Bottom-up Analysis

Theorem:

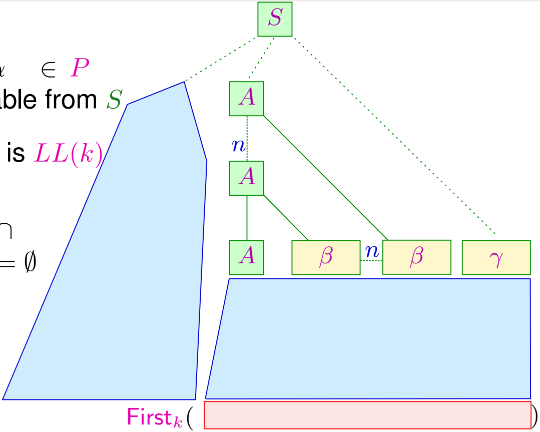
Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

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

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

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



Bottom-up Analysis

Theorem:

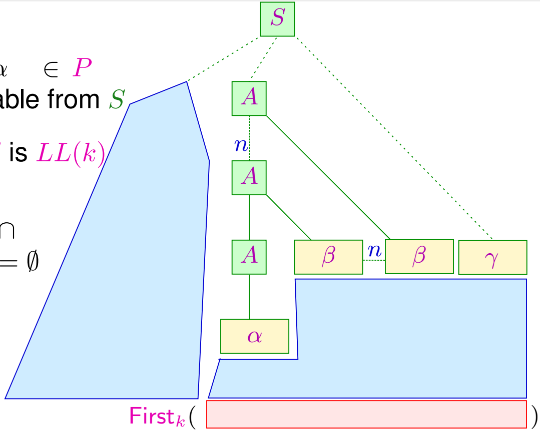
Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

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

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

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



Bottom-up Analysis

Theorem:

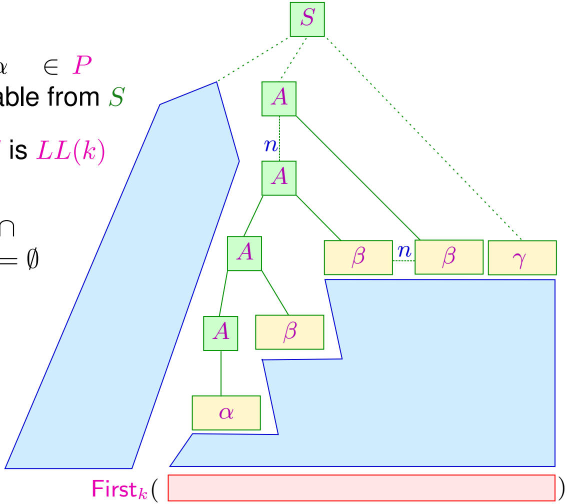
Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

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

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

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



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: $\quad \beta \to^* w \neq \epsilon \Longrightarrow \text{First}_k(\alpha\,w^k\,\gamma) \cap \text{First}_k(\alpha\,w^{k+1}\,\gamma) \neq \emptyset$

Shift-Reduce Parser

Idea:

Donald Knuth

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:

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

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:

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

Example-computation:

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

Shift-Reduce Parser

Observation:

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

$$(\epsilon, w) \vdash^* (A, \epsilon) \qquad \text{iff} \qquad A \rightarrow^* w$$

- 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

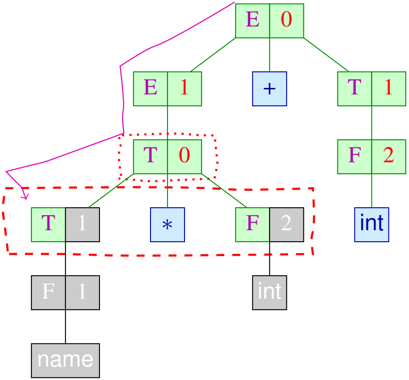
Idea: Observe *reverse rightmost*-derivations of M_G^R!

Input:

$+\ 40$

Pushdown:

$(\ q_0\ T * F\ )$



Generic Observation:

In a sequence of configurations of M_G^R

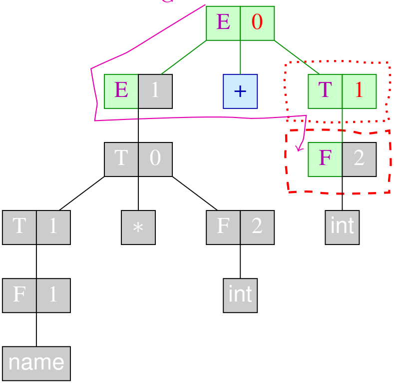
$$(q_0\,\alpha\,\gamma,\ v) \vdash (q_0\,\alpha\,B,\ v) \vdash^* (q_0\,S,\ \epsilon)$$

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:

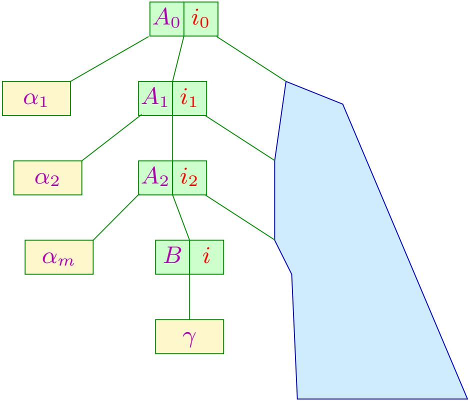
In a sequence of configurations of M_G^R

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

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

Bottom-up Analysis: Viable Prefix

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

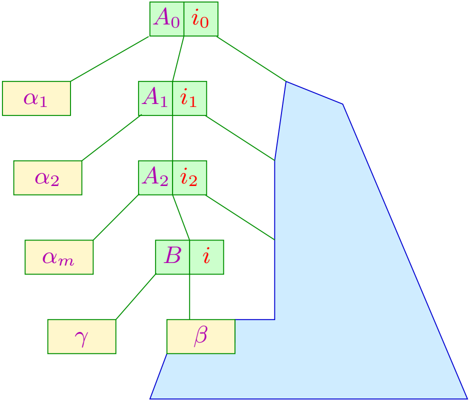


... with $\alpha = \alpha_1 \,\ldots\, \alpha_m$

Conversely, for an arbitrary valid word α' we can determine the set of all later on possibly matching rules ...

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:

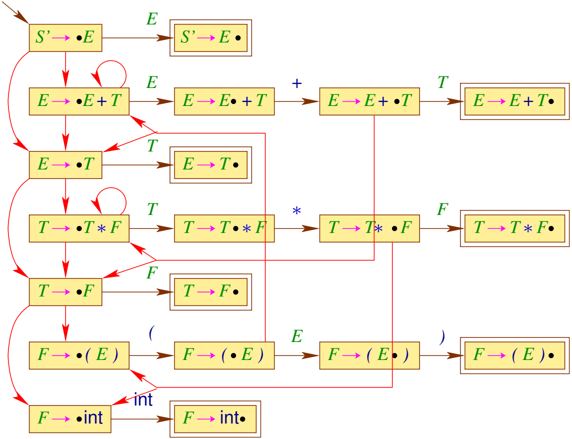
$$
\begin{array}{lll}
(1) & ([A \to \alpha \bullet X\, \beta], X, [A \to \alpha\, X \bullet \beta]), & X \in (N \cup T), A \to \alpha\, X\, \beta \in P; \\
(2) & ([A \to \alpha \bullet B\, \beta], \epsilon, [B \to \bullet\, \gamma]), & A \to \alpha\, B\, \beta\,,\ B \to \gamma \in P;
\end{array}
$$

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

Characteristic Automaton

For example:

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

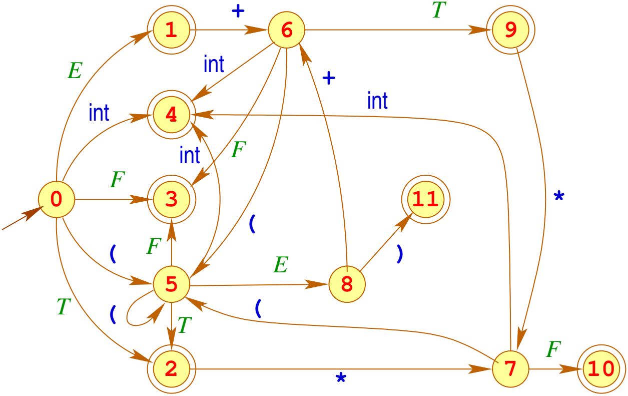


Canonical LR(0)-Automaton

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:

$$\begin{array}{lcl@{\quad}c@{\quad}l} E & \to & E + T & \mid & T \\ T & \to & T * F & \mid & F \\ F & \to & (\ E\ ) & \mid & \mathsf{int} \end{array}$$

Therefore we determine:

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

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

Canonical LR(0)-Automaton

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

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

$$\begin{array}{lllll} q_7 & = & \delta(q_2, *) & = & \{[T \to T * \bullet F], \\ & & & & \ [F \to \bullet (\,E\,)], \\ & & & & \ [F \to \bullet \mathsf{int}]\} \end{array}$$

$$\begin{array}{lllll} q_8 & = & \delta(q_5, E) & = & \{[F \to (\,E \bullet)]\} \\ & & & & \ [E \to E \bullet + T]\} \end{array}$$

$$\begin{array}{lllll} q_9 & = & \delta(q_6, T) & = & \{[E \to E + T \bullet], \\ & & & & \ [T \to T \bullet * F]\} \end{array}$$

$$\begin{array}{lllll} q_{10} & = & \delta(q_7, F) & = & \{[T \to T * F \bullet]\} \end{array}$$

$$\begin{array}{lllll} q_{11} & = & \delta(q_8, \,)\,) & = & \{[F \to (\,E\,) \bullet]\} \end{array}$$

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:

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

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

$\Rightarrow$ non deterministic!

LR(0)-Parser

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:

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

Shift-Reduce-Conflict:

$$[A \to \gamma \bullet], \; [A' \to \alpha \bullet a \, \beta] \; \in \; q \quad \text{with} \quad a \in T$$

for a state $q \in Q$.

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

Revisiting the Conflicts of the LR(0)-Automaton

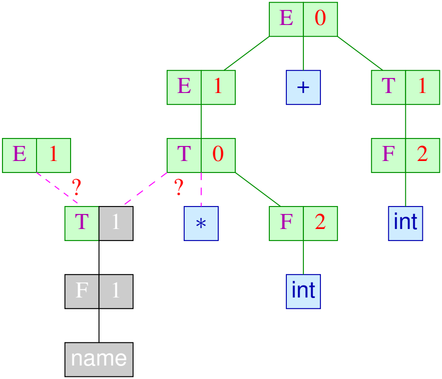
What differenciates the particular Reductions and Shifts?

Input:

$$* \ 2 + 40$$

Pushdown:

$(\ q_0 \ T \ )$



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

Revisiting the Conflicts of the LR(0)-Automaton

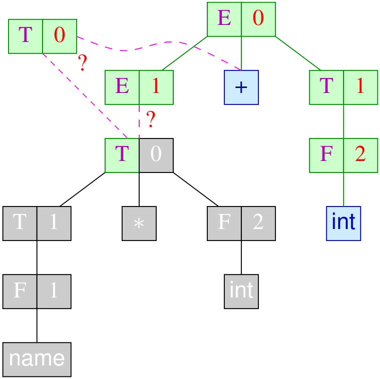
What differenciates the particular Reductions and Shifts?

Input:

$+ \; 40$

Pushdown:

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



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

Revisiting the Conflicts of the LR(0)-Automaton

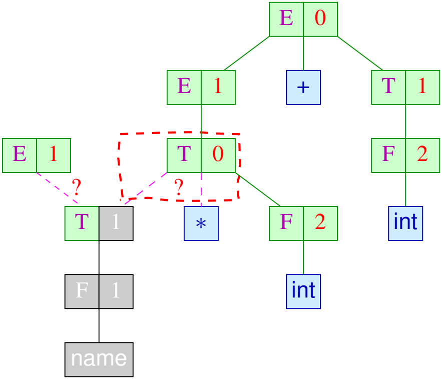
Idea: Matching lookahead with *right context* matters!

Input:

$* \, 2 + 40$

Pushdown:

$(\, q_0 \, T \,)$



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

Revisiting the Conflicts of the LR(0)-Automaton

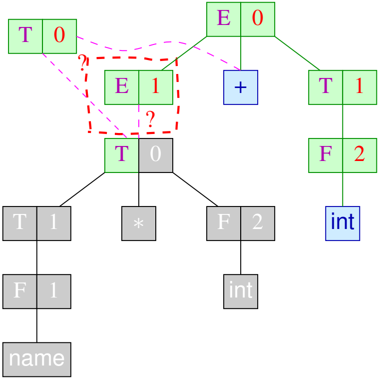
Idea: Matching lookahead with *right context* matters!

Input:

$+ \; 40$

Pushdown:

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



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

LR(k)-Grammars

Idea: Consider k-lookahead in conflict situations.

Definition:

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

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

Strategy for testing Grammars for $LR(k)$-property

1. Focus iteratively on all rightmost derivations $S \to_R^* \alpha\, X\, w \to \alpha\, \beta\, w$
2. Identify handle $\alpha\, \underline{\beta}$ in s. forms $\alpha\, \beta\, w$ ($w \in T^*$, $\alpha, \beta \in (N \cup T)^*$)
3. Determine minimal k, such that $\text{First}_k(w)$ associates β with a unique $X \to \beta$ for non-prefixing $\alpha\, \underline{\beta}$s

LR(k)-Grammars

for example:

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

... is not $LL(k)$ for any k — but $LR(0)$:

Let $S \to_R^* \alpha\,X\,w \to \alpha\,\beta\,w$. Then $\alpha\,\underline{\beta}$ is of one of these forms:

$$\underline{A}\,,\ \underline{B}\,,\ a^n\,\underline{a\,A\,b}\,,\ a^n\,\underline{a\,B\,b\,b}\,,\ a^n\,\underline{0}\,,\ a^n\,\underline{1} \qquad (n \geq 0)$$

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

... is also not $LL(k)$ for any k — but again $LR(0)$:

Let $S \to_R^* \alpha\,X\,w \to \alpha\,\beta\,w$. Then $\alpha\,\underline{\beta}$ is of one of these forms:

$$a\,\underline{b}\,,\ a\,\underline{A\,b\,b}\,,\ \underline{a\,A\,c}$$

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:

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

LR(1)-Parsing

Idea: Let's equip items with 1-lookahead

Definition LR(1)-Item

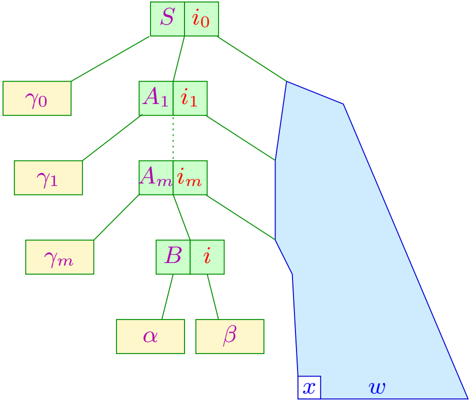
An $LR(1)$-item is a pair $[B \to \alpha \bullet \beta,\ x]$ with

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

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 $\{x\} = \mathsf{First}_1(w)$



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

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

Transitions:

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

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

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:

$$\delta_\epsilon^*(q) = q \cup \{[C \to \bullet\gamma,\, x] \mid \exists\, [A \to \alpha \bullet B\,\beta',\, x'] \in q, \beta \in (N \cup T)^* : \\ B \to^* C\,\beta \;\wedge\; x \in \mathsf{First}_1(\beta\,\beta') \odot_1 \{x'\}\}$$

Then, we define:

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

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

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

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

The Canonical LR(1)-Automaton

For example:

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

$$\mathsf{First}_1(S') = \mathsf{First}_1(E) = \mathsf{First}_1(T) = \mathsf{First}_1(F) = \mathsf{name}, \mathsf{int}, ($$

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

$$\begin{array}{lllll}
q_3 & = & \delta(q_0, F) & = & \{[T \to F\, \bullet\, ,\ \{\epsilon, +, *\}]\} \\
q_4 & = & \delta(q_0, \mathsf{int}) & & \{[F \to \mathsf{int}\, \bullet\, ,\ \{\epsilon, +, *\}]\} \\
q_5 & = & \delta(q_0, (\,) & = & \{[F \to (\ \bullet\, E\ ),\ \{\epsilon, +, *\}], \\
& & & & [E \to \bullet\, E+T,\ \{\ ),+\}], \\
& & & & [E \to \bullet\, T,\ \{\ ),+\}], \\
& & & & [T \to \bullet\, T*F,\ \{\ ),+,*\}], \\
& & & & [T \to \bullet\, F,\ \{\ ),+,*\}], \\
& & & & [F \to \bullet\, (\ E\ ),\ \{\ ),+,*\}], \\
& & & & [F \to \bullet\, \mathsf{int},\ \{\ ),+,*\}]\}
\end{array}$$

The Canonical LR(1)-Automaton

For example:

$$\begin{array}{lllll} E & \to & E+T & \mid & T \\ T & \to & T*F & \mid & F \\ F & \to & (\ E\ ) & \mid & \mathsf{int} \end{array}$$

$$\mathsf{First}_1(S') = \mathsf{First}_1(E) = \mathsf{First}_1(T) = \mathsf{First}_1(F) = \mathsf{name}, \mathsf{int}, ($$

$$\begin{array}{lllll} q_5' & = & \delta(q_5, () & = & \{[F \to (\ \bullet E\ ),\ \{\ )\ , +, *\}], \\ & & & & [E \to \bullet\ E+T,\ \{\ )\ , +\}], \\ & & & & [E \to \bullet\ T,\ \{\ )\ , +\}], \\ & & & & [T \to \bullet\ T*F,\ \{\ )\ , +, *\}], \\ & & & & [T \to \bullet\ F,\ \{\ )\ , +, *\}], \\ & & & & [F \to \bullet\ (\ E\ )\ ,\ \{\ )\ , +, *\}], \\ & & & & [F \to \bullet\ \mathsf{int},\ \{\ )\ , +, *\}]\} \\ q_6 & = & \delta(q_1, +) & = & \{[E \to E + \bullet\ T,\ \{\epsilon, +\}], \\ & & & & [T \to \bullet\ T*F,\ \{\epsilon, +, *\}], \\ & & & & [T \to \bullet\ F,\ \{\epsilon, +, *\}], \\ & & & & [F \to \bullet\ (\ E\ )\ ,\ \{\epsilon, +, *\}], \\ & & & & [F \to \bullet\ \mathsf{int},\ \{\epsilon, +, *\}]\} \end{array}$$

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

The Canonical LR(1)-Automaton

For example:

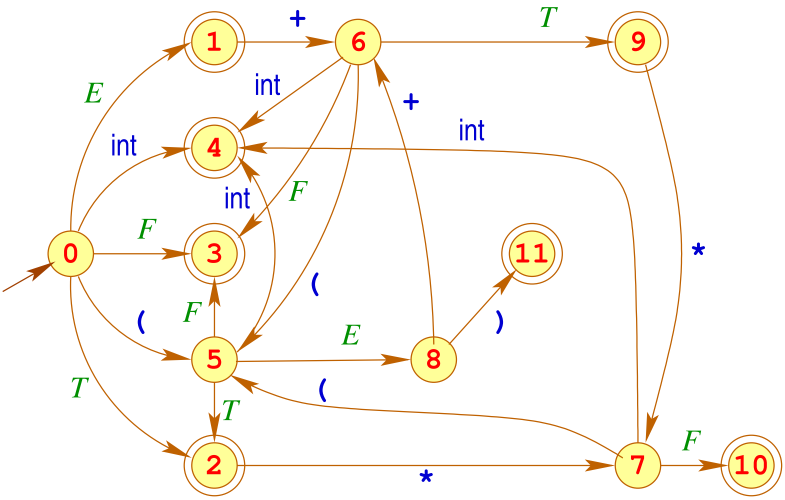
$$\begin{array}{lcll} E & \to & E + T & \mid \; T \\ T & \to & T * F & \mid \; F \\ F & \to & (\ E\ ) & \mid \; \text{int} \end{array}$$

$$\text{First}_1(S') = \text{First}_1(E) = \text{First}_1(T) = \text{First}_1(F) = \text{name}, \text{int}, ($$

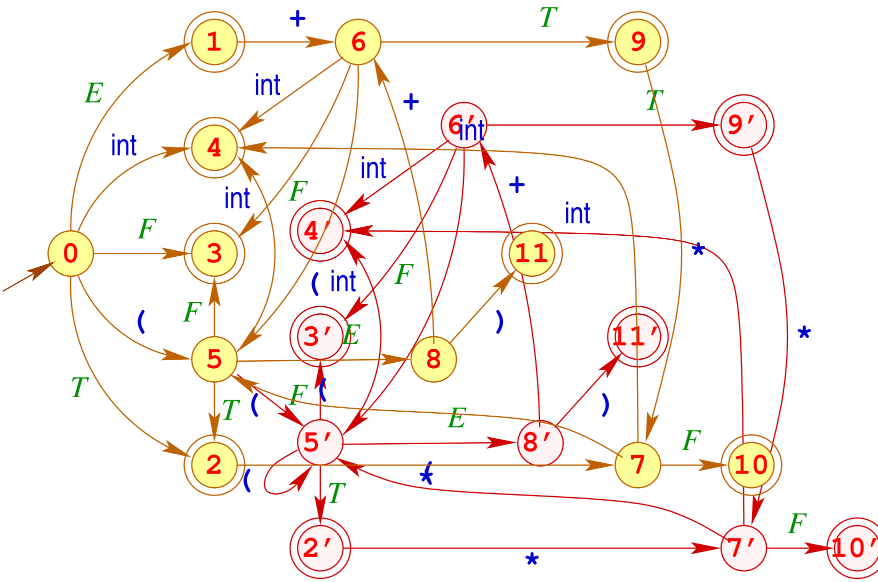
$$\begin{array}{lcccl}
q'_2 & = & \delta(q'_5, T) & = & \{[E \to T\bullet,\ \{\ )\ , +\}], \\
 & & & & [T \to T\ \bullet * F,\ \{\ )\ , +, *\}]\} \\
q'_3 & = & \delta(q'_5, F) & = & \{[T \to F\bullet,\ \{\ )\ , +, *\}]\} \\
q'_4 & = & \delta(q'_5, \text{int}) & = & \{[F \to \text{int}\bullet,\ \{\ )\ , +, *\}]\} \\
q'_6 & = & \delta(q_8, +) & = & \{[E \to E + \bullet\ T,\ \{\ )\ , +\}], \\
 & & & & [T \to\ \bullet\ T * F,\ \{\ )\ , +, *\}], \\
 & & & & [T \to\ \bullet\ F,\ \{\ )\ , +, *\}], \\
 & & & & [F \to\ \bullet\ (\ E\ )\ ,\ \{\ )\ , +, *\}], \\
 & & & & [F \to\ \bullet\ \text{int},\ \{\ )\ , +, *\}]\}
\end{array}$$

$$\begin{array}{lcccl}
q'_7 & = & \delta(q_9, *) & = & \{[T \to T * \bullet\ F,\ \{\ )\ , +, *\}], \\
 & & & & [F \to\ \bullet\ (\ E\ )\ ,\ \{\ ), +, *\}], \\
 & & & & [F \to\ \bullet\ \text{int},\ \{\ )\ , +, *\}]\} \\
q'_8 & = & \delta(q'_5, E) & = & \{[F \to (\ E\ \bullet\ )\ , \{\ )\ , +, *\}]\} \\
 & & & & [E \to E\ \bullet + T,\ \{\ )\ , +\}]\} \\
q'_9 & = & \delta(q'_6, T) & = & \{[E \to E + T\bullet,\ \{\ )\ , +\}], \\
 & & & & [T \to\ T\ \bullet * F,\ \{\ )\ , +, *\}]\} \\
q'_{10} & = & \delta(q'_7, F) & = & \{[T \to T * F\bullet,\ \{\ )\ , +, *\}]\} \\
q'_{11} & = & \delta(q'_8, \ )\ ) & = & \{[F \to (\ E\ )\ \bullet,\ \{\ )\ , +, *\}]\}
\end{array}$$

The Canonical LR(1)-Automaton



The Canonical LR(1)-Automaton



The Canonical LR(1)-Automaton

Discussion:

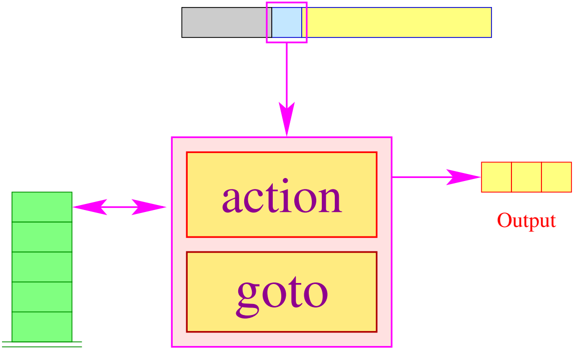
- In the example, the number of states was almost doubled
  ... and it can become even worse
- The conflicts in states q_1, q_2, q_9 are now resolved !
  e.g. we have for:

$$\begin{array}{rl} q_9 = & \{[E \to E + T \bullet,\ \{\epsilon, +\}], \\ & \ [T \to T \bullet * F,\ \{\epsilon, +, *\}]\} \end{array}$$

with:

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

The LR(1)-Parser:



- The goto-table encodes the transitions:

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

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

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

$$
\begin{array}{lcl}
E & \to & E + T^{\,0} \quad | \quad T^{\,1} \\
T & \to & T * F^{\,0} \quad | \quad F^{\,1} \\
F & \to & (\ E\ )^{\,0} \quad | \quad \text{int}^{\,1}
\end{array}
$$

action	ϵ	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:

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

Shift-Reduce-Conflict:

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

with $a \in T$ und $x \in \{a\}$.

for a state $q \in Q$.

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

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 $x \in \{a\} \odot_k \mathsf{First}_k(\beta) \odot_k \{y\}$.

for a state $q \in Q$.

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

Special LR(k)-Subclasses

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.

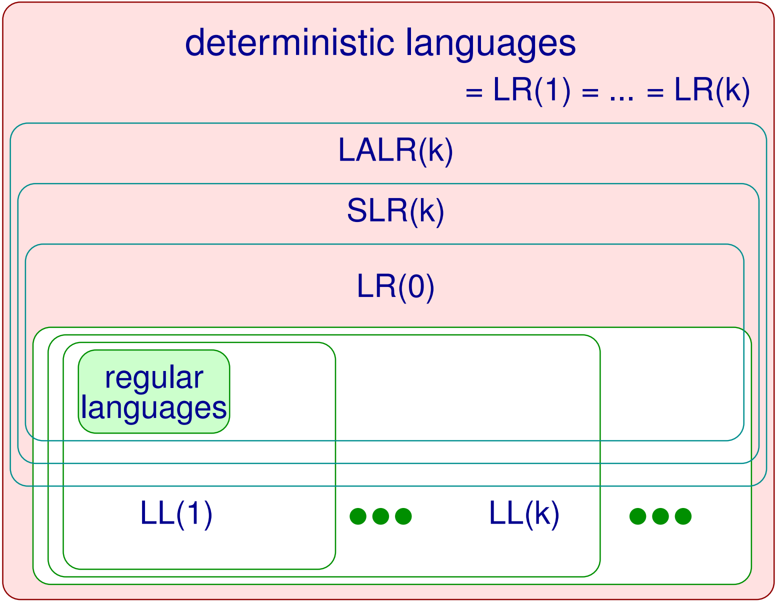
Discussion:

- Our example apparently is $LR(1)$
- In general, the canonical $LR(k)$-automaton has much more states then $LR(G) = LR(G, 0)$
- Therefore in practice, subclasses of $LR(k)$-grammars are often considered, which only use $LR(G)$ ...
- For resolving conflicts, the items are assigned special lookahead-sets:
  1. independently on the state itself $\Longrightarrow$ Simple $LR(k)$
  2. dependent on the state itself $\Longrightarrow$ $LALR(k)$

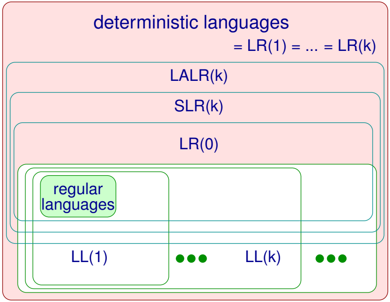
Chapter 5:

Summary

Parsing Methods



Parsing Methods



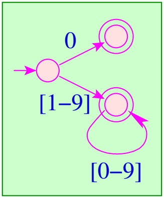
Discussion:

- All contextfree languages, that can be parsed with a deterministic pushdown automaton, can be characterized with an LR(1)-grammar.
- LR(0)-grammars describe all prefixfree deterministic contextfree languages
- The language-classes of LL(k)-grammars form a hierarchy within the deterministic contextfree languages.

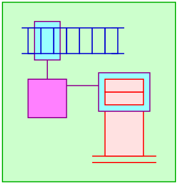
Lexical and Syntactical Analysis:

Concept of specification and implementation:

0 | [1-9][0-9]* ⇨ **Generator** ⇨

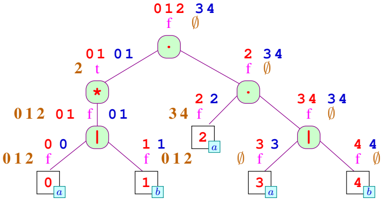


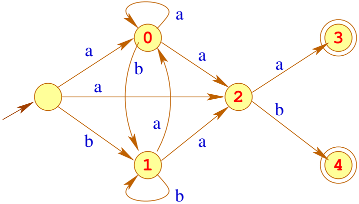
E→E{op}E ⇨ **Generator** ⇨



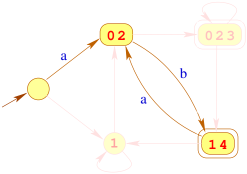
Lexical and Syntactical Analysis:

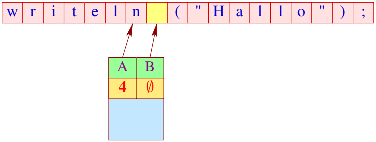
From Regular Expressions to Finite Automata





From Finite Automata to Scanners

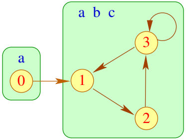


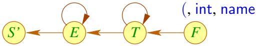


Lexical and Syntactical Analysis:

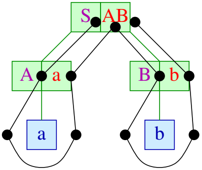
Computation of lookahead sets:

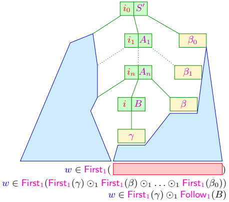
$$
\begin{array}{lcllcl}
F_\epsilon(S') & \supseteq & F_\epsilon(E) & F_\epsilon(E) & \supseteq & F_\epsilon(E) \\
F_\epsilon(E) & \supseteq & F_\epsilon(T) & F_\epsilon(T) & \supseteq & F_\epsilon(T) \\
F_\epsilon(T) & \supseteq & F_\epsilon(F) & F_\epsilon(F) & \supseteq & \{\,(\,,\mathsf{name},\mathsf{int}\}
\end{array}
$$

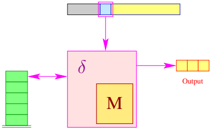




From Item-Pushdown Automata to LL(1)-Parsers:

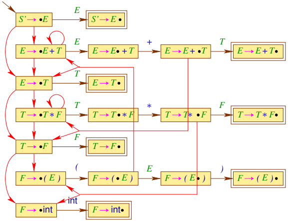


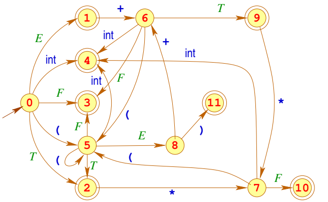




Lexical and Syntactical Analysis:

From characteristic to canonical Automata:





From Shift-Reduce-Parsers to LR(1)-Parsers: