Script **generated by TTT**

Title: Petter: Compilerbau (30.05.2016)

Date: Mon May 30 14:25:45 CEST 2016

Duration: 88:45 min

Pages: 44

Chapter 4:
Bottom-up Analysis

Bottom-up Analysis

Attention:
Many grammars are not $LL(k)$!

A reason for that is:

Definition
Grammar G is called left-recursive, if

$$\boxed{A \rightarrow^{+} A\,\beta} \qquad \text{for an} \quad A \in N\,,\ \beta \in (T \cup N)^{*}$$

Bottom-up Analysis

Attention:
Many grammars are not $LL(k)$!

A reason for that is:

Definition
Grammar G is called left-recursive, if

$$A \rightarrow^{+} A\,\beta \qquad \text{for an} \quad A \in N\,,\ \beta \in (T \cup N)^{*}$$

Example:

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

... is left-recursive

Bottom-up Analysis

Theorem:

Let a grammar G be reduced and left-recursive, then $\boxed{G \text{ is not } LL(k)}$ for any k.

Proof:

Let $\boxed{A \to A\,\beta \mid \alpha \quad \in P}$
and A be reachable from S

Assumption: G is $LL(k)$

Bottom-up Analysis

Theorem:

Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

Let $A \to A\,\beta \mid \alpha \quad \in P$
and A be reachable from S

Assumption: G is $LL(k)$

$\Rightarrow$ First$_k(\alpha\,\boxed{\beta^n}\,\gamma)\,\cap$
First$_k(\alpha\,\boxed{\beta^{n+1}}\,\gamma) = \emptyset$

Bottom-up Analysis

Theorem:

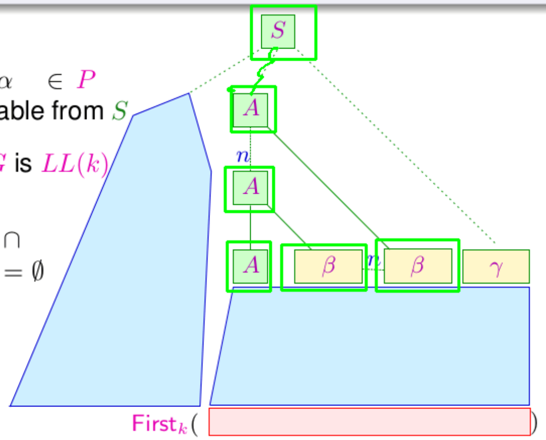
Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

Let $A \to A\,\beta \mid \alpha \quad \in P$
and A be reachable from S

Assumption: G is $LL(k)$

$\Rightarrow$ First$_k(\alpha\,\beta^n\,\gamma)\,\cap$
First$_k(\alpha\,\beta^{n+1}\,\gamma) = \emptyset$



Bottom-up Analysis

Theorem:

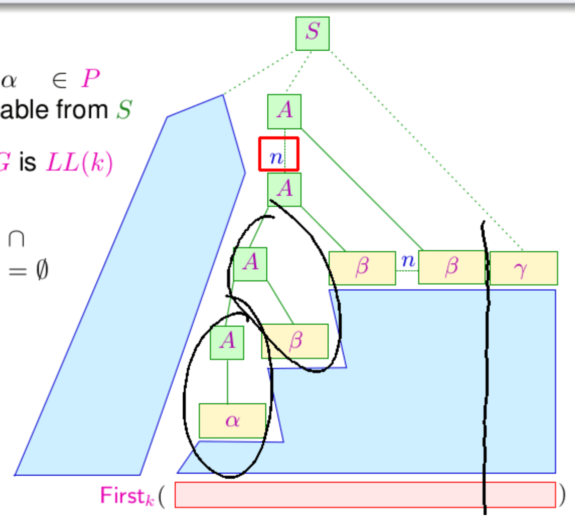
Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

Let $A \to A\,\beta \mid \alpha \quad \in P$
and A be reachable from S

Assumption: G is $LL(k)$

$\Rightarrow$ First$_k(\alpha\,\beta^n\,\gamma)\,\cap$
First$_k(\alpha\,\beta^{n+1}\,\gamma) = \emptyset$



Bottom-up Analysis

Theorem:

Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

Let $\quad A \to A\,\beta \,|\, \alpha \quad \in P$
and A be reachable from S

Assumption: $\quad G$ is $LL(k)$

$\Rightarrow \text{First}_k(\alpha\,\beta^n\,\gamma) \,\cap$
$\text{First}_k(\alpha\,\beta^{n+1}\,\gamma) = \emptyset$

Case 1: $\quad \beta \to^* \epsilon \quad$ — Contradiction !!!
Case 2: $\quad \beta \to^* w \neq \epsilon \implies \text{First}_k(\alpha\,w^k\,\gamma) \cap \text{First}_k(\alpha\,w^{k+1}\,\gamma) \neq \emptyset$

Shift-Reduce Parser

Donald Knuth

Idea:

We *delay* the decision whether to reduce until we know, whether the input matches the right-hand-side of a rule!

Construction: Shift-Reduce parser M_G^R

- The input is shifted successively to the pushdown.
- Is there a complete right-hand side (a handle) atop the pushdown, it is replaced (reduced) by the corresponding left-hand side

Shift-Reduce Parser

Example:

$$\begin{array}{lll} S & \to & A\,B \\ A & \to & a \\ B & \to & b \end{array}$$

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:

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

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

Shift-Reduce Parser

Construction:

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

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

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

Example-computation:

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

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

Reverse Rightmost Derivations in Shift-Reduce-Parsers

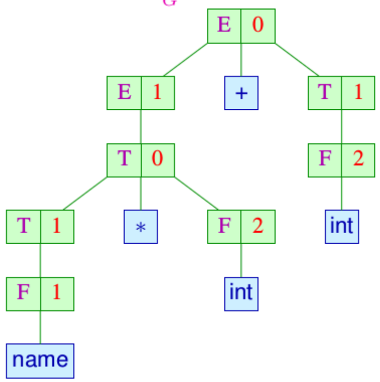
Idea: Observe *reverse rightmost*-derivations of M_G^R!

Input:

counter $* \; 2 + 40$

Pushdown:

(q_0)



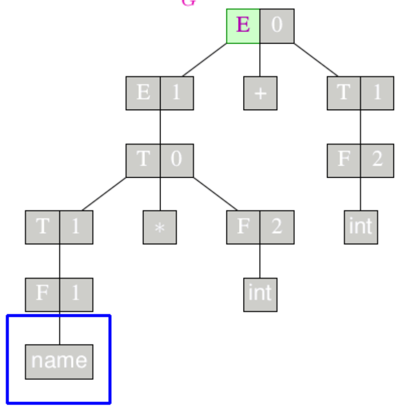
Reverse Rightmost Derivations in Shift-Reduce-Parsers

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

Input:

Pushdown:

(f)



Reverse Rightmost Derivations in Shift-Reduce-Parsers

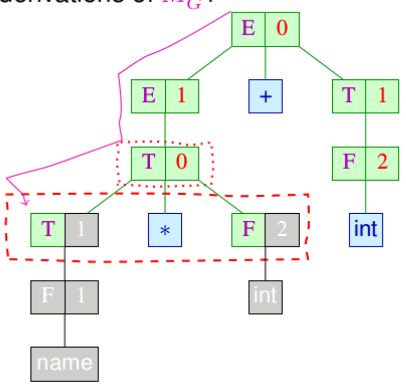
Idea: Observe *reverse rightmost*-derivations of M_G^R!

Input:

$+\ 40$

Pushdown:

$(\ q_0\ T * F\ )$



Generic Observation:
In a sequence of configurations of M_G^R

$$(q_0\,\alpha\,\gamma,\ v) \vdash (q_0\,\alpha\,B,\ v) \vdash^* (q_0\,S,\ \epsilon)$$

we call $\alpha\,\gamma$ a viable prefix for the complete item $[B \to \gamma\bullet]$.

Reverse Rightmost Derivations in Shift-Reduce-Parsers

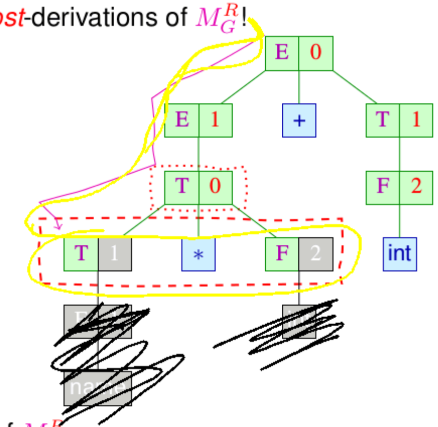
Idea: Observe *reverse rightmost*-derivations of M_G^R!

Input:

$+\ 40$

Pushdown:

$(\ q_0\ T * F\ )$



Generic Observation:
In a sequence of configurations of M_G^R

$$(q_0\,\alpha\,\gamma,\ v) \vdash (q_0\,\alpha\,B,\ v) \vdash^* (q_0\,S,\ \epsilon)$$

we call $\alpha\,\gamma$ a viable prefix for the complete item $[B \to \gamma\bullet]$.

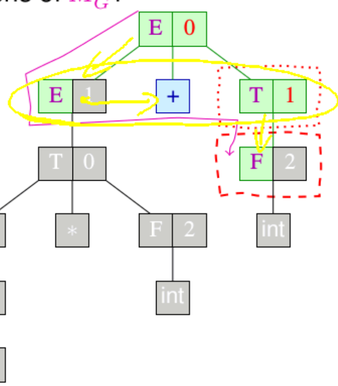
Reverse Rightmost Derivations in Shift-Reduce-Parsers

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

Input:

Pushdown:

$(\ q_0\ E + F\ )$



Generic Observation:
In a sequence of configurations of M_G^R

$$(q_0\,\alpha\,\gamma,\ v) \vdash (q_0\,\alpha\,B,\ v) \vdash^* (q_0\,S,\ \epsilon)$$

we call $\alpha\,\gamma$ a viable prefix for the complete item $[B \to \gamma\bullet]$.

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:

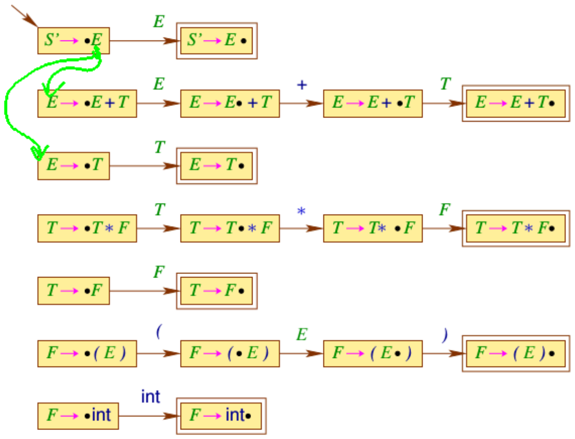
(1) $([A \to \alpha \bullet X\,\beta], X, [A \to \alpha\, X \bullet \beta]),\quad X \in (N \cup T), A \to \alpha\, X\, \beta \in P;$
(2) $([A \to \alpha \bullet B\,\beta], \epsilon, [B \to \bullet\,\gamma]),\quad A \to \alpha\, B\, \beta\ ,\ B \to \gamma \in P;$

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

Characteristic Automaton

For example:

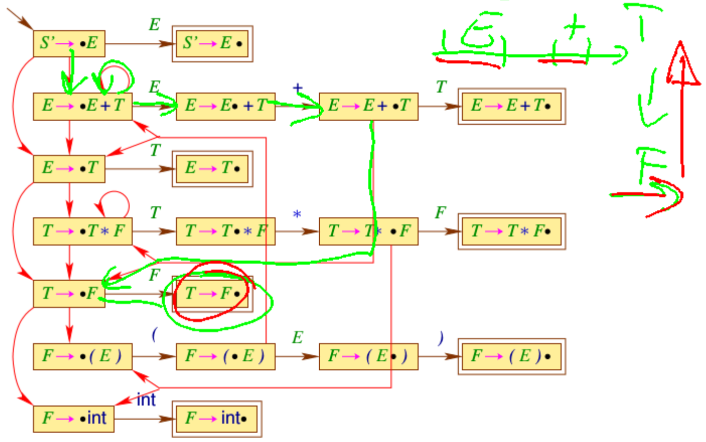
$$\begin{array}{lcl} E & \to & E+T \quad | \quad T \\ T & \to & T*F \quad | \quad F \\ F & \to & (\,E\,) \quad | \quad \mathsf{int} \end{array}$$



Characteristic Automaton

For example:

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

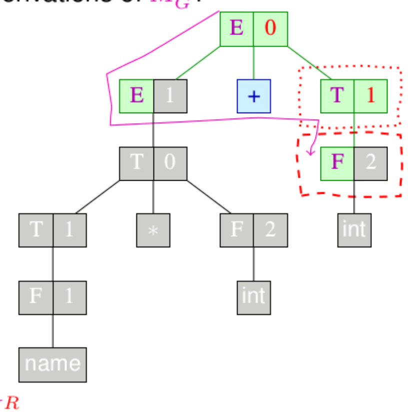


Reverse Rightmost Derivations in Shift-Reduce-Parsers

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

Input:

Pushdown:
$(\, q_0 \; E + F \,)$



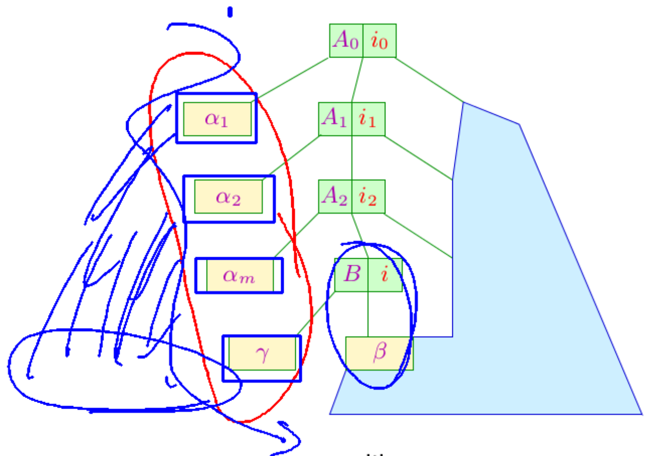
Generic Observation:
In a sequence of configurations of M_G^R

$$(q_0\, \alpha\, \gamma,\; v) \vdash (q_0\, \alpha\, B,\; v) \vdash^* (q_0\, S,\; \epsilon)$$

we call $\alpha\,\gamma$ a viable prefix for the complete item $[B \to \gamma \bullet]$.

Bottom-up Analysis: Admissible Items

The item $[B \to \gamma \bullet \beta]$ is called admissible for α' iff
$S \to_R^* \alpha\, B\, v$ 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) $([A \to \alpha \bullet X\,\beta], X, [A \to \alpha\, X \bullet \beta])$, $\quad X \in (N \cup T), A \to \alpha\, X\,\beta \in P$;

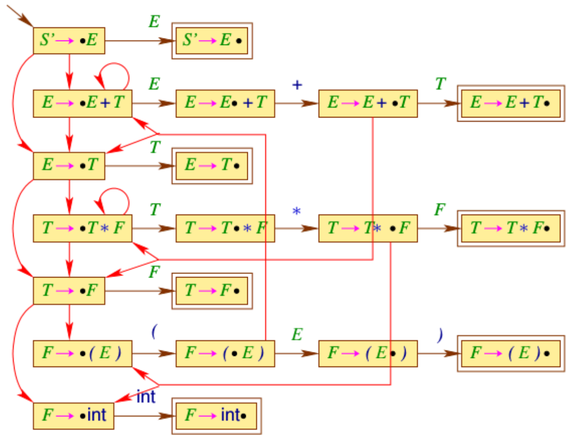
(2) $([A \to \alpha \bullet B\,\beta], \epsilon, [B \to \bullet\, \gamma])$ $\quad A \to \alpha\, B\, \beta\,, \quad B \to \gamma \in P$;

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

Characteristic Automaton

For example:

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

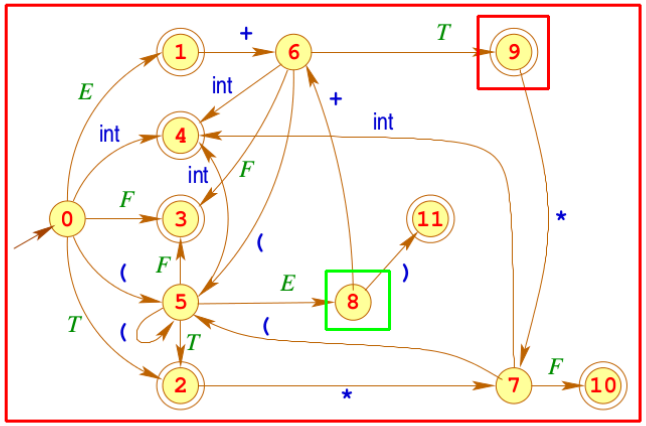


Canonical LR(0)-Automaton

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:

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

q_0

Therefore we determine:

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

E q_1

$[S' \to E\bullet]$
$[E \to E\bullet + T]$

(

$[F \to (\bullet E)]$
$[E \to \bullet E+T]$
$[E \to \bullet T]$
$T \to \ldots$

Canonical LR(0)-Automaton

Example:

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

Therefore we determine:

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

$$\begin{array}{lcl} q_1 & = \delta(q_0,E) = & \{[S'\to E\,\bullet], \\ & & [E\to E\,\bullet+T]\} \\ q_2 & = \delta(q_0,T) = & \{[E\to T\,\bullet], \\ & & [T\to T\,\bullet*F]\} \\ q_3 & = \delta(q_0,F) = & \{[T\to F\,\bullet]\} \\ q_4 & = \delta(q_0,\text{int}) = & \{[F\to \text{int}\,\bullet]\} \end{array}$$

Canonical LR(0)-Automaton

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

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

Canonical LR(0)-Automaton

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

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

129 / 290

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:** $\{q \mid \exists\, A\to\alpha \in P: \boxed{[A\to\alpha\,\bullet]} \in q\}$
- **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

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.

131 / 290

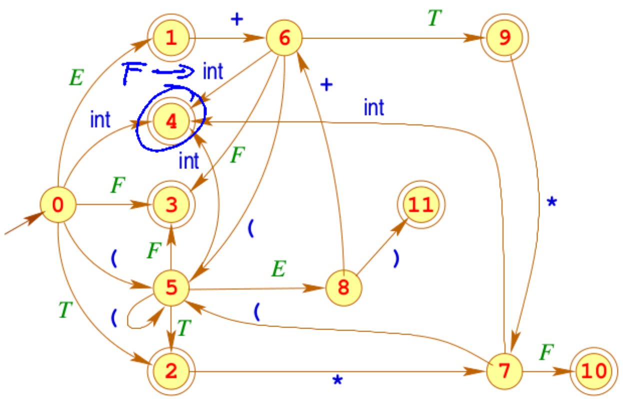
Canonical LR(0)-Automaton

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

~~int~~ ~~*~~ int

... for example:



Reverse Rightmost Derivations in Shift-Reduce-Parsers

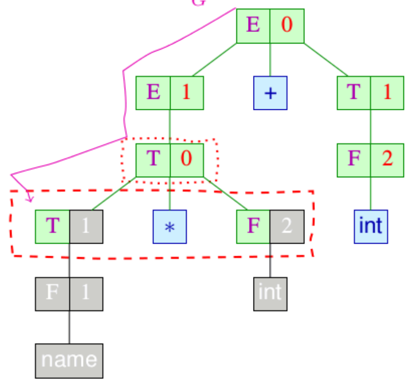
Idea: Observe *reverse rightmost*-derivations of M_G^R!

Input:

$+ \; 40$

Pushdown:

$(\; q_0 \; T * F \;)$



Generic Observation:

In a sequence of configurations of M_G^R

$$(q_0 \, \alpha \, \gamma, \; v) \vdash (q_0 \, \alpha \, B, \; v) \vdash^* (q_0 \, S, \; \epsilon)$$

we call $\alpha \, \gamma$ a viable prefix for the complete item $[B \to \gamma \bullet]$.

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 $\alpha = X_1 \dots X_m$ 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:

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

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

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

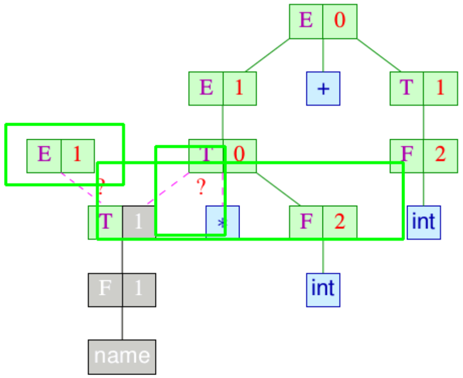
What differenciates the particular Reductions and Shifts?

Input:

$* 2 + 40$

Pushdown:

$(q_0 \; T)$



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

Revisiting the Conflicts of the LR(0)-Automaton

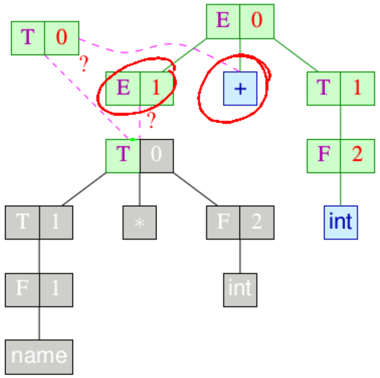
What differenciates the particular Reductions and Shifts?

Input:

$+ 40$

Pushdown:

$(q_0 \; T)$



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

Revisiting the Conflicts of the LR(0)-Automaton

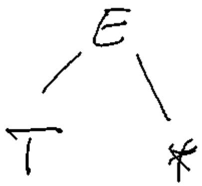
Idea: Matching lookahead with *right context* matters!

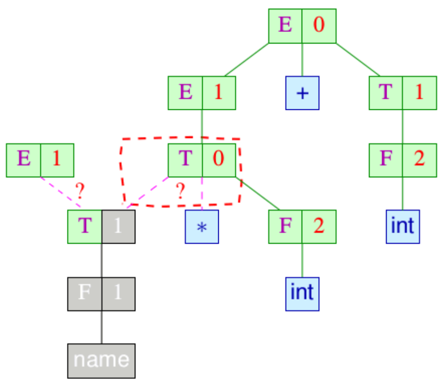
Input:

$* 2 + 40$

Pushdown:

$(q_0 \; T)$





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

Revisiting the Conflicts of the LR(0)-Automaton

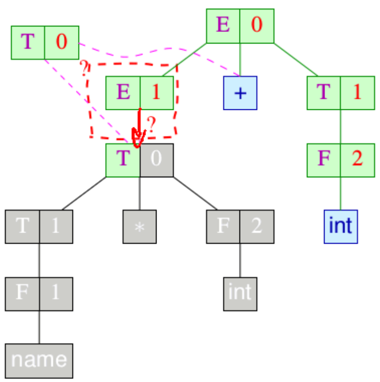
Idea: Matching lookahead with *right context* matters!

Input:

$+ 40$

Pushdown:

$(q_0 \; T)$



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

LR(k)-Grammars

Idea: Consider k-lookahead in conflict situations.

Definition:

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

$$\left.\begin{array}{llll} S & \rightarrow_R^* & \alpha\, A\, w & \rightarrow & \alpha\,\beta\, w \\ S & \rightarrow_R^* & \alpha'\, A'\, w' & \rightarrow & \alpha\,\beta\, x \end{array}\right\} \text{follows: } \alpha = \alpha' \;\wedge\; A = A' \;\wedge\; w' = x$$