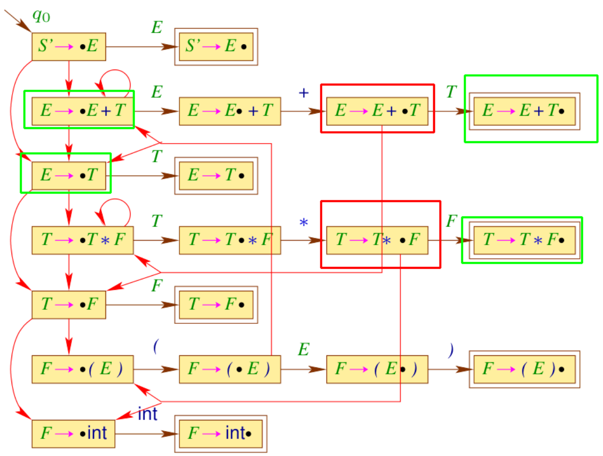


Characteristic Automaton

For example:

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

Transitions (2)

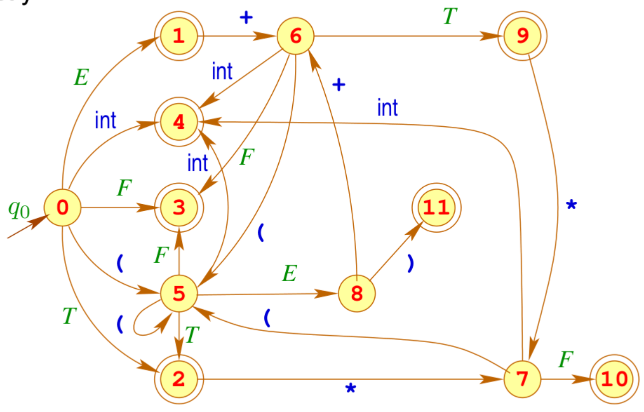


Canonical LR(0)-Automaton

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
3. Idea: or rather apply characteristic automaton construction to powersets directly?

... for example:



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:

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

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

LR(k)-Grammars

for example:

(3) $S \to a\,A\,c \qquad A \to b\,b\,A \mid b$

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

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

LR(1)-Parsing

Idea: Let's equip items with 1-lookahead

Definition LR(1)-Item

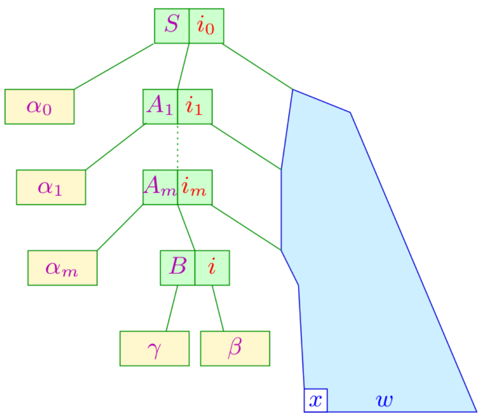
An $LR(1)$-item is a pair $[B \to \alpha \bullet \beta,\ x]$ with

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

Admissible LR(1)-Items

The $LR(1)$-Item $[B \to \gamma \bullet \beta,\ x]$ is *admissable* for $\alpha\,\gamma$ if:

$S \to^*_R \alpha\,B\,w$ with $\{x\} = \text{First}_1(w)$



... with $\alpha_0 \dots \alpha_m = \alpha$

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 ...

But again, it can be constructed directly from the grammar; analoguously to $LR(0)$, we need the ϵ-closure δ^*_ϵ as a helper function:

$$\delta^*_\epsilon(q) = q \cup \{[C \to \bullet\,\gamma,\ x] \mid \begin{array}{l} C \to \gamma \in P\,, \\ [A \to \alpha \bullet B\,\beta',\ x'] \in q\,, \\ B \to^* C\,\beta\,, \\ x \in \text{First}_1(\beta\,\beta') \odot_1 \{x'\}\} \end{array}$$

Then, we define:

States: Sets of $LR(1)$-items;

Start state: $\delta^*_\epsilon\ \{[S' \to \bullet\,S,\ \$]\}$

Final states: $\{q \mid [A \to \alpha\,\bullet,\ x] \in q\}$

Transitions: $\delta(q, X) = \delta^*_\epsilon\ \{[A \to \alpha\,X \bullet \beta,\ x] \mid [A \to \alpha \bullet X\,\beta,\ x] \in q\}$

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

This automaton works like $c(G)$ — but additionally manages a 1-prefix from Follow_1 of the left-hand sides.

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 ...

But again, it can be constructed directly from the grammar; analoguously to $LR(0)$, we need the ϵ-closure δ_ϵ^* as a helper function:

$$\delta_\epsilon^*(q) = q \cup \{[C \to \bullet\,\gamma,\ x] \mid \begin{array}{l} C \to \gamma \in P\,, \\ [A \to \alpha \bullet B\,\beta',\ x'] \in q\,, \\ B \to^* C\,\beta\,, \\ x \in \text{First}_1(\beta\,\beta') \odot_1 \{x'\}\} \end{array}$$

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

This automaton works like $c(G)$ — but additionally manages a 1-prefix from Follow_1 of the left-hand sides.

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 ...

But again, it can be constructed directly from the grammar; analoguously to $LR(0)$, we need the ϵ-closure δ_ϵ^* as a helper function:

$$\delta_\epsilon^*(q) = q \cup \{[C \to \bullet\,\gamma,\ x] \mid \begin{array}{l} C \to \gamma \in P\,, \\ [A \to \alpha \bullet B\,\beta',\ x'] \in q\,, \\ B \to^* C\,\beta\,, \\ x \in \text{First}_1(\beta\,\beta') \odot_1 \{x'\}\} \end{array}$$

Then, we define:

States: Sets of $LR(1)$-items;

Start state: $\delta_\epsilon^*\ \{[S' \to \bullet\, S,\ \$]\}$

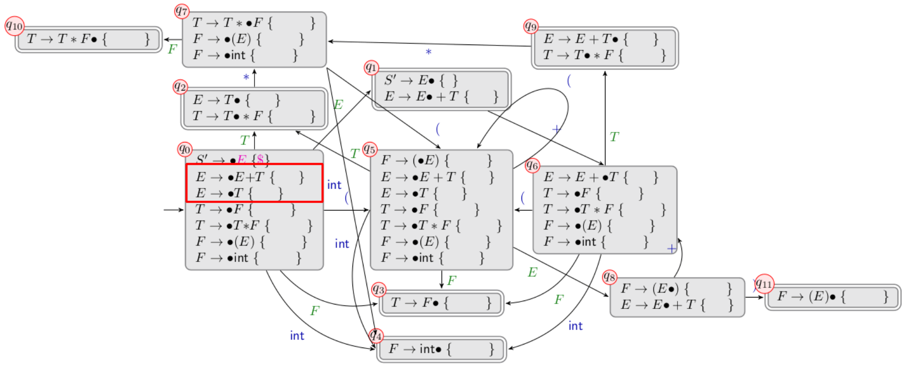
Final states: $\{q \mid [A \to \alpha\,\bullet,\ x] \in q\}$

Transitions: $\delta(q, X) = \delta_\epsilon^*\ \{[A \to \alpha\, X \bullet\, \beta,\ x] \mid [A \to \alpha \bullet X\,\beta,\ x] \in q\}$

Canonical LR(1)-Automaton

For example:

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



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 ...

But again, it can be constructed directly from the grammar; analoguously to $LR(0)$, we need the ϵ-closure δ^*_ϵ as a helper function:

$$
\begin{array}{rl}
\delta^*_\epsilon(q) = q \cup \{[C \rightarrow \bullet\,\gamma,\, x] \mid & C \rightarrow \gamma \in P, \\
& [A \rightarrow \alpha \bullet B\,\beta',\, x'] \in q, \\
& B \rightarrow^* C\,\beta, \\
& x \in \text{First}_1(\beta\,\beta') \odot_1 \{x'\}\}
\end{array}
$$

Then, we define:

States: Sets of $LR(1)$-items;

Start state: $\delta^*_\epsilon\ \{[S' \rightarrow \bullet\, S,\ \$]\}$

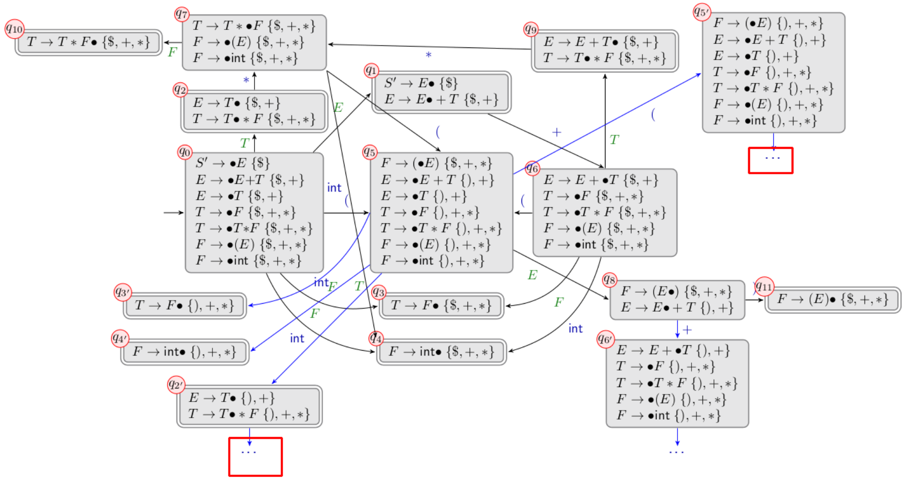
Final states: $\{q \mid [A \rightarrow \alpha\,\bullet,\ x] \in q\}$

Transitions: $\delta(q, X) = \delta^*_\epsilon\ \{[A \rightarrow \alpha\, X \bullet\ \beta,\ x] \mid [A \rightarrow \alpha \bullet X\,\beta,\ x] \in q\}$

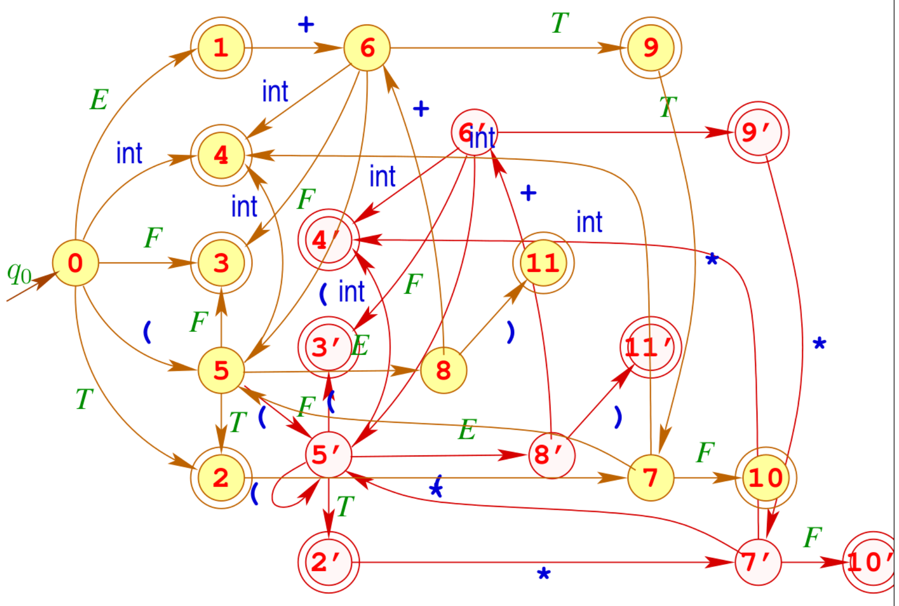
Canonical LR(1)-Automaton

For example:

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



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:

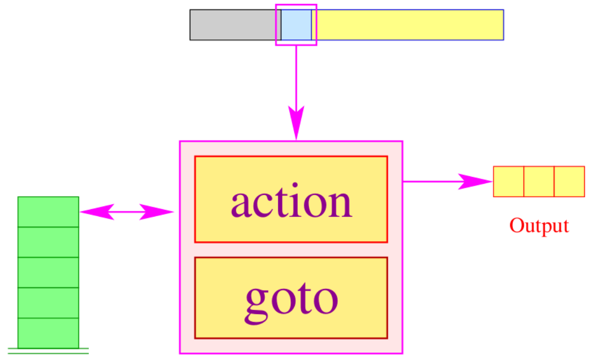
q_9:

$$E \to E{+}T\bullet \quad \{\$,+\}$$
$$T \to T\bullet{*}F \quad \{\$,+,*\}$$

with:

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

The LR(1)-Parser:



- The goto-table encodes the transitions:

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

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

The 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:

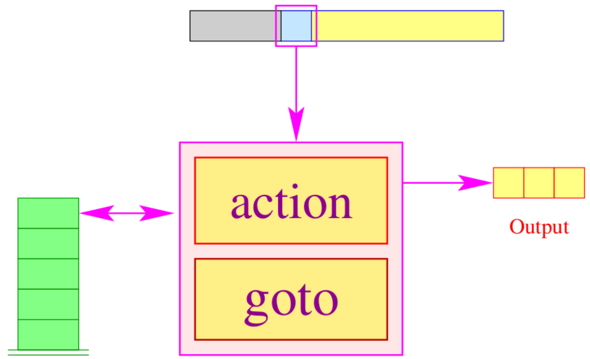
$$E \to E{+}T\bullet \quad \{\$,+\}$$
$$T \to T\bullet{*}F \quad \{\$,+,*\}$$

with:

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

141 / 286

The LR(1)-Parser:



- The goto-table encodes the transitions:

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

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

142 / 286

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 $q = \text{goto}[q, a]$, $\text{s} = \text{action}[p, w]$

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

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:

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

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 LR(1)-Parser:

Possible actions are:

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

... for example:

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

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:

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

Shift-Reduce-Conflict:

$$[A \to \gamma\,\bullet,\, x],\ [A' \to \alpha\,\bullet\,a\,\beta,\, y] \in q$$

with $a \in T$ und $x \in \{a\} \odot_k \text{First}_k(\beta) \odot_k \{y\}$.

for a state $q \in Q$.

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

Theorem:

A reduced contextfree grammar G is called $LR(k)$ iff the canonical $LR(k)$-automaton $LR(G, k)$ has no $LR(k)$-unsuited states.

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:

$$
\begin{array}{lll}
S' & \to & E^{\,0} \\
E & \to & E + E^{\,0} \\
 & | & E * E^{\,1} \\
 & | & (\, E\,)^{\,2} \\
 & | & \texttt{int}^{\,3}
\end{array}
$$

Shift-/Reduce Conflict in states 8, 7:

$$
\begin{array}{l}
[E \to E \bullet * E^{\,1} \quad] \\
[E \to E + E \bullet^{\,0} \quad , *] \\
< \gamma\, E * E\, , + \omega > \\
[E \to E \bullet + E^{\,0} \quad] \\
[E \to E * E \bullet^{\,1} \quad , +] \\
< \gamma\, E + E\, , * \omega >
\end{array}
$$

$*$ *higher precedence*

$+$ *lower precedence*

action	\$	int	(	)	+	*
q_0	$S', 0$				s	s
q_1	$E, 3$			$E, 3$	$E, 3$	$E, 3$
q_2	s				s	s
q_3	s				s	s
q_4	s			s	s	s
q_5	$E, 2$			$E, 2$	$E, 2$	$E, 2$
q_6	s			s	s	s
q_7	$E, 1$			$E, 1$	$E, 1$	s
q_8	$E, 0$			$E, 0$	$E, 0$	s
q_9	s			s	s	s

What if precedences are not enough?

Example (very simplified lambda expressions):

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