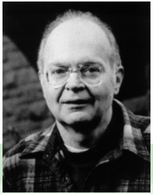
Topic:

Syntactic Analysis - Part II

Chapter 1:

Bottom-up Analysis

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:

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

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:

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

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

Shift-Reduce Parser

Construction:

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

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

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

Example-computation:

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

Shift-Reduce Parser

Observation:

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

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

- 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

The Pushdown During an RR-Derivation

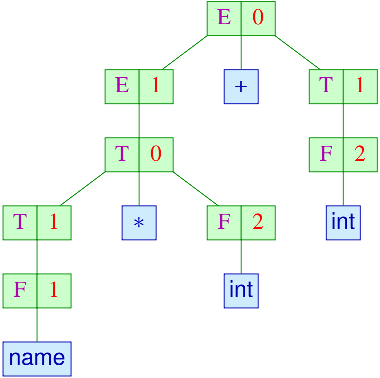
Idea: Observe a successful run of M_G^R!

Input:

counter $* \; 2 + 40$

Pushdown:

$(\; q_0 \;)$



$$
\begin{array}{lllll}
E & \rightarrow & E{+}T \;^{0} & | & T \;^{1} \\
T & \rightarrow & T{*}F \;^{0} & | & F \;^{1} \\
F & \rightarrow & (\; E \;) \;^{0} & | & \text{name} \;^{1} \quad | \quad \text{int} \;^{2}
\end{array}
$$

The Pushdown During an RR-Derivation

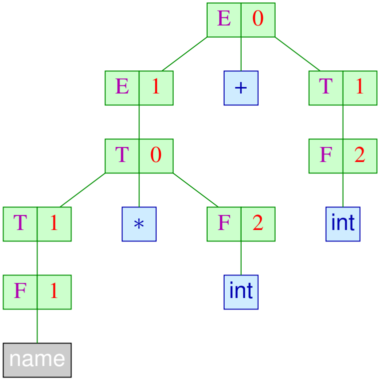
Idea: Observe a successful run of M_G^R!

Input:

$* \; 2 + 40$

Pushdown:

(q_0 name)



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

The Pushdown During an RR-Derivation

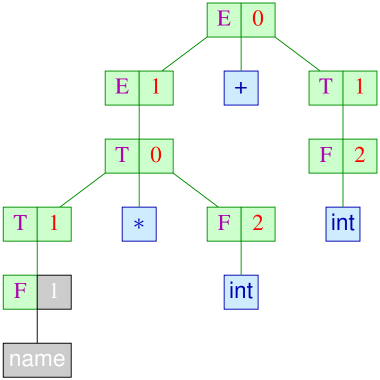
Idea: Observe a successful run of M_G^R!

Input:

$* \; 2 + 40$

Pushdown:

$(\; q_0 \; F \;)$



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

The Pushdown During an RR-Derivation

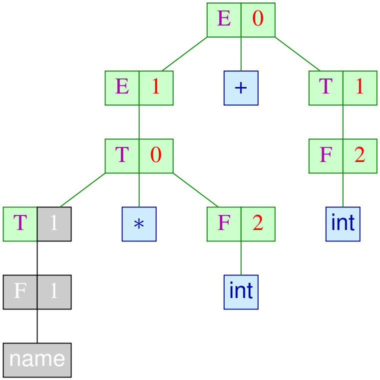
Idea: Observe a successful run of M_G^R!

Input:

$* \; 2 + 40$

Pushdown:

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



$$
\begin{array}{lcl}
E & \rightarrow & E{+}T^{\;0} \quad | \quad T^{\;1} \\
T & \rightarrow & T{*}F^{\;0} \quad | \quad F^{\;1} \\
F & \rightarrow & (\;E\;)^{\;0} \quad | \quad \text{name}^{\;1} \quad | \quad \text{int}^{\;2}
\end{array}
$$

The Pushdown During an RR-Derivation

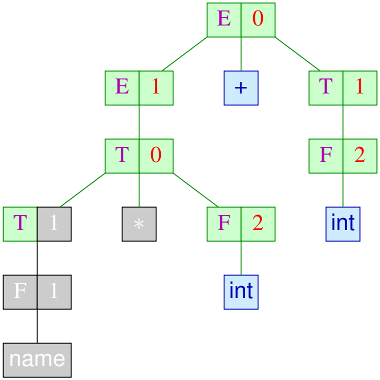
Idea: Observe a successful run of M_G^R!

Input:

$$2 + 40$$

Pushdown:

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



$$
\begin{array}{lllll}
E & \rightarrow & E{+}T \;^{0} & \mid & T \;^{1} \\
T & \rightarrow & T{*}F \;^{0} & \mid & F \;^{1} \\
F & \rightarrow & (\; E \;) \;^{0} & \mid & \text{name}\;^{1} \quad \mid \quad \text{int}\;^{2}
\end{array}
$$

The Pushdown During an RR-Derivation

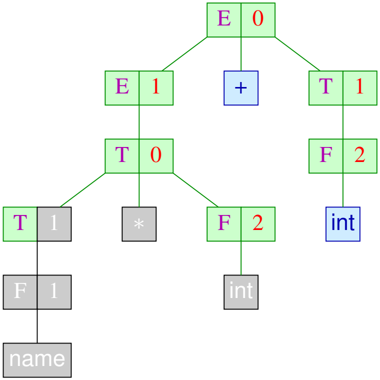
Idea: Observe a successful run of M_G^R!

Input:

$+ \; 40$

Pushdown:

$(\; q_0 \; T * \texttt{int} \;)$



$$
\begin{array}{lcl}
E & \to & E{+}T^{\;0} \;\;|\;\; T^{\;1} \\
T & \to & T{*}F^{\;0} \;\;|\;\; F^{\;1} \\
F & \to & (\;E\;)^{\;0} \;\;|\;\; \texttt{name}^{\;1} \;\;|\;\; \texttt{int}^{\;2}
\end{array}
$$

The Pushdown During an RR-Derivation

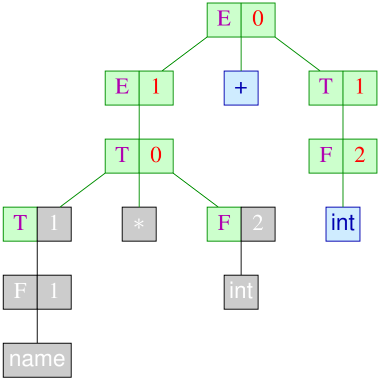
Idea: Observe a successful run of M_G^R!

Input:

$+ \; 40$

Pushdown:

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



$$
\begin{array}{lcllll}
E & \to & E{+}T\ ^0 & | & T\ ^1 & \\
T & \to & T{*}F\ ^0 & | & F\ ^1 & \\
F & \to & (\ E\ )\ ^0 & | & \text{name}\ ^1 & | \quad \text{int}\ ^2
\end{array}
$$

The Pushdown During an RR-Derivation

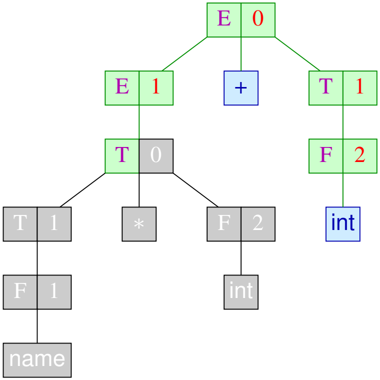
Idea: Observe a successful run of M_G^R!

Input:

$+ \; 40$

Pushdown:

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



$$
\begin{array}{llllllll}
E & \rightarrow & E{+}T^{\;0} & \mid & T^{\;1} & & \\
T & \rightarrow & T{*}F^{\;0} & \mid & F^{\;1} & & \\
F & \rightarrow & (\;E\;)^{\;0} & \mid & \text{name}^{\;1} & \mid & \text{int}^{\;2}
\end{array}
$$

The Pushdown During an RR-Derivation

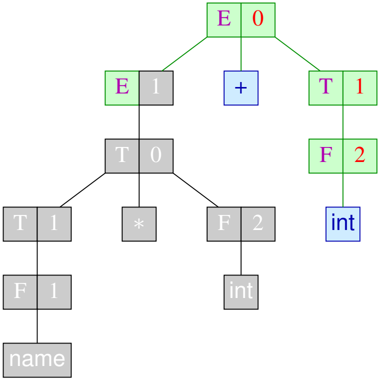
Idea: Observe a successful run of M_G^R!

Input:

$+ \; 40$

Pushdown:

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



$$
\begin{array}{lllll}
E & \to & E{+}T^{\;0} & | & T^{\;1} \\
T & \to & T{*}F^{\;0} & | & F^{\;1} \\
F & \to & (\;E\;)^{\;0} & | & \text{name}^{\;1} \quad | \quad \text{int}^{\;2}
\end{array}
$$

The Pushdown During an RR-Derivation

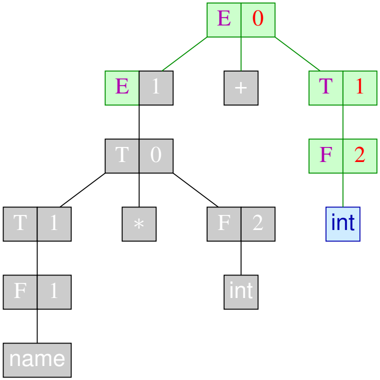
Idea: Observe a successful run of M_G^R!

Input:

40

Pushdown:

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



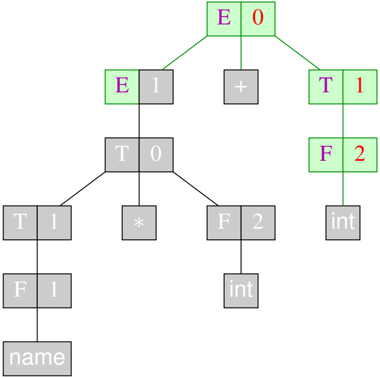
$$
\begin{array}{lcllll}
E & \rightarrow & E{+}T^{\;0} & | & T^{\;1} & \\
T & \rightarrow & T{*}F^{\;0} & | & F^{\;1} & \\
F & \rightarrow & (\;E\;)^{\;0} & | & \text{name}^{\;1} & | \quad \text{int}^{\;2}
\end{array}
$$

The Pushdown During an RR-Derivation

Idea: Observe a successful run of M_G^R!

Input:

Pushdown:
$(\; q_0 \; E + \text{int} \;)$



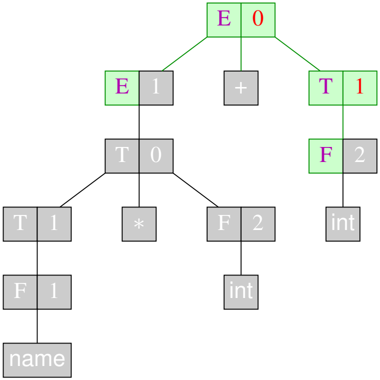
$$
\begin{array}{lcllll}
E & \rightarrow & E{+}T^{\,0} & | & T^{\,1} & \\
T & \rightarrow & T{*}F^{\,0} & | & F^{\,1} & \\
F & \rightarrow & (\; E \;)^{\,0} & | & \text{name}^{\,1} & | \;\; \text{int}^{\,2}
\end{array}
$$

The Pushdown During an RR-Derivation

Idea: Observe a successful run of M_G^R!

Input:

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



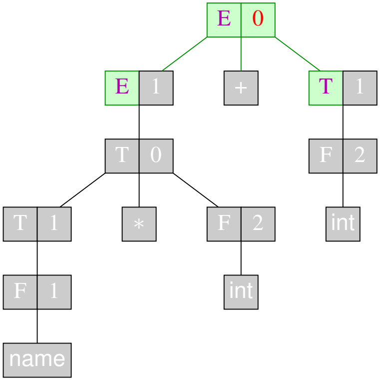
$$
\begin{array}{lclcl}
E & \rightarrow & E{+}T^{\;0} & | & T^{\;1} \\
T & \rightarrow & T{*}F^{\;0} & | & F^{\;1} \\
F & \rightarrow & (\,E\,)^{\;0} & | & \text{name}^{\;1} \quad | \quad \text{int}^{\;2}
\end{array}
$$

The Pushdown During an RR-Derivation

Idea: Observe a successful run of M_G^R!

Input:

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



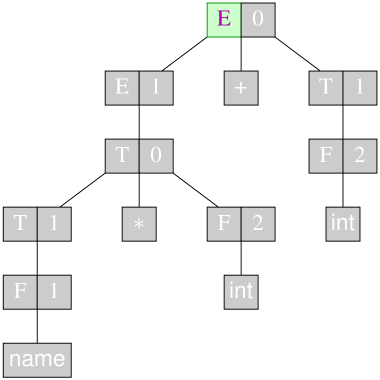
$$
\begin{array}{lllll}
E & \rightarrow & E{+}T^{\;0} & \mid & T^{\;1} \\
T & \rightarrow & T{*}F^{\;0} & \mid & F^{\;1} \\
F & \rightarrow & (\,E\,)^{\;0} & \mid & \text{name}^{\;1} \quad \mid \quad \text{int}^{\;2}
\end{array}
$$

The Pushdown During an RR-Derivation

Idea: Observe a successful run of M_G^R!

Input:

Pushdown:
$(\; q_0 \; E \;)$



$$
\begin{array}{lcl}
E & \rightarrow & E{+}T^{\;0} \quad | \quad T^{\;1} \\
T & \rightarrow & T{*}F^{\;0} \quad | \quad F^{\;1} \\
F & \rightarrow & (\;E\;)^{\;0} \quad | \quad \text{name}^{\;1} \quad | \quad \text{int}^{\;2}
\end{array}
$$

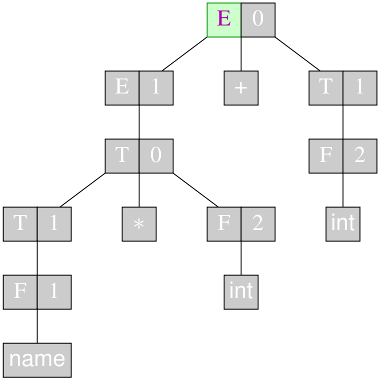
The Pushdown During an RR-Derivation

Idea: Observe a successful run of M_G^R!

Input:

Pushdown:

$(\, f \,)$



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

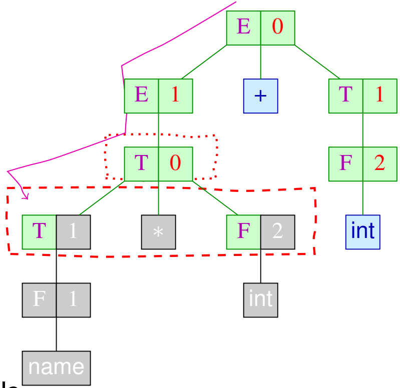
The Pushdown During an RR-Derivation

Idea: Observe a successful run of M_G^R!

Input:

$+ \; 40$

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



Result:

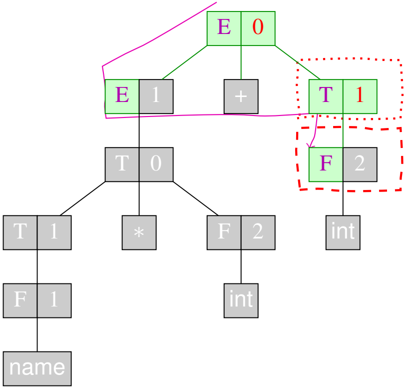
- the pushdown contains sequences of symbols, which are already processed *prefixes of righthandsides of productions* leading to the topmost few states. $\rightarrow$ documentation of the *processing history*

The Pushdown During an RR-Derivation

Idea: Observe a successful run of M_G^R!

Input:

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



Result:

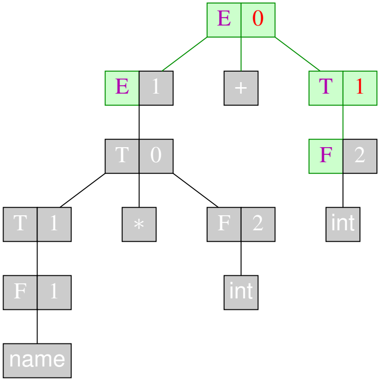
- the pushdown contains sequences of symbols, which are already processed *prefixes of righthandsides of productions* leading to the topmost few states. $\rightarrow$ documentation of the *processing history*

The Pushdown During an RR-Derivation

Idea: Observe a successful run of M_G^R!

Input:

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



Result:

- the pushdown contains sequences of symbols, which are already processed *prefixes of righthandsides of productions* leading to the topmost few states. $\rightarrow$ documentation of the *processing history*

$\rightarrow$ a righthandside on top of the pushdown is only a handle in the correct historical context

Viable Prefixes and Admissable Items

Formalism: use *Items* as representations of *prefixes of righthandsides*

Generic Agreement

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 $[B \to \gamma\bullet]$.

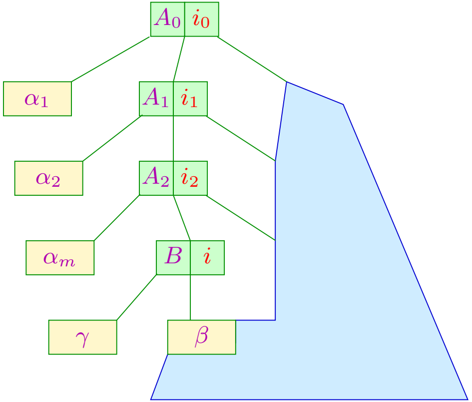
Reformulating the Shift-Reduce-Parsers main problem:

Find the items, for which the content of M_G^R's stack is the viable prefix....

$\rightarrow$ *Admissable Items*

Admissible Items

The item $[B \to \gamma \bullet \beta]$ is called admissible for $\alpha\gamma$ iff $S \to_R^* \alpha\, B\, v$:



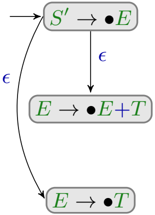
... with $\alpha = \alpha_1 \ldots \alpha_m$

Characteristic Automaton

An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

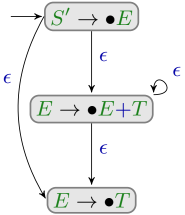
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

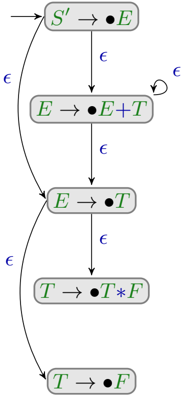
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

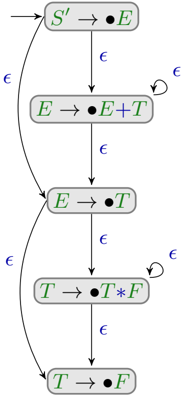
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

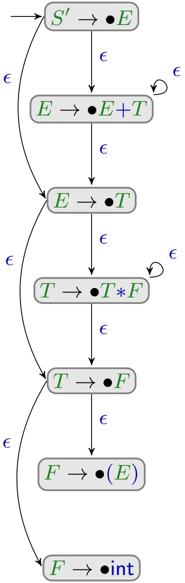
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

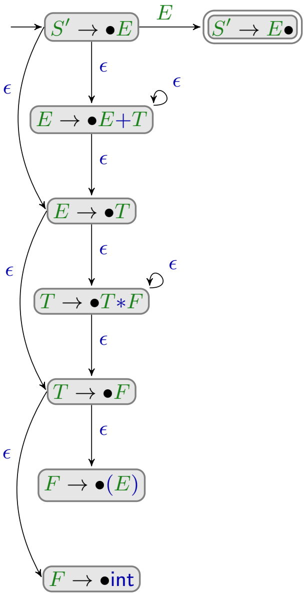
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

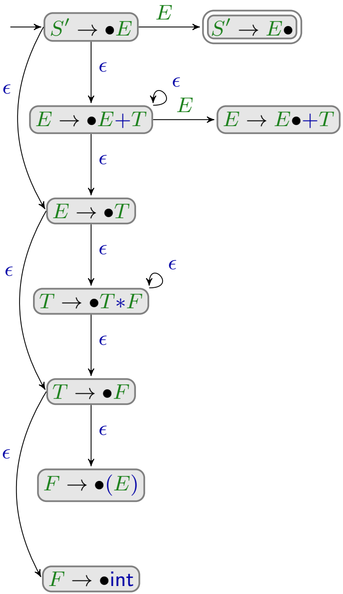
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

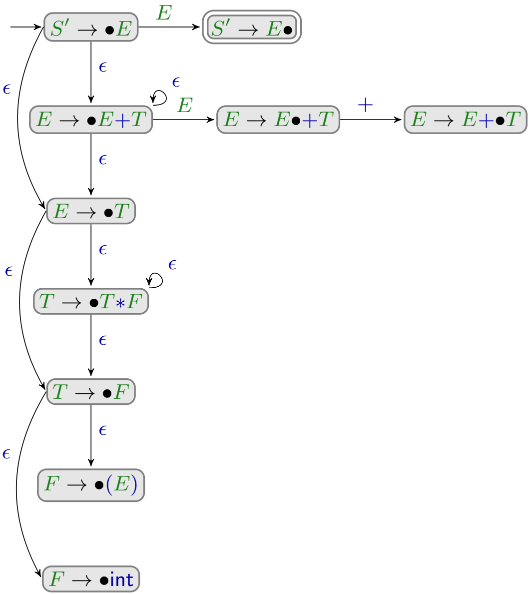
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

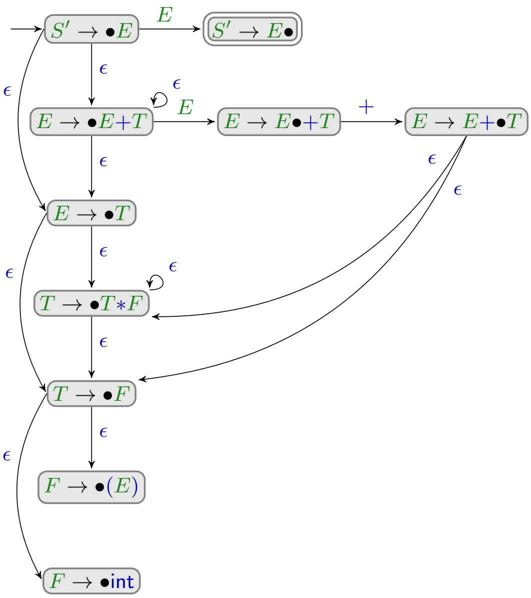
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

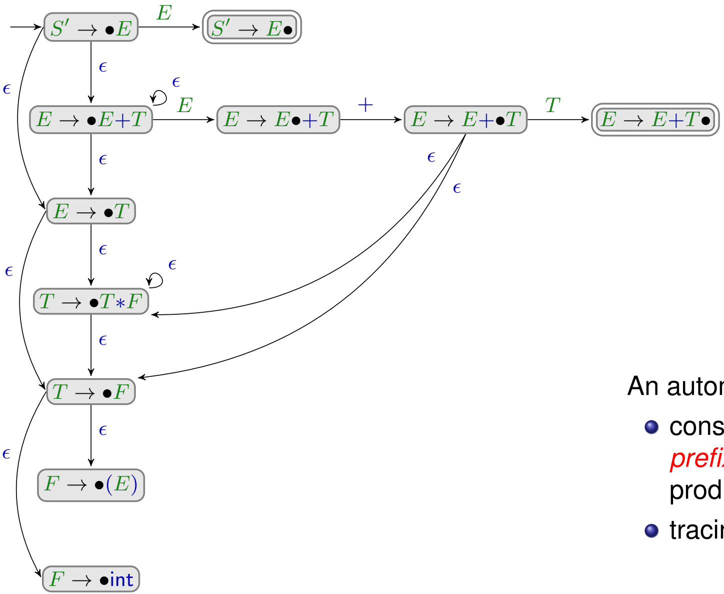
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

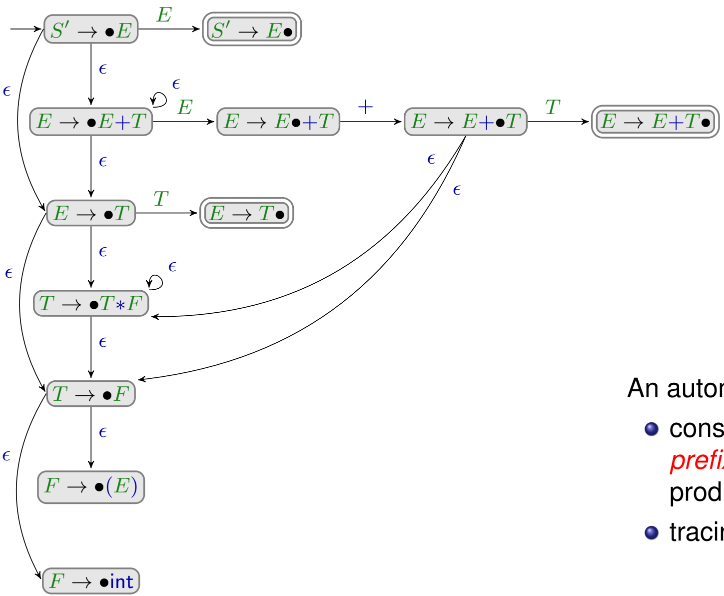
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

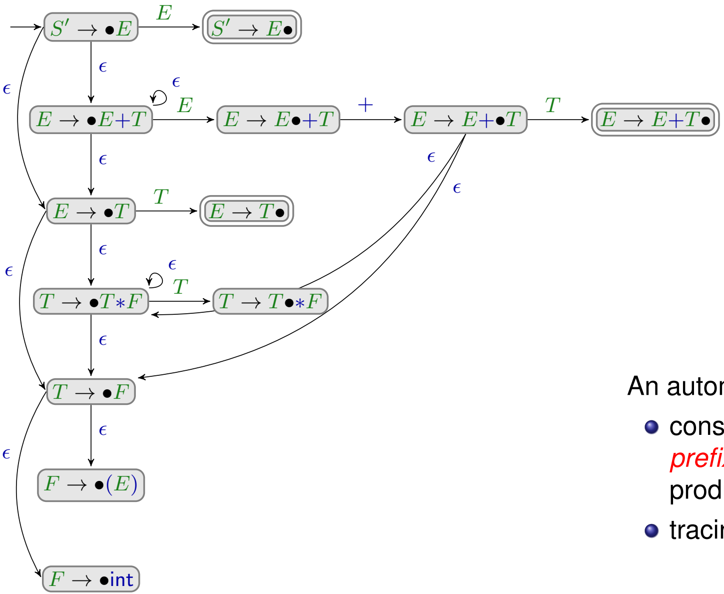
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

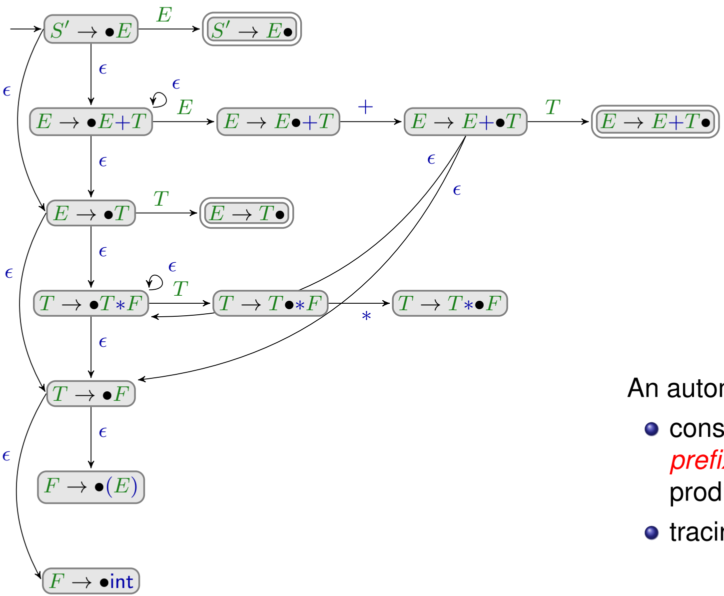
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

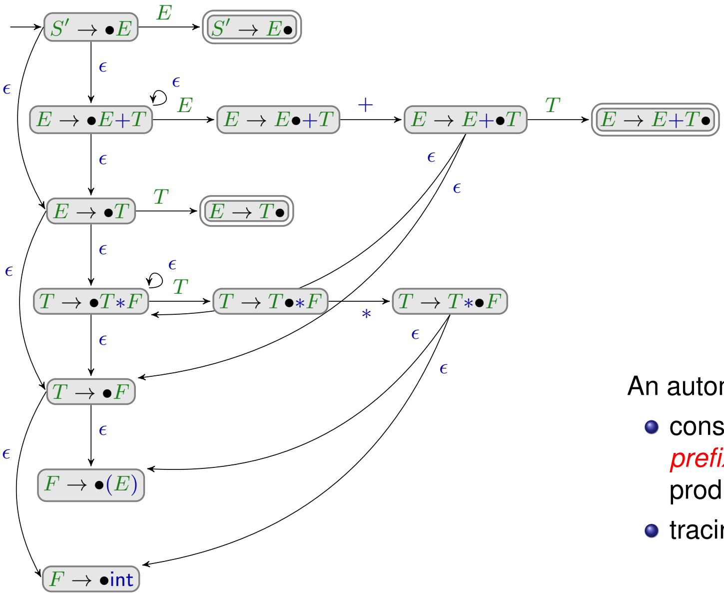
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

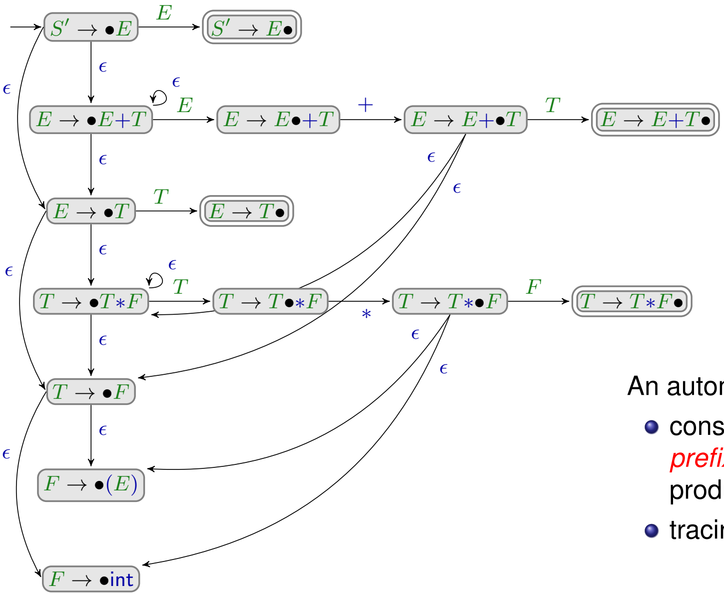
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

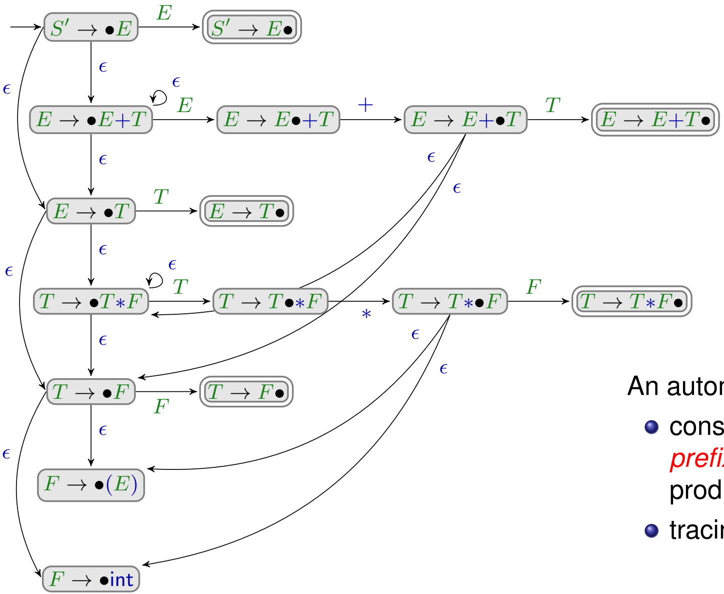
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

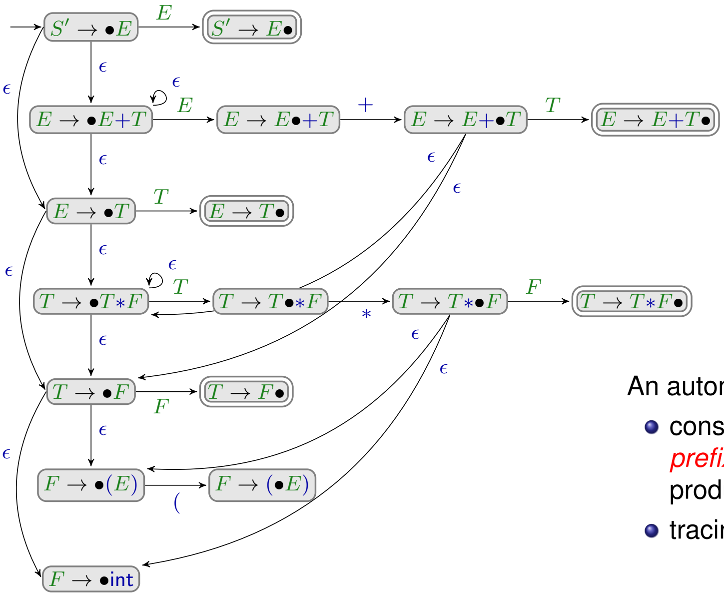
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

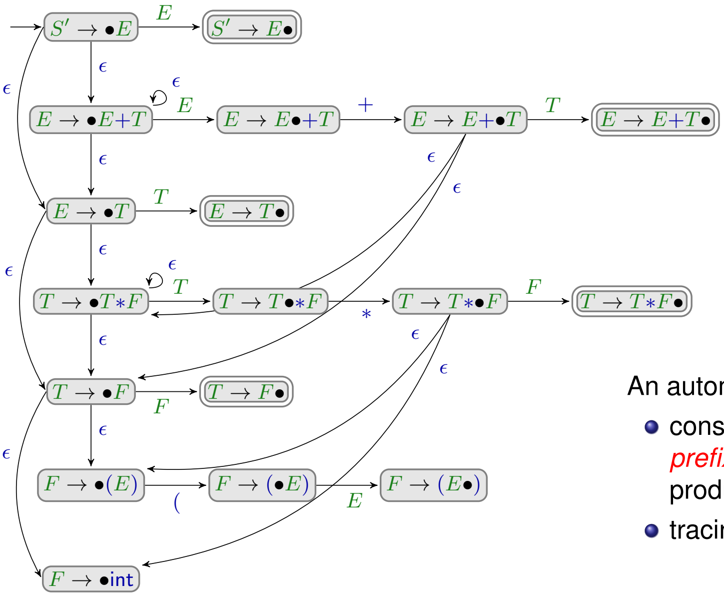
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

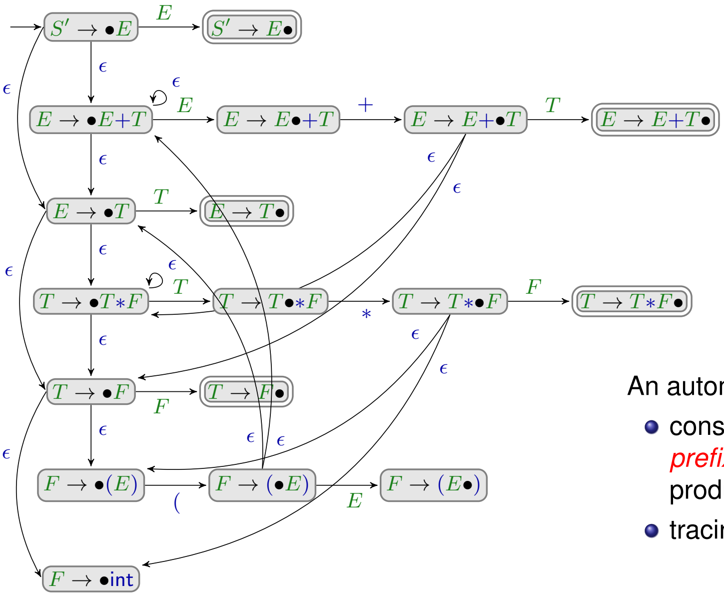
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

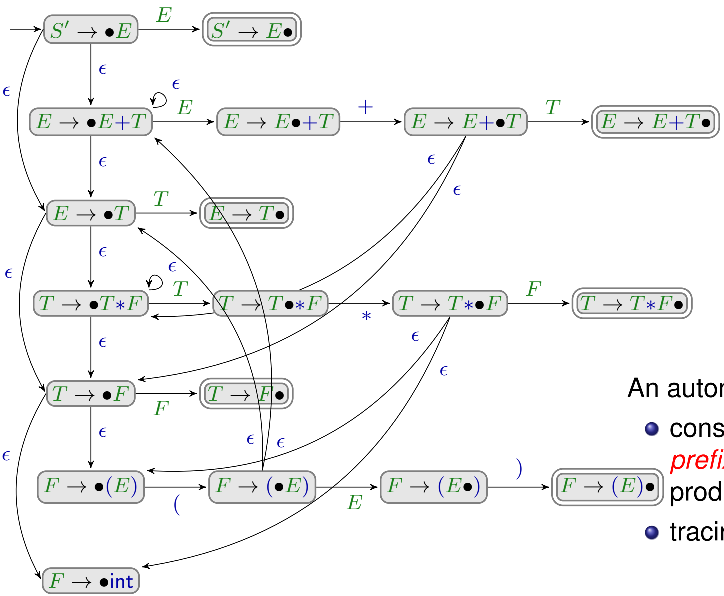
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

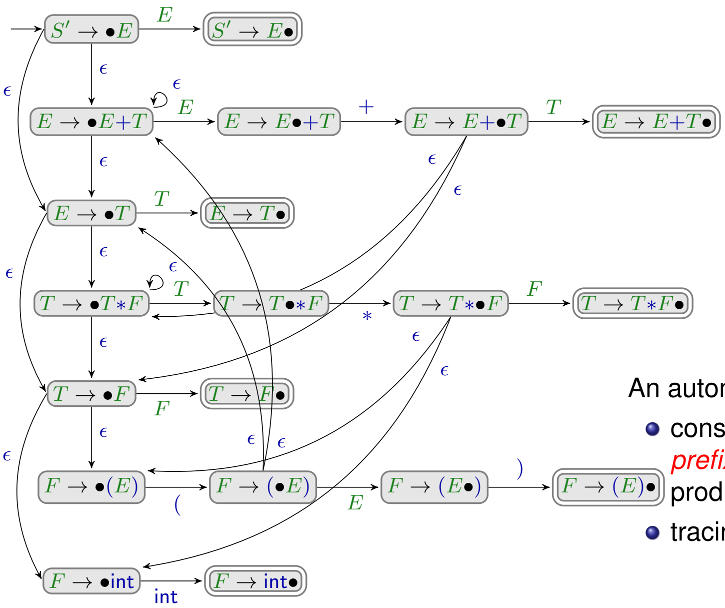
Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

Characteristic Automaton



An automaton...

- consuming pushdown symbols, i.e. *prefixes of righthandsides* of productions expanding from S
- tracing admissible items in its states

Characteristic Automaton

Observation:

One can now consume theshift-reduce parser's pushdown with the characteristic automaton: If the input $(N \cup T)^*$ for the characteristic automaton corresponds to a viable prefix, its state contains the admissible items.

States: Items

Start state: $[S' \rightarrow \bullet\, S]$

Final states: $\{[B \rightarrow \gamma \bullet] \mid B \rightarrow \gamma \in P\}$

Transitions:

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

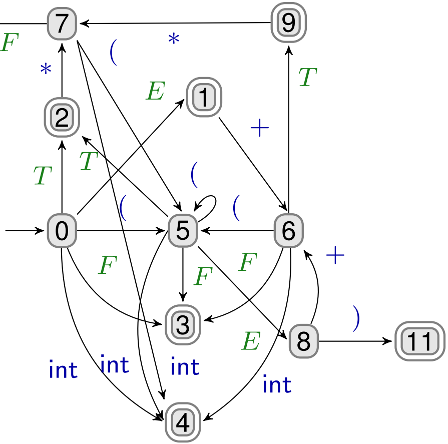
The automaton $c(G)$ is called characteristic automaton for G.

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:

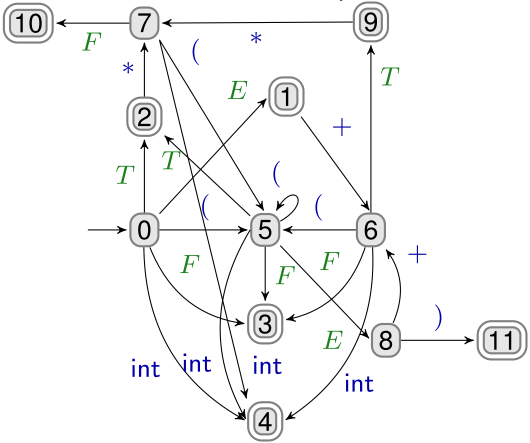


Canonical LR(0)-Automaton

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

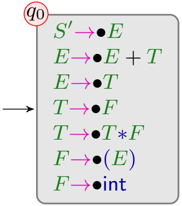
1. performing arbitrarily many ϵ-transitions after every consuming transition
2. performing the powerset construction
3. Idea: or rather apply characteristic automaton construction to powersets directly?

... for example:



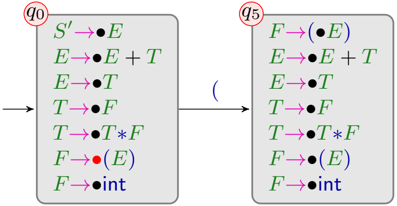
Canonical LR(0)-Automaton – Example:

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



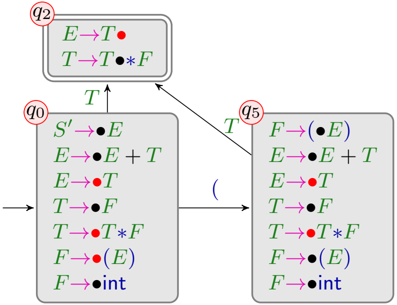
Canonical LR(0)-Automaton – Example:

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



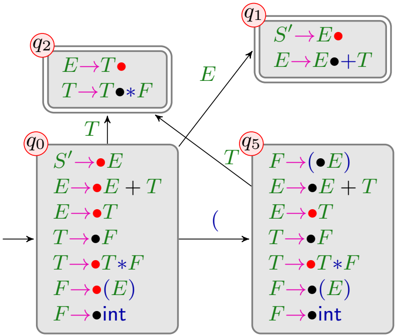
Canonical LR(0)-Automaton – Example:

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



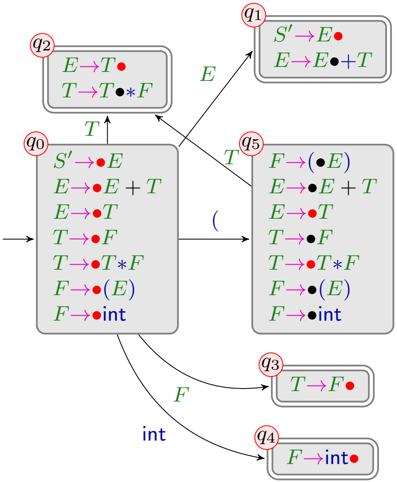
Canonical LR(0)-Automaton – Example:

$$
\begin{array}{lll}
S' & \rightarrow & E \\
E & \rightarrow & E + T \quad | \quad T \\
T & \rightarrow & T * F \quad | \quad F \\
F & \rightarrow & (\, E \,) \quad | \quad \mathsf{int}
\end{array}
$$



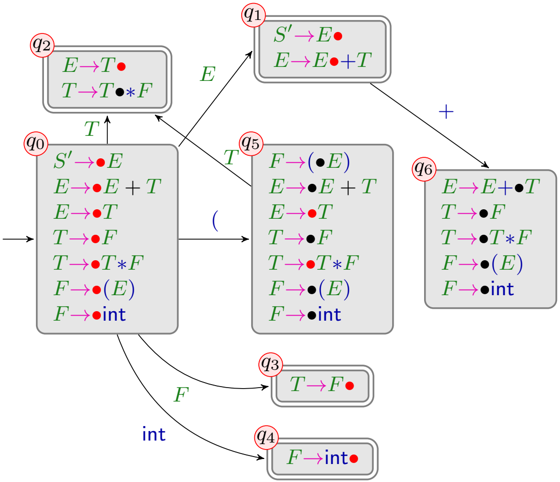
Canonical LR(0)-Automaton – Example:

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



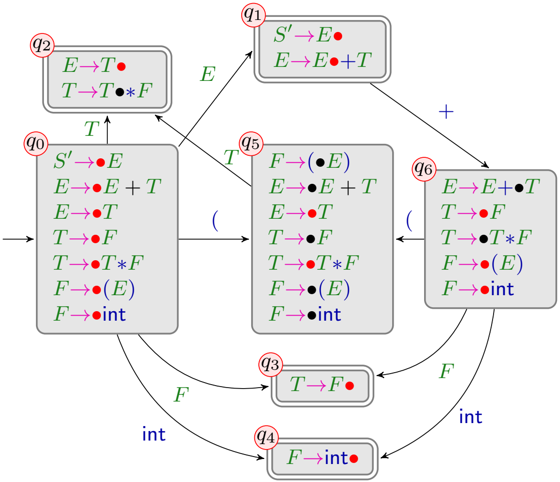
Canonical LR(0)-Automaton – Example:

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



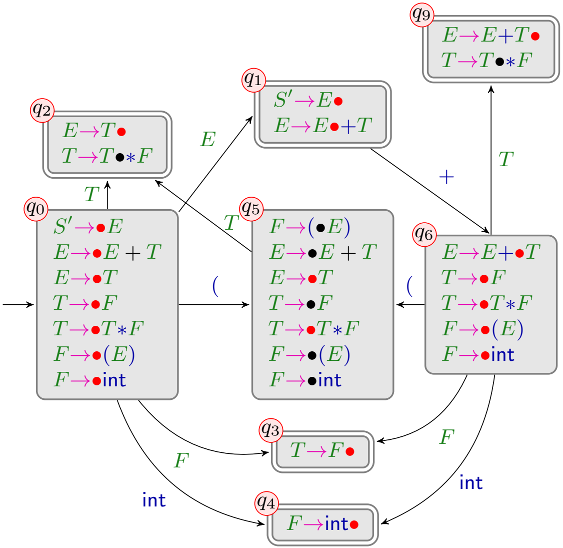
Canonical LR(0)-Automaton – Example:

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



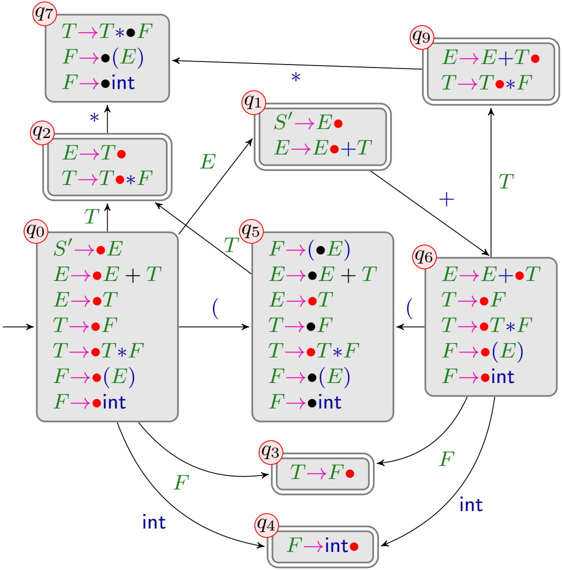
Canonical LR(0)-Automaton – Example:

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



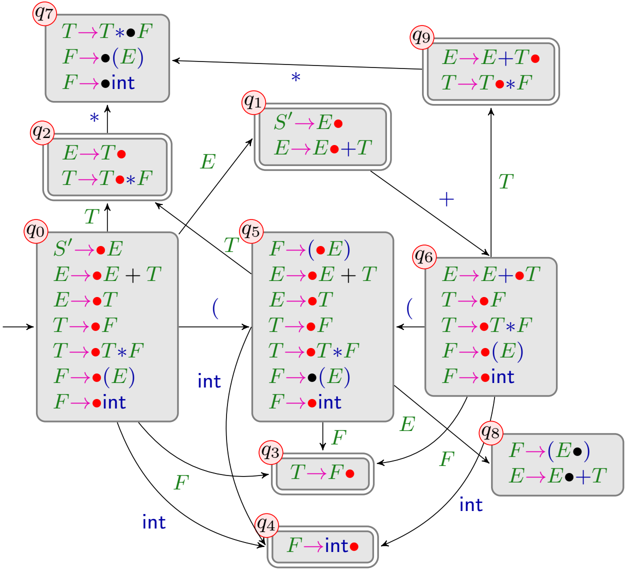
Canonical LR(0)-Automaton – Example:

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



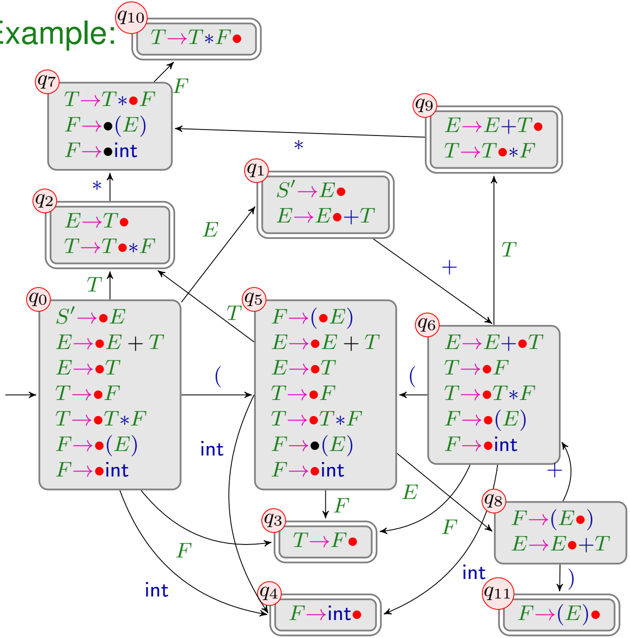
Canonical LR(0)-Automaton – Example:

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



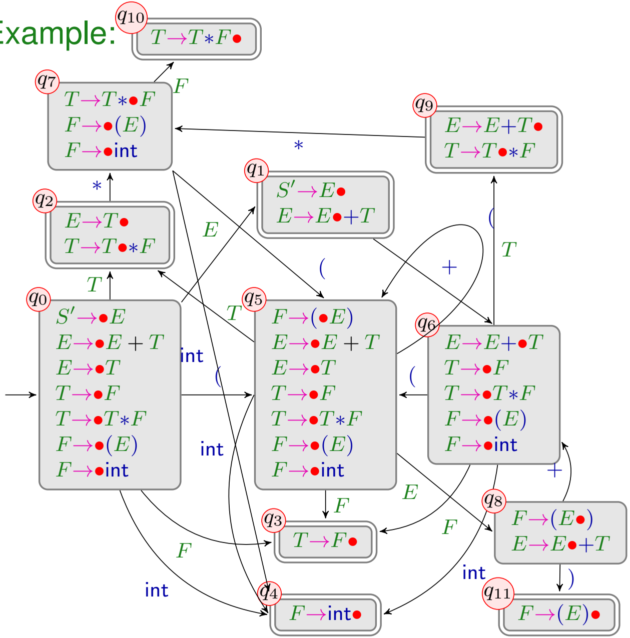
Canonical LR(0)-Automaton – Example:

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



Canonical LR(0)-Automaton – Example:

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



Canonical LR(0)-Automaton

Observation:

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

$$\delta_\epsilon^*(q) = q \cup \{[B \rightarrow \bullet\, \gamma] \mid B \rightarrow \gamma \in P, \; [A \rightarrow \alpha \bullet B'\, \beta'] \in q, \; B' \rightarrow^* B\, \beta\}$$

We define:

States: Sets of items;

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

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

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

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:

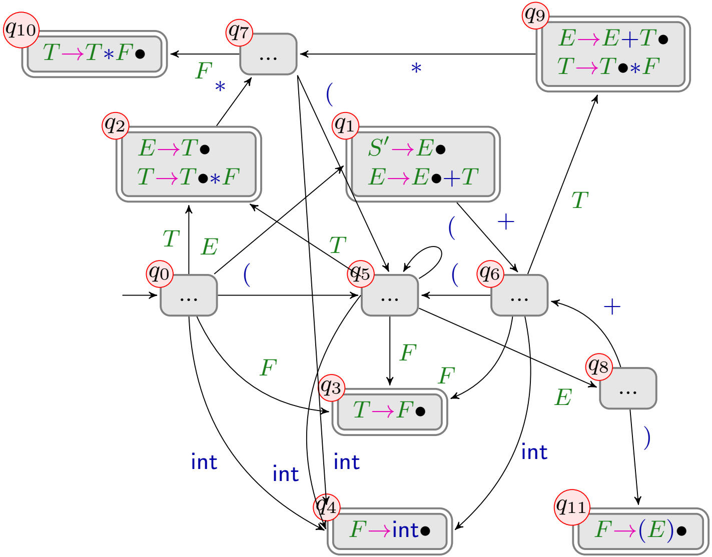
We push the states instead of the X_i in order not to process the pushdown's content with the automaton anew all the time.
Reduction with $A \to \gamma$ leads to popping the uppermost $|\gamma|$ states and continue with the state on top of the stack and input A.

Attention:

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

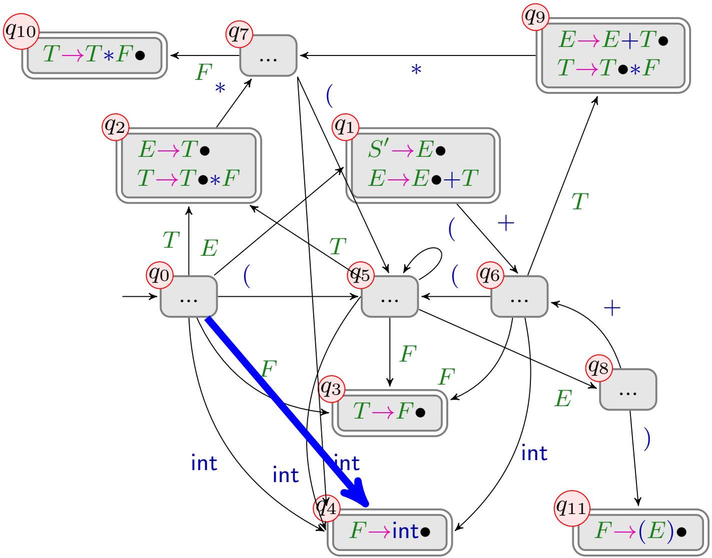
LR(0)-Parser – Example:

q_0
...



int $*$ int $+$ int

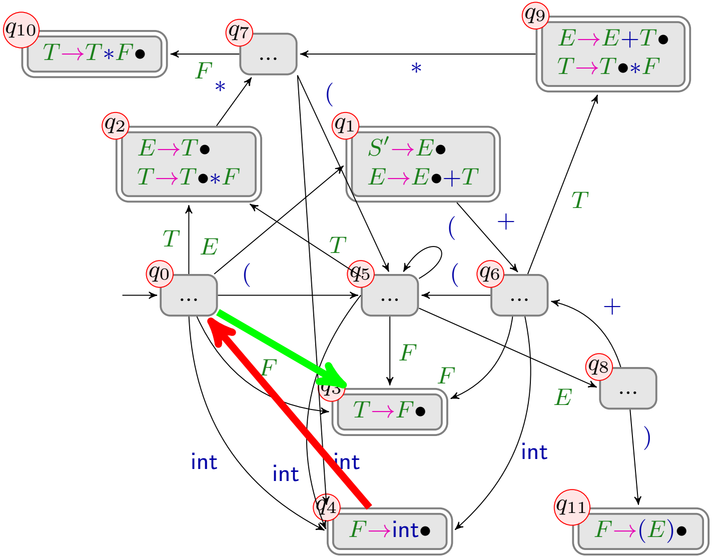
LR(0)-Parser – Example:



q_4
q_0
...

$*$ int $+$ int

LR(0)-Parser – Example:

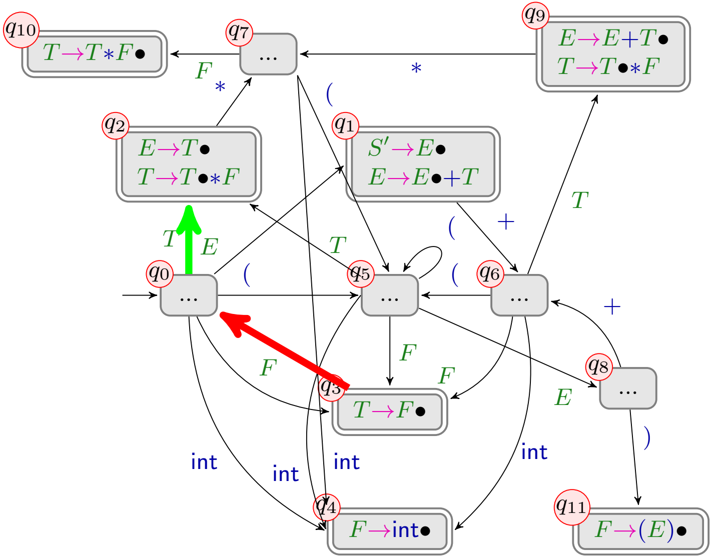


q_3
q_0
...

$*$ int $+$ int

LR(0)-Parser – Example:

q_2
q_0
...

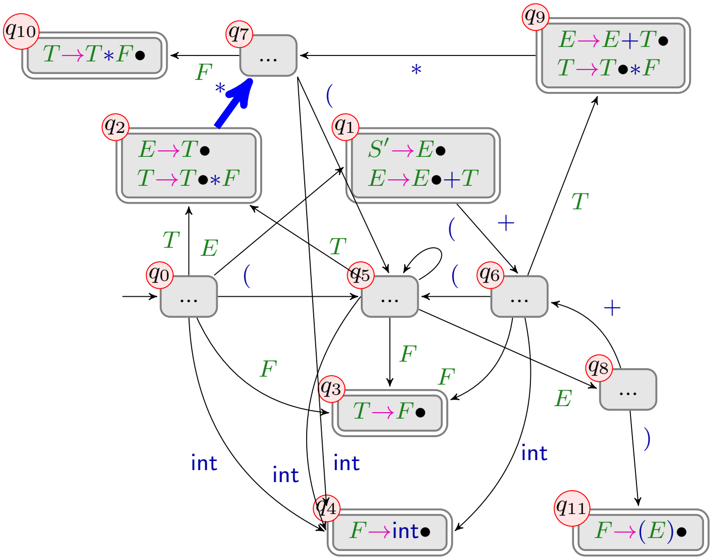


$*$ int $+$ int

LR(0)-Parser – Example:

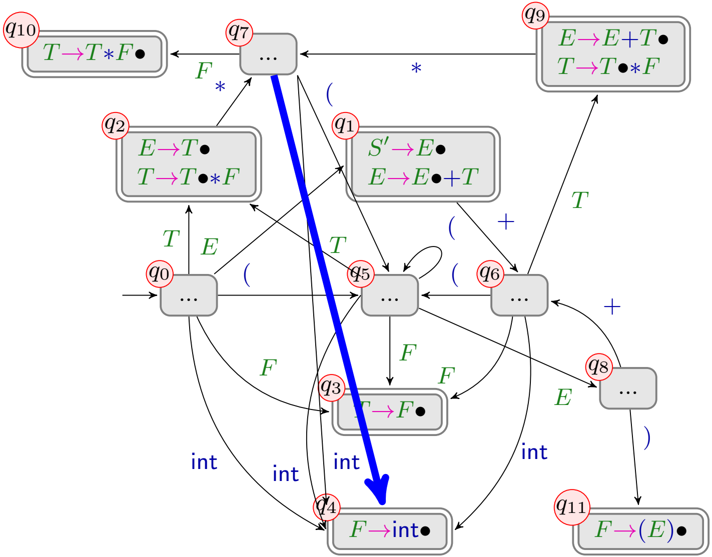
q_7
q_2
q_0
...

int + int



LR(0)-Parser – Example:

q_4
q_7
q_2
q_0
...

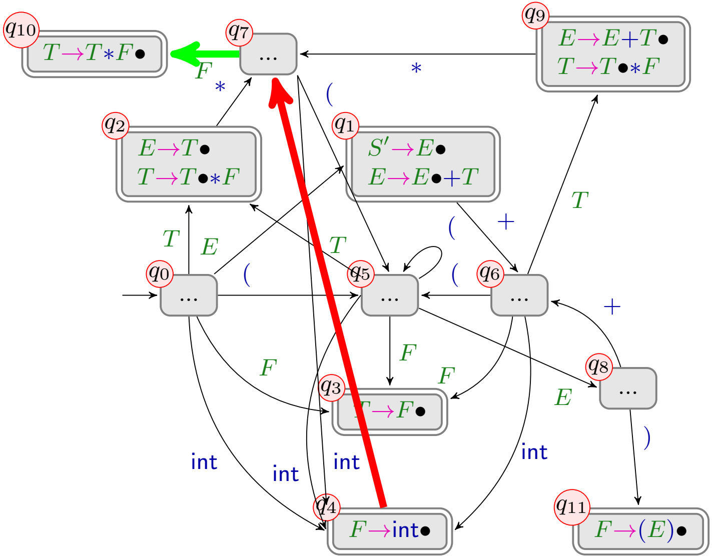


$+$ int

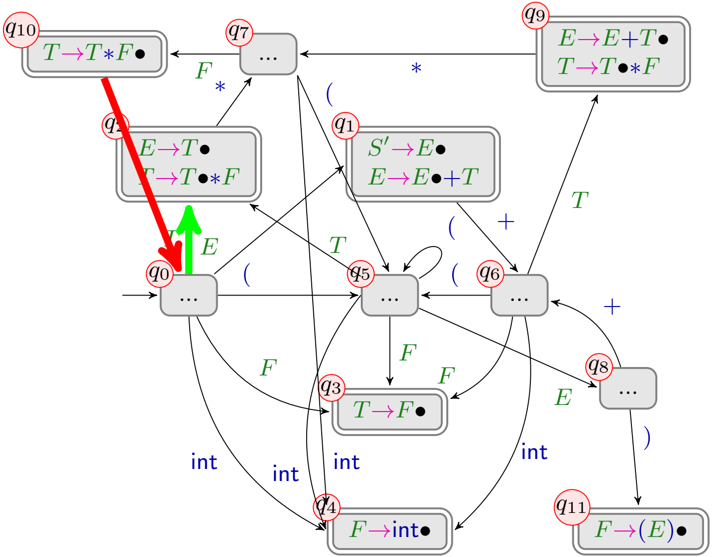
LR(0)-Parser – Example:

q_{10}
q_7
q_2
q_0
...

$+$ int



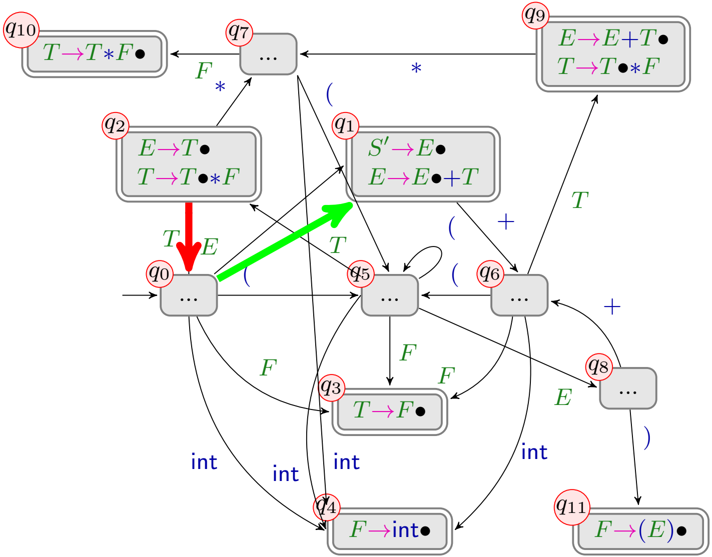
LR(0)-Parser – Example:



q_2
q_0
...

$+$ int

LR(0)-Parser – Example:



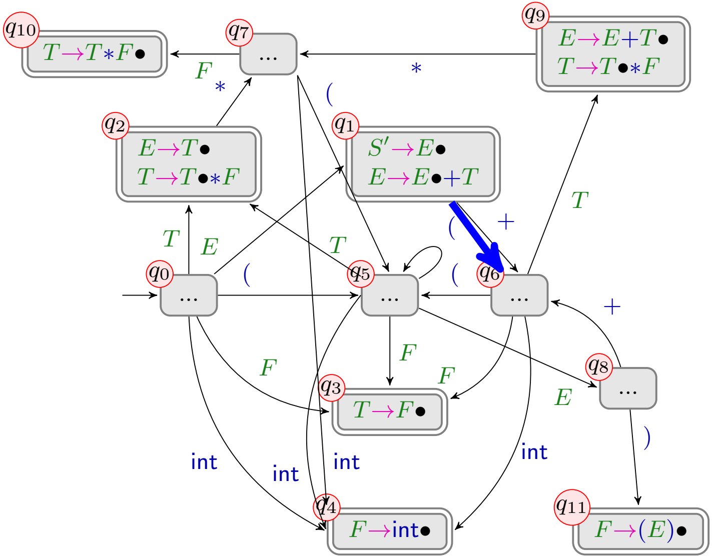
q_1
q_0
...

$+$ int

LR(0)-Parser – Example:

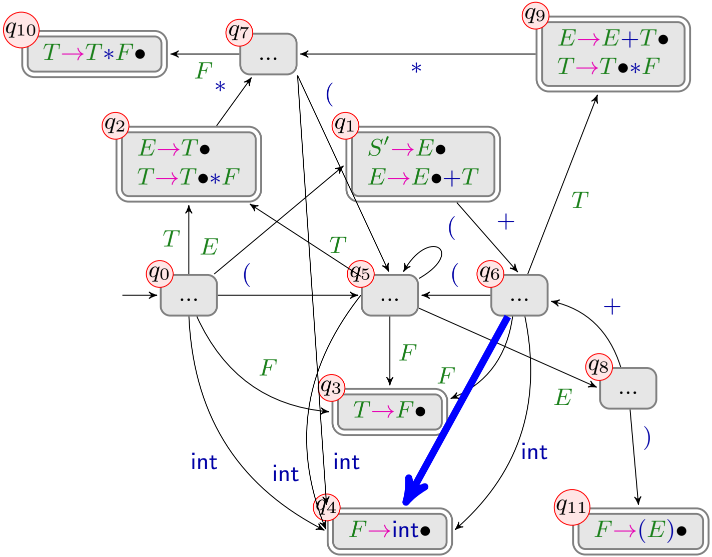
q_6
q_1
q_0
...

int



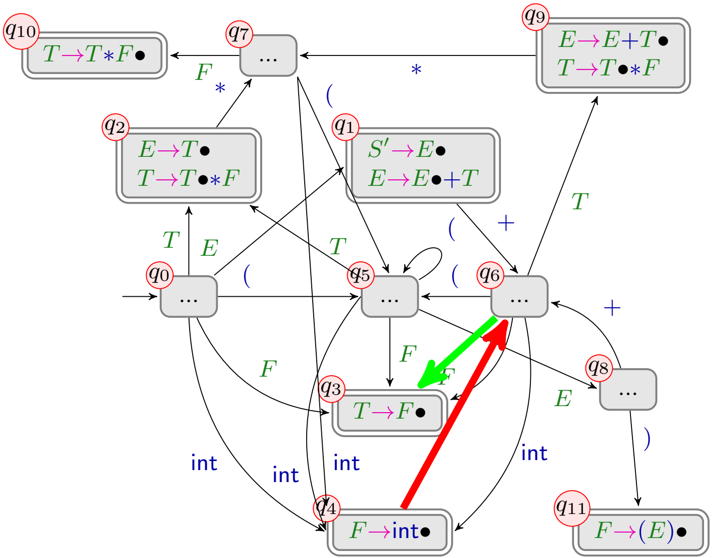
LR(0)-Parser – Example:

q_4
q_6
q_1
q_0
...



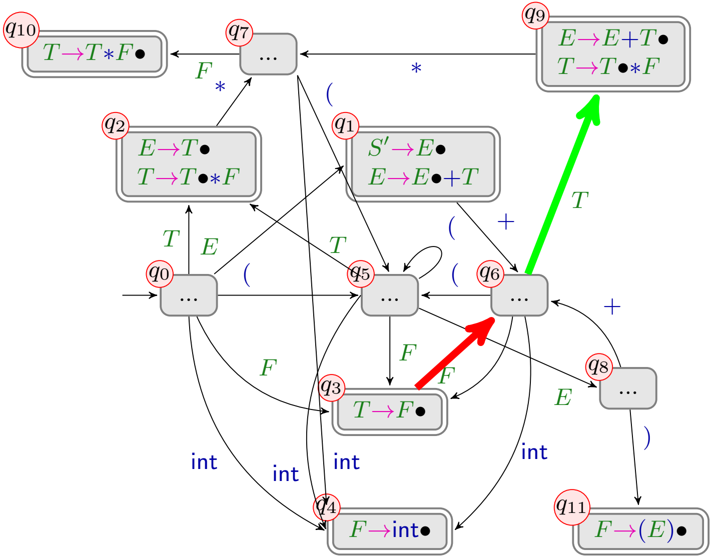
LR(0)-Parser – Example:

q_3
q_6
q_1
q_0
...



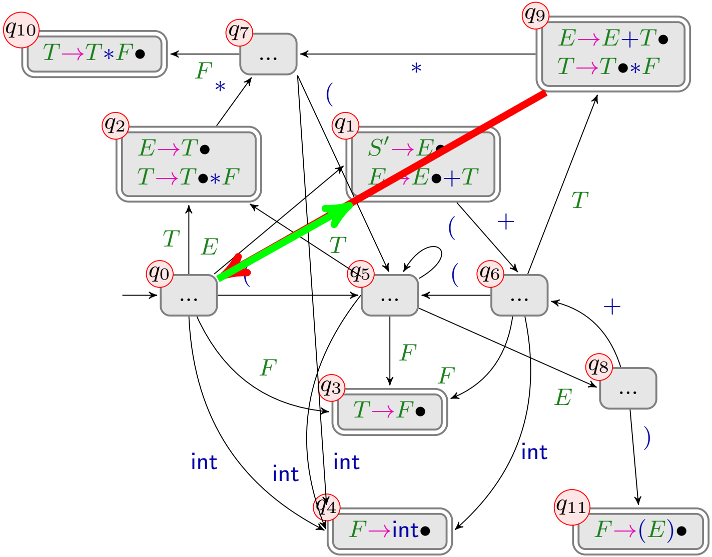
LR(0)-Parser – Example:

q_9
q_6
q_1
q_0
...



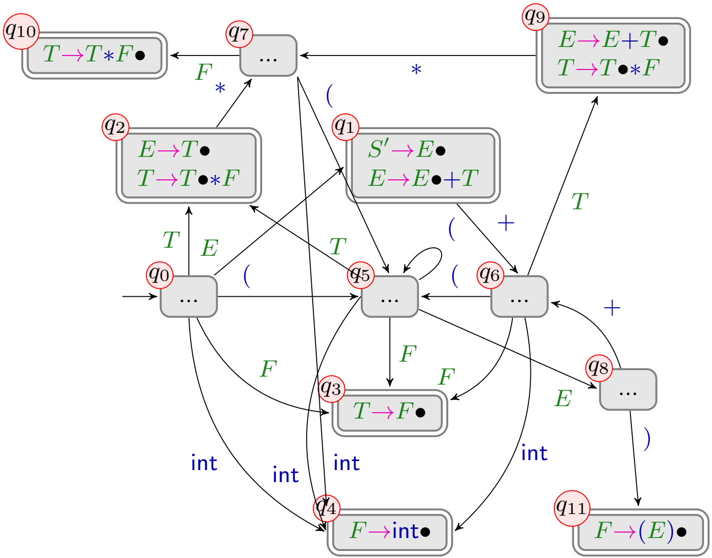
LR(0)-Parser – Example:

q_1
q_0
...



LR(0)-Parser – Example:

f
...



LR(0)-Parser

... we observe:

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: $F \to \texttt{int} \bullet$

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 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	$[A \to X_1 \ldots X_m \bullet] \in q_m, \quad q = \delta(p, A)$
Finish:	$(q_0\, p, \epsilon, f)$	if	$[S' \to S\bullet] \in p$

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

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 for a state $q \in Q$:

Reduce-Reduce-Conflict:

q:
$$A \to \gamma \bullet$$
$$A' \to \gamma' \bullet$$

with $A \neq A' \vee \gamma \neq \gamma'$

Shift-Reduce-Conflict:

q:
$$A \to \gamma \bullet$$
$$A' \to \alpha \bullet a \, \beta$$

with $a \in T$

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

Revisiting the Conflicts of the LR(0)-Automaton

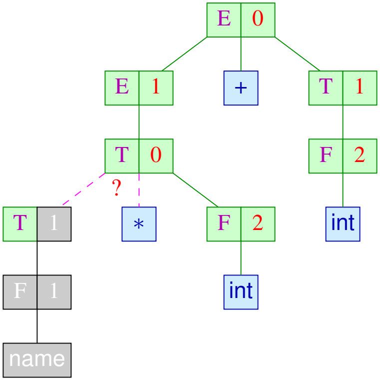
What differenciates the particular Reductions and Shifts?

Input:

$* \; 2 + 40$

Pushdown:

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



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

Revisiting the Conflicts of the LR(0)-Automaton

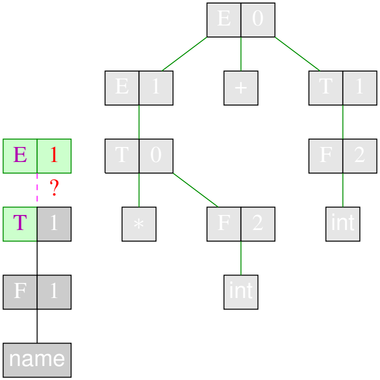
What differenciates the particular Reductions and Shifts?

Input:

$$* \; 2 + 40$$

Pushdown:

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



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

Revisiting the Conflicts of the LR(0)-Automaton

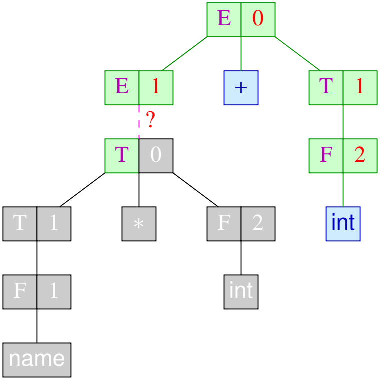
What differenciates the particular Reductions and Shifts?

Input:

$+ \; 40$

Pushdown:

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



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

Revisiting the Conflicts of the LR(0)-Automaton

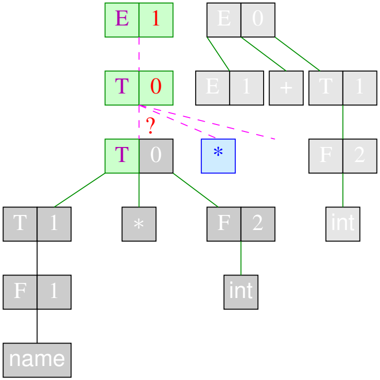
What differenciates the particular Reductions and Shifts?

Input:

$+ \; 40$

Pushdown:

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



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

Revisiting the Conflicts of the LR(0)-Automaton

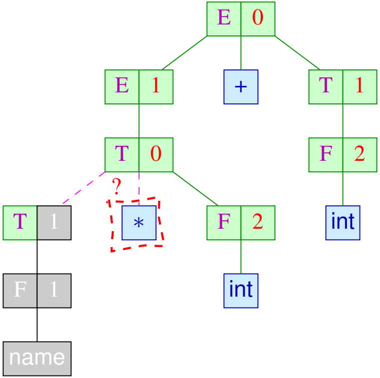
Idea: In reverse rightmost derivations, *right context* determines derivations!

Input:

$* 2 + 40$

Pushdown:

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



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

Revisiting the Conflicts of the LR(0)-Automaton

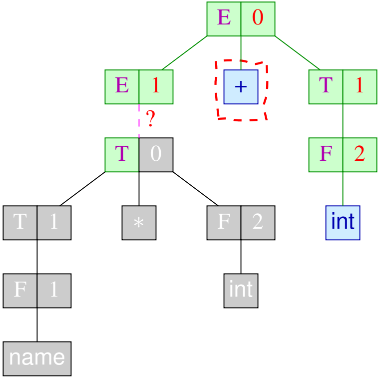
Idea: In reverse rightmost derivations, *right context* determines derivations!

Input:

$+ \; 40$

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 $\alpha\,\beta\,w\big|_{|\alpha\beta|+k} = \alpha'\,\beta'\,w'\big|_{|\alpha\beta|+k}$ with:

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

LR(k)-Grammars

Idea: Consider k-lookahead in conflict situations.

Definition:

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

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

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

1. Focus iteratively on all rightmost derivations $S \rightarrow^*_R \alpha\,X\,w \rightarrow \alpha\,\beta\,w$
2. Iterate over $k \geq 0$
   1. For each $\gamma = \alpha\,\beta w\big|_{|\alpha\beta|+k}$ (*handle with k-lookahead*) check if there exists a differently right-derivable $\alpha'\beta'w'$ for which $\gamma = \alpha'\,\beta'w'\big|_{|\alpha\beta|+k}$
   2. if there is none, we have found no objection against k being enough lookahead to disambiguate $\alpha\beta w$ from other rightmost derivations

LR(k)-Grammars

for example:

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

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 :

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

LR(k)-Grammars

for example:

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

... is not $LL(k)$ for any k :

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

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

LR(k)-Grammars

for example:

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

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

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

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

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

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 :

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

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 :

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:

(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) $S \rightarrow a\, A\, c \qquad A \rightarrow b\, b\, A \;\mid\; b$

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 $\alpha\, \underline{\beta}\, y$ is of one of these forms:

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 $\alpha\,\underline{\beta}\,y$ is of one of these forms:

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

LR(k)-Grammars

for example:

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

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

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

LR(k)-Grammars

for example:

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

Let $S \rightarrow_R^* \alpha\,X\,w \rightarrow \alpha\,\beta\,w$ 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) $S \rightarrow a\,A\,c \qquad A \rightarrow b\,A\,b \mid b$

LR(k)-Grammars

for example:

(3) $S \to a\,A\,c \qquad A \to b\,b\,A \;|\; b$ ... 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) $S \to a\,A\,c \qquad A \to b\,A\,b \;|\; b$

Consider the rightmost derivations:

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

LR(k)-Grammars

for example:

(3) $S \to a\,A\,c \qquad A \to b\,b\,A \;|\; b$ ... 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) $S \to a\,A\,c \qquad A \to b\,A\,b \;|\; b$ ... 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

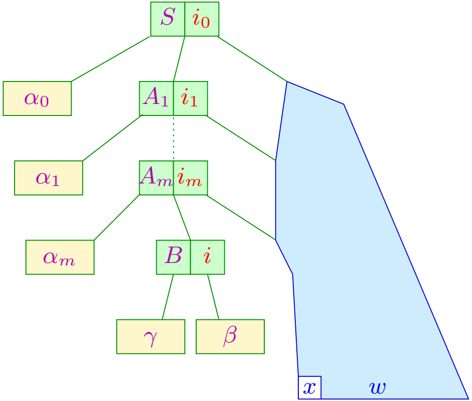
An $LR(1)$-item is a pair $[B \to \alpha \bullet \beta,\ x]$ with

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

Admissible LR(1)-Items

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

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



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

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: $[S' \to \bullet S, \$]$

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

Transitions:

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

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

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: $[S' \to \bullet\, S,\ \$]$

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 ...

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:

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

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:

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

Then, we define:

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

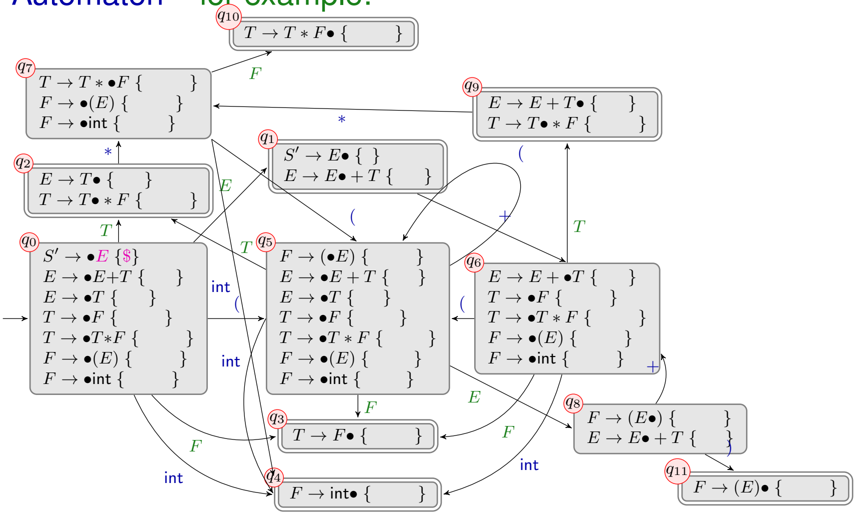
Start state: $\delta^*_\epsilon\ \{[S' \to \bullet\, S, \$]\}$

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

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

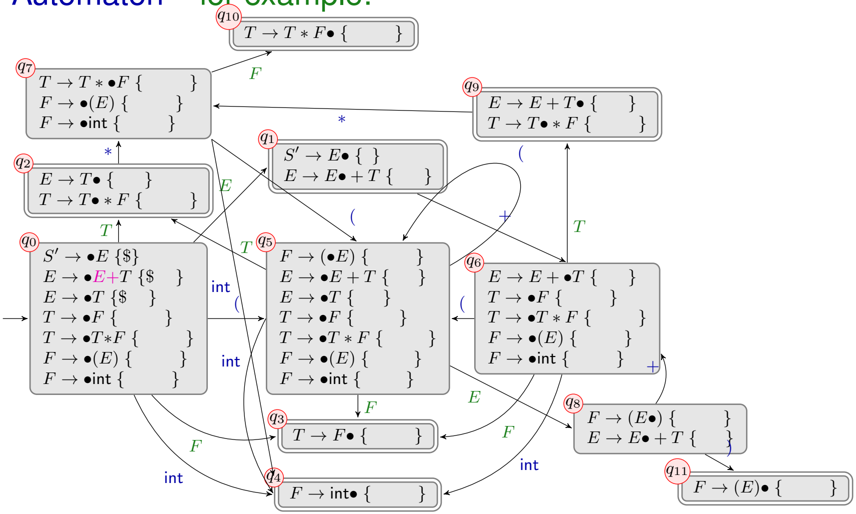
The Canonical LR(1)-Automaton – for example:

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



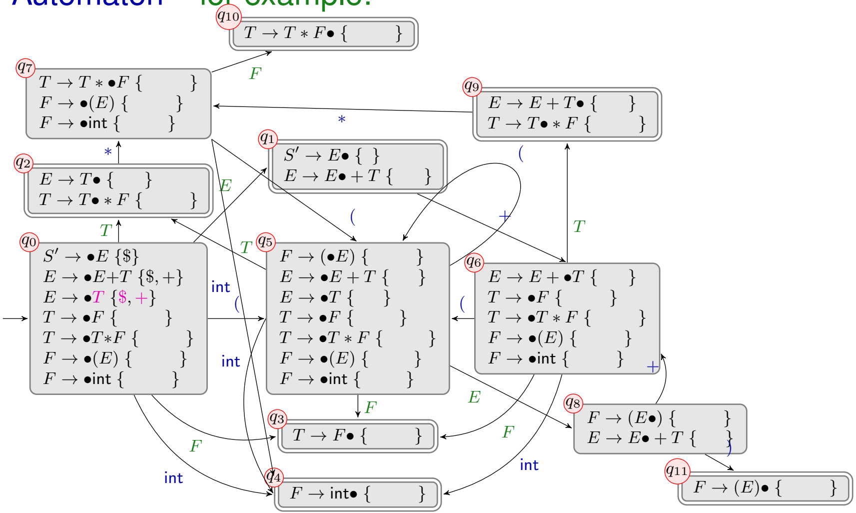
The Canonical LR(1)-Automaton – for example:

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



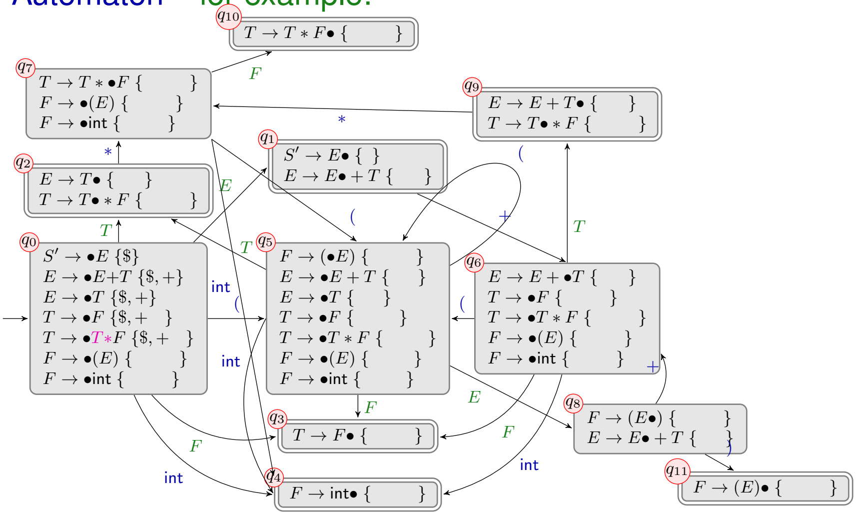
The Canonical LR(1)-Automaton – for example:

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



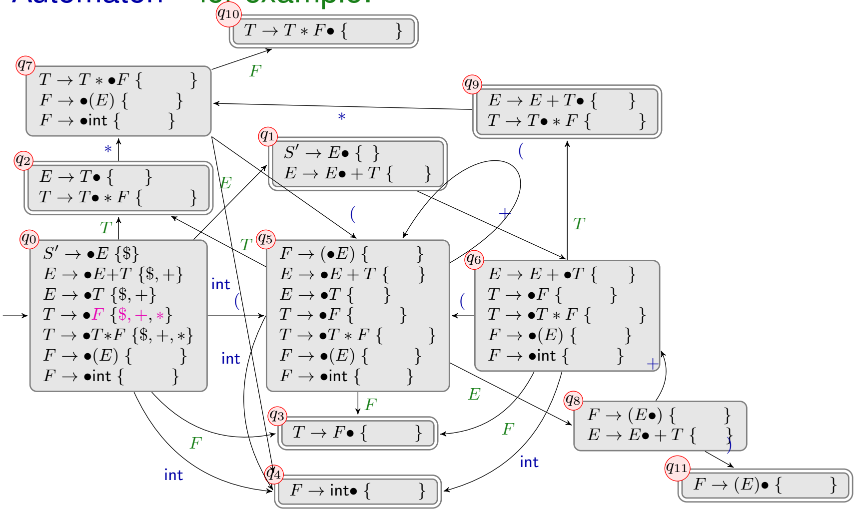
The Canonical LR(1)-Automaton – for example:

$$
\begin{array}{lcl}
S' & \to & E \\
E & \to & E + T \quad | \quad T \\
T & \to & T * F \quad | \quad F \\
F & \to & (\ E\ ) \quad | \quad \texttt{int}
\end{array}
$$



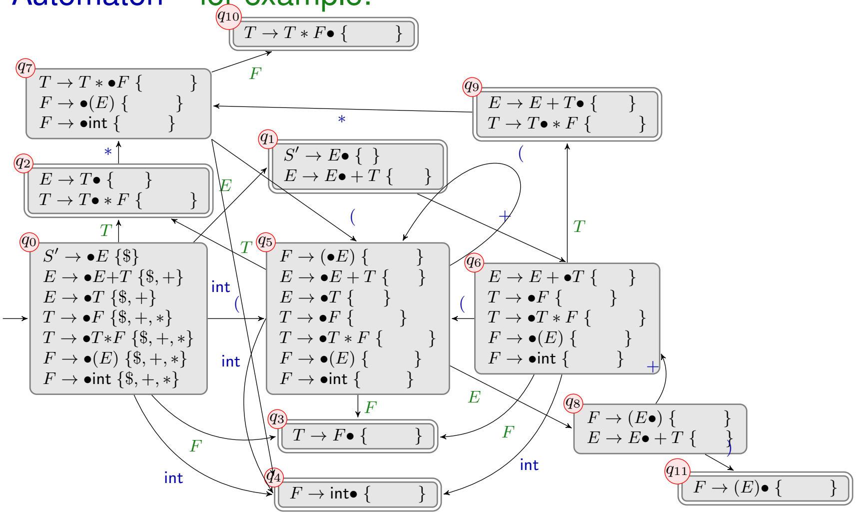
The Canonical LR(1)-Automaton – for example:

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



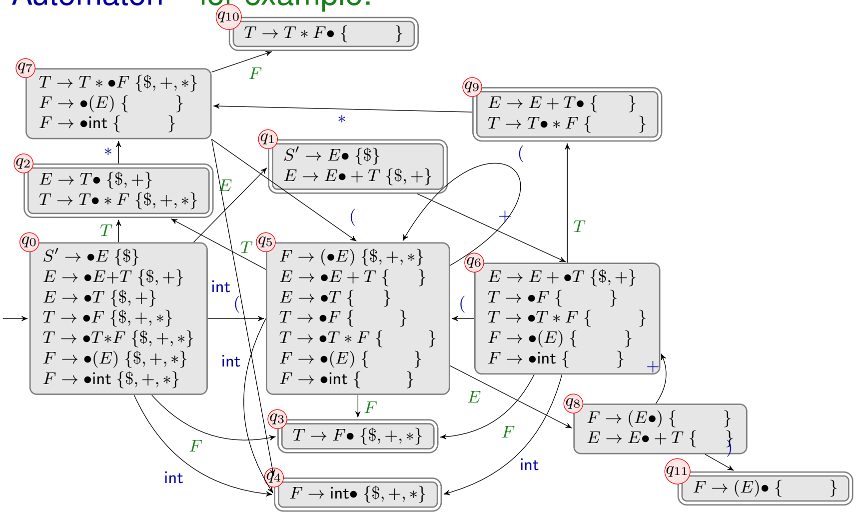
The Canonical LR(1)-Automaton – for example:

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



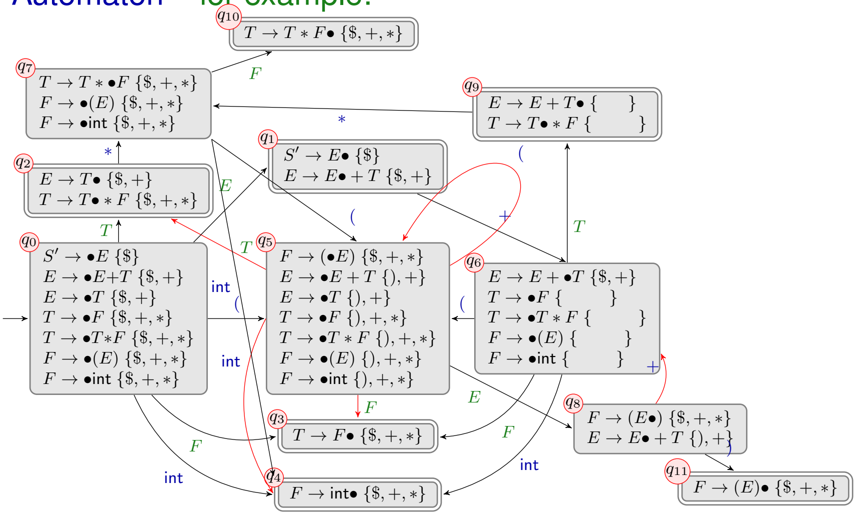
The Canonical LR(1)-Automaton – for example:

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



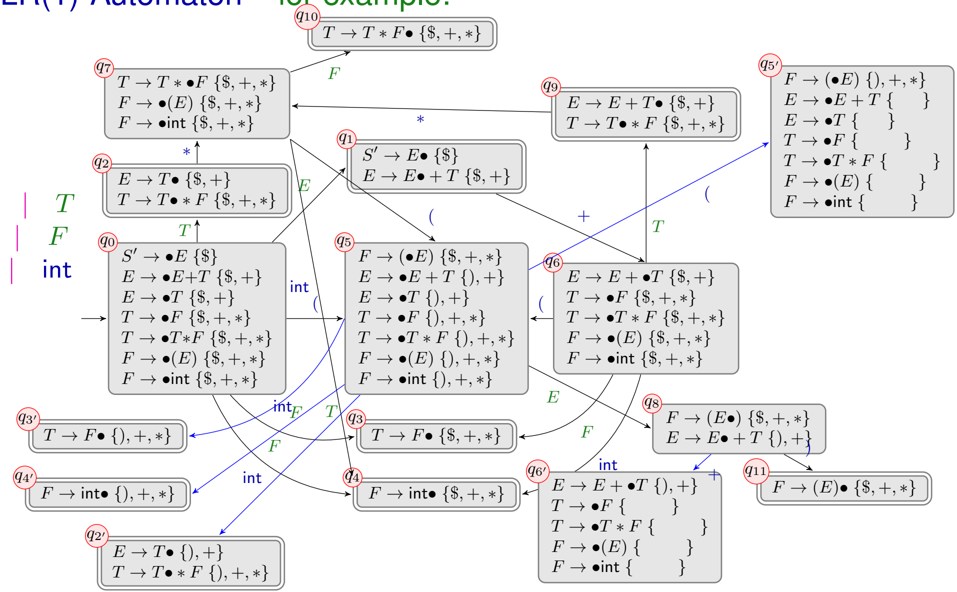
The Canonical LR(1)-Automaton – for example:

$$
\begin{array}{lcl}
S' & \to & E \\
E & \to & E + T \quad | \quad T \\
T & \to & T * F \quad | \quad F \\
F & \to & (\ E\ ) \quad | \quad \texttt{int}
\end{array}
$$



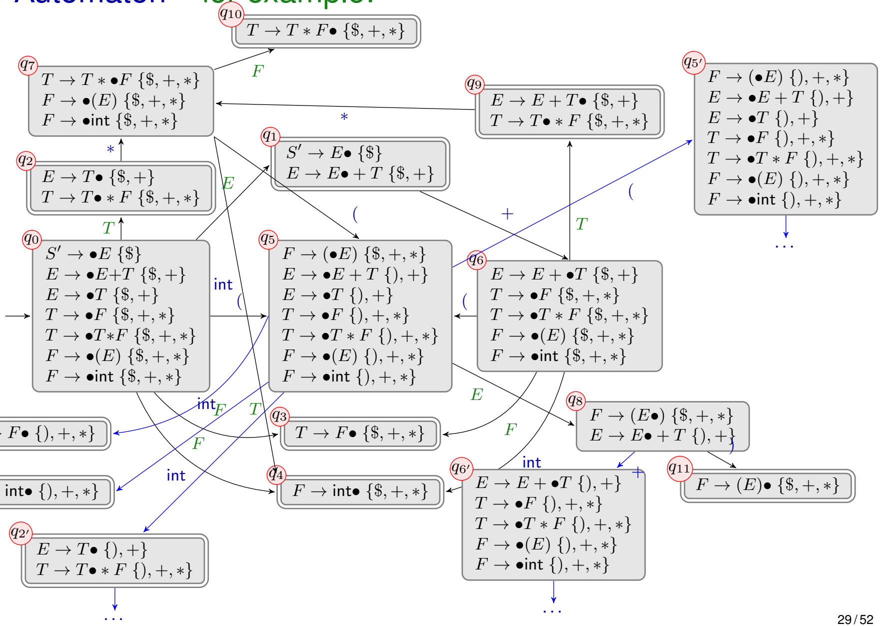
The Canonical LR(1)-Automaton – for example:

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



The Canonical LR(1)-Automaton – for example:

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



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

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

  with:

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

The Action Table:

During practical parsing, we want to represent states just via an integer id. However, when the canonical $LR(1)$-automaton reaches a final state, we want to know *how to reduce/shift*. Thus we introduce...

The construction of the action table:

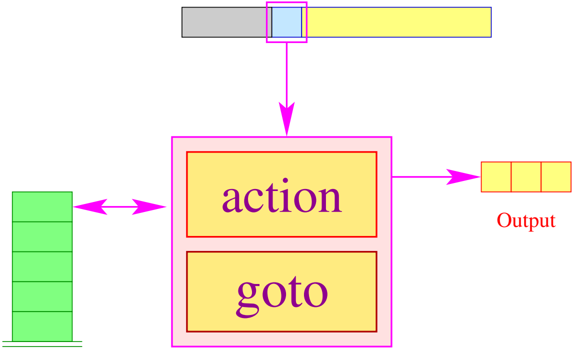
Type: $\text{action} : Q \times T \to LR(0)$-Items $\cup \{\text{s}, \text{error}\}$

Reduce: $\text{action}[q, w] = [A \to \beta \bullet]$ if $[A \to \beta \bullet, w] \in q$

Shift: $\text{action}[q, w] = \text{s}$ if $[A \to \beta \bullet b\gamma, a] \in q,\ w \in \text{First}_1(b\gamma) \odot_1 \{a\}$

Error: $\text{action}[q, w] = \text{error}$ else

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

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

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

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

The LR(1)-Parser:

Possible actions are:

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

... for example:

$$
\begin{array}{lll}
S' & \to & E \\
E & \to & E + T^{\,0} \quad | \quad T^{\,1} \\
T & \to & T * F^{\,0} \quad | \quad F^{\,1} \\
F & \to & (\ E\ )^{\,0} \quad | \quad \text{int}^{\,1}
\end{array}
$$

action	\$	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 for a state $q \in Q$:

Reduce-Reduce-Conflict:

q:

$$A \to \gamma \bullet,\ x$$
$$A' \to \gamma' \bullet,\ x$$

with $A \neq A' \vee \gamma \neq \gamma'$

Shift-Reduce-Conflict:

q:

$$A \to \gamma \bullet,\ x$$
$$A' \to \alpha' \bullet a\, \beta,\ y$$

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

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

The Canonical LR(1)-Automaton

In general:

We identify two conflicts for a state $q \in Q$:

Reduce-Reduce-Conflict:

q: $A \to \gamma \bullet,\ x$; $A' \to \gamma' \bullet,\ x$

with $A \neq A' \vee \gamma \neq \gamma'$

Shift-Reduce-Conflict:

q: $A \to \gamma \bullet,\ x$; $A' \to \alpha' \bullet a\,\beta,\ y$

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

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

Theorem:

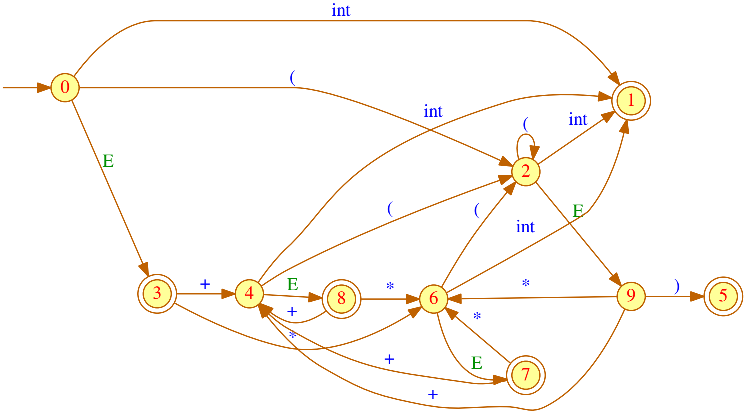
A reduced contextfree grammar G is called $LR(k)$ iff the canonical $LR(k)$-automaton $LR(G,k)$ has no $LR(k)$-unsuited states.

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:

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



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:

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

Shift-/Reduce Conflict in state 8:

$$
\begin{array}{l}
[E \to E \bullet + E^{\ 0} \qquad] \\
[E \to E + E \bullet^{\ 0} \quad , +]
\end{array}
$$

$< \gamma\, E + E\, , + \omega > \quad \Rightarrow$ *Associativity*

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	$E, 1$			$E, 1$	?	?
q_8	$E, 0$			$E, 0$	?	?
q_9	s			s	s	s

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 state 8:

$$[E \to E \bullet + E \;^{0} \quad]$$
$$[E \to E + E \bullet \;^{0} \quad , +]$$

$< \gamma\, E + E \,,\, + \omega > \quad \Rightarrow$ *Associativity*

$+$ *left associative*

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	$E, 1$			$E, 1$	?	?
q_8	$E, 0$			$E, 0$	$E, 0$	?
q_9	s			s	s	s

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 state 7:

$$[E \to E \bullet * E\ ^1 \quad]$$
$$[E \to E * E \bullet\ ^1 \quad , *]$$

$< \gamma\, E * E\, , * \omega > \quad \Rightarrow$ *Associativity*

$*$ *right associative*

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	$E, 1$			$E, 1$	?	s
q_8	$E, 0$			$E, 0$	$E, 0$	?
q_9	s			s	s	s

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:

$$
\begin{array}{lcl}
S' & \to & E \;^{0} \\
E & \to & E + E \;^{0} \\
 & \mid & E * E \;^{1} \\
 & \mid & (\; E \;) \;^{2} \\
 & \mid & \texttt{int} \;^{3}
\end{array}
$$

Shift-/Reduce Conflict in states 8, 7:

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

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	$E, 1$			$E, 1$	?	s
q_8	$E, 0$			$E, 0$	$E, 0$	?
q_9	s			s	s	s

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):

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

What if precedences are not enough?

Example (very simplified lambda expressions):

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

E rightmost-derives these forms among others:

$$(\ \underline{\textsf{ident}}\ ),\ (\ \underline{\textsf{ident}}\ ) \Rightarrow \textsf{ident}\ , \ldots \quad \Rightarrow \quad \text{at least } LR(2)$$

Naive Idea:

poor man's $LR(2)$ by combining the tokens $)$ and $\Rightarrow$ during lexical analysis into a single token $)\!\Rightarrow$.

What if precedences are not enough?

Example (very simplified lambda expressions):

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

E rightmost-derives these forms among others:

$$(\ \underline{\mathsf{ident}}\ ),\ (\ \underline{\mathsf{ident}}\ ) \Rightarrow \mathsf{ident}\ , \ldots \quad \Rightarrow \quad \text{at least } LR(2)$$

Naive Idea:

poor man's $LR(2)$ by combining the tokens) and ⇒ during lexical analysis into a single token)⇒.

⚠ in this case obvious solution, but in general not so simple

What if precedences are not enough?

In practice, $LR(k)$-parser generators working with the lookahead sets of sizes larger then $k = 1$ are not common, since computing lookahead sets with $k > 1$ blows up exponentially. However,

1. there exist several practical $LR(k)$ grammars of $k > 1$,
   e.g. Java 1.6+ ($LR(2)$), ANSI C, etc.
2. often, more lookahead is only exhausted locally
3. should we really give up, whenever we are confronted with a Shift-/Reduce-Conflict?

What if precedences are not enough?

In practice, $LR(k)$-parser generators working with the lookahead sets of sizes larger then $k = 1$ are not common, since computing lookahead sets with $k > 1$ blows up exponentially. However,

1. there exist several practical $LR(k)$ grammars of $k > 1$, e.g. Java 1.6+ ($LR(2)$), ANSI C, etc.
2. often, more lookahead is only exhausted locally
3. should we really give up, whenever we are confronted with a Shift-/Reduce-Conflict?

Victor Schneider

Dennis Mickunas

Theorem: $LR(k)$-to-$LR(1)$

Any $LR(k)$ grammar can be directly transformed into an equivalent $LR(1)$ grammar.

LR(2) to LR(1)

... Example:

$$\begin{array}{lll} S & \to & A\, b\, b^{0} \mid B\, b\, c^{1} \\ A & \to & a\, A^{0} \mid a^{1} \\ B & \to & a\, B^{0} \mid a^{1} \end{array}$$

LR(2) to LR(1)

... Example:

$$\begin{array}{lcl} S & \to & A\,b\,b^{0} \mid B\,b\,c^{1} \\ A & \to & a\,A^{0} \mid a^{1} \\ B & \to & a\,B^{0} \mid a^{1} \end{array}$$

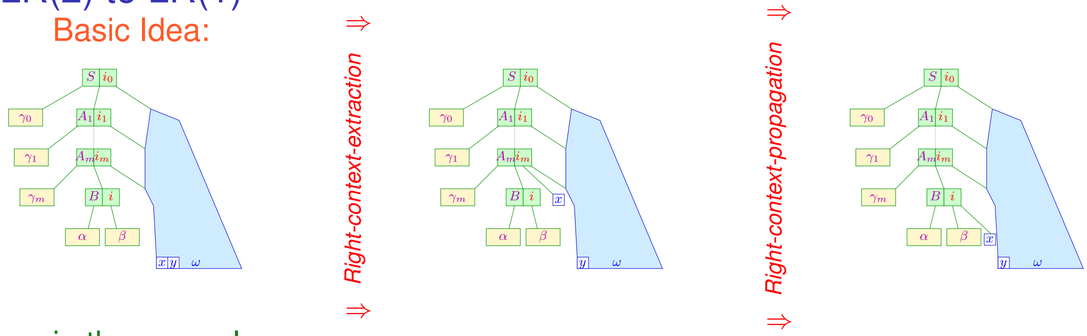
S rightmost-derives one of these forms:

$$a^n\underline{a}bb\ ,\ a^n\underline{a}bc\ ,\ a^n\underline{a\,A}bb\ ,\ a^n\underline{a\,B}bc,\ \underline{A}bb\ ,\ \underline{B}bc \quad \Rightarrow \quad LR(2)$$

in $LR(1)$, you will have Reduce-/Reduce-Conflicts between the productions $A, 1$ and $B, 1$ under lookahead b

LR(2) to LR(1)

Basic Idea:



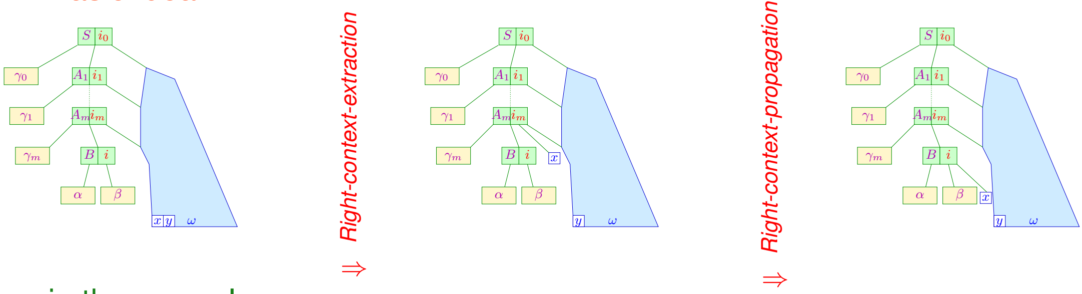
in the example:

Right-context is already extracted, so we only perform *Right-context-propagation*:

$$
\begin{array}{lcl}
S & \rightarrow & A\,b\,b^{0} \mid B\,b\,c^{1} \\
A & \rightarrow & a\,A^{0} \mid a^{1} \\
B & \rightarrow & a\,B^{0} \mid a^{1}
\end{array}
\quad \Rightarrow
$$

LR(2) to LR(1)

Basic Idea:



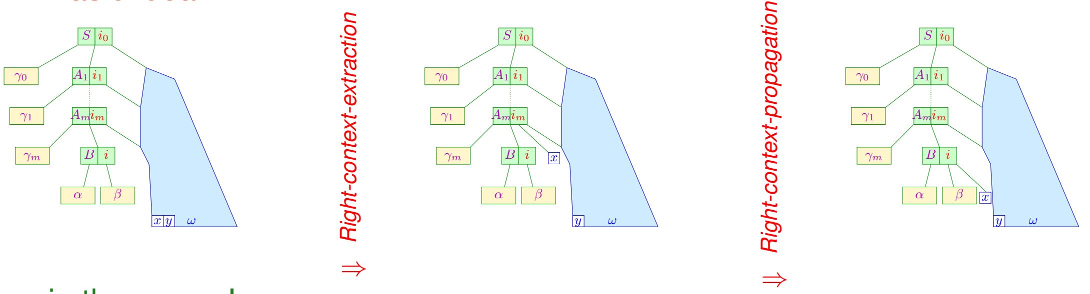
in the example:

Right-context is already extracted, so we only perform *Right-context-propagation*:

$$\begin{array}{lcl} S & \rightarrow & A\,b\,b^{0} \mid B\,b\,c^{1} \\ A & \rightarrow & a\,A^{0} \mid a^{1} \\ B & \rightarrow & a\,B^{0} \mid a^{1} \end{array} \quad \Rightarrow \quad S \rightarrow \langle A\,b\rangle\,b^{0} \mid \langle B\,b\rangle\,c^{1}$$

LR(2) to LR(1)

Basic Idea:



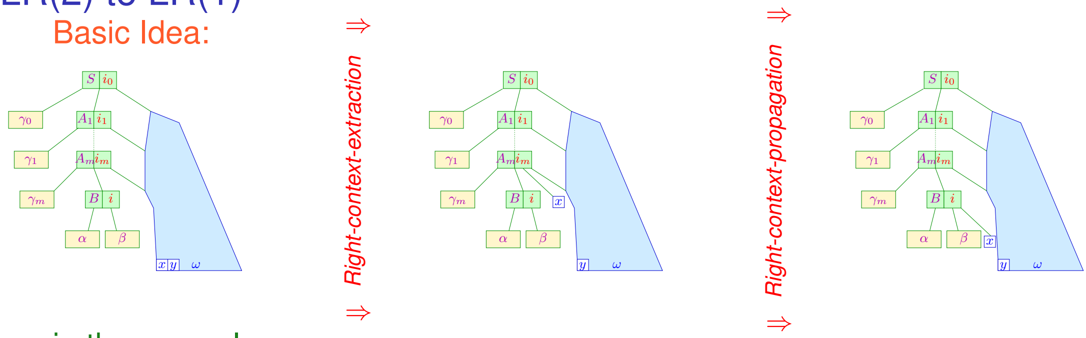
in the example:

Right-context is already extracted, so we only perform *Right-context-propagation*:

$$
\begin{array}{lll}
S & \to & A\,b\,b^0 \mid B\,b\,c^1 \\
A & \to & a\,A^0 \mid a^1 \\
B & \to & a\,B^0 \mid a^1
\end{array}
\quad \Rightarrow \quad
\begin{array}{lll}
S & \to & \langle A\,b\rangle\,b^0 \mid \langle B\,b\rangle\,c^1 \\
\langle A\,b\rangle & \to & a\,\langle A\,b\rangle^0 \mid a\,b^1
\end{array}
$$

LR(2) to LR(1)

Basic Idea:



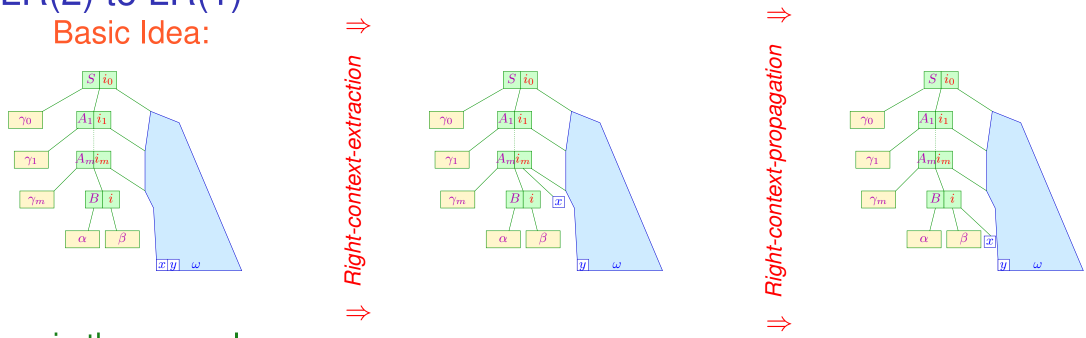
in the example:

Right-context is already extracted, so we only perform *Right-context-propagation*:

$$\begin{array}{lcl} S & \to & A\,b\,b^0 \mid B\,b\,c^1 \\ A & \to & a\,A^0 \mid a^1 \\ B & \to & a\,B^0 \mid a^1 \end{array} \quad \Rightarrow \quad \begin{array}{lcl} S & \to & \langle A\,b\rangle\,b^0 \mid \langle B\,b\rangle\,c^1 \\ \langle A\,b\rangle & \to & a\,\langle A\,b\rangle^0 \mid a\,b^1 \\ \langle B\,b\rangle & \to & a\,\langle B\,b\rangle^0 \mid a\,b^1 \end{array}$$

LR(2) to LR(1)

Basic Idea:



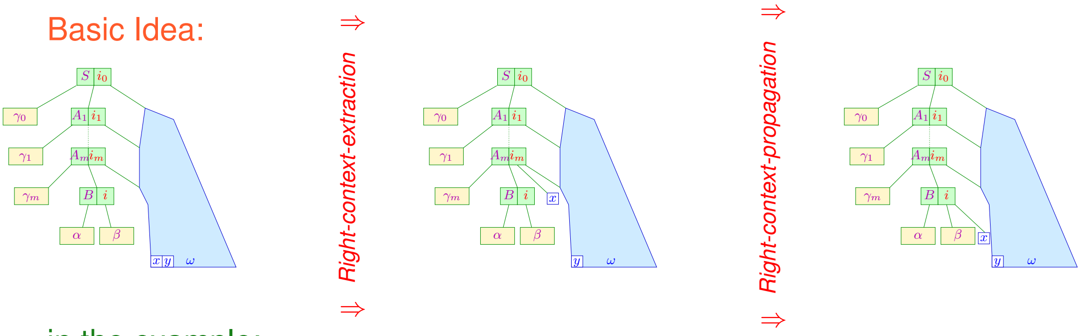
in the example:

Right-context is already extracted, so we only perform *Right-context-propagation*:

$$
\begin{array}{lll}
S & \to & A\, b\, b^0 \mid B\, b\, c^1 \\
A & \to & a\, A^0 \mid a^1 \\
B & \to & a\, B^0 \mid a^1
\end{array}
\quad \Rightarrow \quad
\begin{array}{lll}
S & \to & \langle A\, b \rangle\, b^0 \mid \langle B\, b \rangle\, c^1 \\
\langle A\, b \rangle & \to & a\, \langle A\, b \rangle^0 \mid a\, b^1 \\
\langle B\, b \rangle & \to & a\, \langle B\, b \rangle^0 \mid a\, b^1 \\
A & \to & a\, A^0 \mid a^1 \\
B & \to & a\, B^0 \mid a^1
\end{array}
$$

LR(2) to LR(1)

Basic Idea:



in the example:

Right-context is already extracted, so we only perform *Right-context-propagation*:

$$
\begin{array}{lll}
S & \to & A\, b\, b^{0} \mid B\, b\, c^{1} \\
A & \to & a\, A^{0} \mid a^{1} \\
B & \to & a\, B^{0} \mid a^{1}
\end{array}
\quad \Rightarrow \quad
\begin{array}{lll}
S & \to & \langle A\, b\rangle\, b^{0} \mid \langle B\, b\rangle\, c^{1} \\
\langle A\, b\rangle & \to & a\, \langle A\, b\rangle^{0} \mid a\, b^{1} \\
\langle B\, b\rangle & \to & a\, \langle B\, b\rangle^{0} \mid a\, b^{1}
\end{array}
$$

unreachable

LR(2) to LR(1)

Example cont'd:

$$\begin{array}{lll} S & \rightarrow & A'\, b^{0} \mid B'\, c^{1} \\ A' & \rightarrow & a\, A'^{0} \mid a\, b^{1} \\ B' & \rightarrow & a\, B'^{0} \mid a\, b^{1} \end{array}$$

LR(2) to LR(1)

Example cont'd:

$$\begin{array}{lcl} S & \rightarrow & A'\,b^{0} \mid B'\,c^{1} \\ A' & \rightarrow & a\,A'^{0} \mid a\,b^{1} \\ B' & \rightarrow & a\,B'^{0} \mid a\,b^{1} \end{array}$$

S rightmost-derives one of these forms:

$$a^n \underline{a\,b}b\;,\; a^n \underline{a\,b}c\;,\; a^n \underline{a\,A'}b\;,\; a^n \underline{a\,B'}c,\; \underline{A'}b\;,\; \underline{B'}c \quad \Rightarrow \quad LR(1)$$

LR(2) to LR(1)

Example 2:

$$
\begin{array}{lcl}
S & \rightarrow & b\,S\,S\ ^{0} \\
 & | & a\ ^{1} \\
 & | & a\,a\,c\ ^{2}
\end{array}
$$

LR(2) to LR(1)

Example 2:

$$\begin{array}{lcl} S & \rightarrow & b\,S\,S^{\,0} \\ & | & a^{\,1} \\ & | & a\,a\,c^{\,2} \end{array}$$

S rightmost-derives these forms among others:

$$\underline{b\,S\,S},\ b\,S\,\underline{a},\ b\,S\,\underline{a\,a\,c},\ b\,\underline{a}\,a,\ b\,\underline{a\,a\,c}\,a,\ b\,\underline{a}\,a\,a\,c,\ b\,\underline{a\,a\,c}\,a\,a\,c,\ \ldots\ \Rightarrow\ \text{min. } LR(2)$$

in $LR(1)$, you will have (at least) Shift-/Reduce-Conflicts between the items $[S{\rightarrow}a\,\bullet\,,a]$ and $[S{\rightarrow}a\bullet ac]$

$[\,S{\rightarrow}a\,]$'s right context is a nonterminal $\Rightarrow$ perform *Right-context-extraction*

$$\begin{array}{lcl} S & \rightarrow & b\,S\,S^{\,0} \\ & | & a^{\,1} \\ & | & a\,a\,c^{\,2} \end{array} \qquad \Rightarrow$$

LR(2) to LR(1)

Example 2:

$$
\begin{array}{lcl} S & \rightarrow & b\,S\,S^{\;0} \\ & | & a^{\;1} \\ & | & a\,a\,c^{\;2} \end{array}
$$

S rightmost-derives these forms among others:

$\underline{b\,S\,S}$, $b\,S\,\underline{a}$, $b\,S\,\underline{a\,a\,c}$, $b\,\underline{a}\,a$, $b\,\underline{a\,a\,c}\,a$, $b\,\underline{a}\,a\,a\,c$, $b\,\underline{a\,a\,c}\,a\,a\,c$, $\ldots$ $\Rightarrow$ min. $LR(2)$

in $LR(1)$, you will have (at least) Shift-/Reduce-Conflicts between the items $[S{\rightarrow}a\bullet, a]$ and $[S{\rightarrow}a\bullet ac]$

$[S{\rightarrow}a]$'s right context is a nonterminal $\Rightarrow$ perform *Right-context-extraction*

$$
\begin{array}{lcl} S & \rightarrow & b\,S\,S^{\;0} \\ & | & a^{\;1} \\ & | & a\,a\,c^{\;2} \end{array}
\quad\Rightarrow\quad
\begin{array}{lcl} S & \rightarrow & b\,S\,a\,\langle a/S\rangle^{\;0} \mid b\,S\,b\,\langle b/S\rangle^{\;0'} \\ & | & a^{\;1} \mid a\,a\,c^{\;2} \end{array}
$$

LR(2) to LR(1)

Example 2:

$$S \rightarrow b\,S\,S^{\,0} \mid a^{\,1} \mid a\,a\,c^{\,2}$$

S rightmost-derives these forms among others:

$$\underline{b\,S\,S},\ b\,S\,\underline{a},\ b\,S\,\underline{a\,a\,c},\ b\,\underline{a}\,a,\ b\,\underline{a\,a\,c}\,a,\ b\,\underline{a}\,a\,a\,c,\ b\,\underline{a\,a\,c}\,a\,a\,c,\ \ldots\ \Rightarrow \text{ min. } LR(2)$$

in $LR(1)$, you will have (at least) Shift-/Reduce-Conflicts between the items $[S \rightarrow a\bullet, a]$ and $[S \rightarrow a \bullet ac]$

$[\,S \rightarrow a\,]$'s right context is a nonterminal $\Rightarrow$ perform *Right-context-extraction*

$$S \rightarrow b\,S\,S^{\,0} \mid a^{\,1} \mid a\,a\,c^{\,2} \quad \Rightarrow \quad \begin{array}{lcl} S & \rightarrow & b\,S\,a\,\langle a/S\rangle^{\,0} \mid b\,S\,b\,\langle b/S\rangle^{\,0'} \\ & \mid & a^{\,1} \mid a\,a\,c^{\,2} \\ \langle a/S\rangle & \rightarrow & \epsilon^{0} \mid a\,c^{\,1} \end{array}$$

LR(2) to LR(1)

Example 2:

$$S \rightarrow b\,S\,S^{\,0} \mid a^{\,1} \mid a\,a\,c^{\,2}$$

S rightmost-derives these forms among others:

$\underline{b\,S\,S},\ b\,S\,\underline{a},\ b\,S\,\underline{a\,a\,c},\ b\,\underline{a}\,a,\ b\,\underline{a\,a\,c}\,a,\ b\,\underline{a}\,a\,a\,c,\ b\,\underline{a\,a\,c}\,a\,a\,c,\ \ldots \ \Rightarrow$ min. $LR(2)$

in $LR(1)$, you will have (at least) Shift-/Reduce-Conflicts between the items $[S{\rightarrow}a\bullet, a]$ and $[S{\rightarrow}a\bullet ac]$

$[\,S{\rightarrow}a\,]$'s right context is a nonterminal $\Rightarrow$ perform *Right-context-extraction*

$$S \rightarrow b\,S\,S^{\,0} \mid a^{\,1} \mid a\,a\,c^{\,2} \quad \Rightarrow \quad \begin{array}{lcl} S & \rightarrow & b\,S\,a\,\langle a/S\rangle^{\,0} \mid b\,S\,b\,\langle b/S\rangle^{\,0'} \\ & \mid & a^{\,1} \mid a\,a\,c^{\,2} \\ \langle a/S\rangle & \rightarrow & \epsilon^{0} \mid a\,c^{\,1} \\ \langle b/S\rangle & \rightarrow & S\,S^{0} \end{array}$$

LR(2) to LR(1)

Example 2:

$$\begin{array}{lcl} S & \rightarrow & b\,S\,S^{\,0} \\ & | & a^{\,1} \\ & | & a\,a\,c^{\,2} \end{array}$$

S rightmost-derives these forms among others:

$$\underline{b\,S\,S},\ b\,S\,\underline{a},\ b\,S\,\underline{a\,a\,c},\ b\,\underline{a}\,a,\ b\,\underline{a\,a\,c}\,a,\ b\,\underline{a}\,a\,a\,c,\ b\,\underline{a\,a\,c}\,a\,a\,c,\ \ldots\ \Rightarrow\ \text{min. } LR(2)$$

in $LR(1)$, you will have (at least) Shift-/Reduce-Conflicts between the items $[S{\rightarrow}a\bullet, a]$ and $[S{\rightarrow}a\bullet ac]$

$[\,S{\rightarrow}a\,]$'s right context is a nonterminal $\Rightarrow$ perform *Right-context-extraction*

$$\begin{array}{lcl} S & \rightarrow & b\,S\,S^{\,0} \\ & | & a^{\,1} \\ & | & a\,a\,c^{\,2} \end{array} \quad \Rightarrow \quad \begin{array}{lcl} S & \rightarrow & b\,S\,a\,\langle a/S\rangle^{\,0} \mid b\,S\,b\,\langle b/S\rangle^{\,0'} \\ & | & a^{\,1} \mid a\,a\,c^{\,2} \\ \langle a/S\rangle & \rightarrow & \epsilon^{0} \mid a\,c^{\,1} \\ \langle b/S\rangle & \rightarrow & S\,a\,\langle a/S\rangle^{0} \mid S\,b\,\langle b/S\rangle^{0'} \end{array}$$

LR(2) to LR(1)

Example 2 cont'd:

$[\, S \rightarrow a \,]$'s right context is now terminal a $\Rightarrow$ perform *Right-context-propagation*

LR(2) to LR(1)

Example 2 cont'd:

$[\, S \rightarrow a \,]$'s right context is now terminal a $\Rightarrow$ perform *Right-context-propagation*

$$
\begin{array}{lcl}
S & \rightarrow & b\, S\, a\, \langle a/S \rangle^{\,0} \\
 & | & b\, S\, b\, \langle b/S \rangle^{\,0'} \\
 & | & a^{\,1} \mid a\, a\, c^{\,2} \\
\langle a/S \rangle & \rightarrow & \epsilon^{0} \mid a\, c^{\,1} \\
\langle b/S \rangle & \rightarrow & S\, a\, \langle a/S \rangle^{0} \mid S\, b\, \langle b/S \rangle^{0'}
\end{array}
\quad \Rightarrow
$$

LR(2) to LR(1)

Example 2 cont'd:

$[\,S \rightarrow a\,]$'s right context is now terminal a $\Rightarrow$ perform *Right-context-propagation*

$$
\begin{array}{lcl}
S & \rightarrow & b\,S\,a\,\langle a/S\rangle^{\,0} \\
 & | & b\,S\,b\,\langle b/S\rangle^{\,0'} \\
 & | & a^{\,1} \mid a\,a\,c^{\,2} \\
\langle a/S\rangle & \rightarrow & \epsilon^{0} \mid a\,c^{\,1} \\
\langle b/S\rangle & \rightarrow & S\,a\,\langle a/S\rangle^{0} \mid S\,b\,\langle b/S\rangle^{0'}
\end{array}
\quad \Rightarrow \quad
\begin{array}{lcl}
S & \rightarrow & b\,\langle Sa\rangle\,\langle a/S\rangle^{\,0} \\
 & | & b\,S\,b\,\langle b/S\rangle^{\,0'} \\
 & | & a^{\,1} \mid a\,a\,c^{\,2} \\
\langle a/S\rangle & \rightarrow & \epsilon^{0} \mid a\,c^{\,1} \\
\langle b/S\rangle & \rightarrow & \langle Sa\rangle\,\langle a/S\rangle^{0} \mid S\,b\,\langle b/S\rangle^{0'}
\end{array}
$$

LR(2) to LR(1)

Example 2 cont'd:

$[\,S \to a\,]$'s right context is now terminal a $\Rightarrow$ perform *Right-context-propagation*

$$
\begin{array}{lcl}
S & \to & b\,S\,a\,\langle a/S\rangle^{\,0} \\
 & | & b\,S\,b\,\langle b/S\rangle^{\,0'} \\
 & | & a^{\,1} \mid a\,a\,c^{\,2} \\
\langle a/S\rangle & \to & \epsilon^{0} \mid a\,c^{\,1} \\
\langle b/S\rangle & \to & S\,a\,\langle a/S\rangle^{0} \mid S\,b\,\langle b/S\rangle^{0'}
\end{array}
\quad \Rightarrow \quad
\begin{array}{lcl}
S & \to & b\,\langle Sa\rangle\,\langle a/S\rangle^{\,0} \\
 & | & b\,S\,b\,\langle b/S\rangle^{\,0'} \\
 & | & a^{\,1} \mid a\,a\,c^{\,2} \\
\langle a/S\rangle & \to & \epsilon^{0} \mid a\,c^{\,1} \\
\langle b/S\rangle & \to & \langle Sa\rangle\,\langle a/S\rangle^{0} \mid S\,b\,\langle b/S\rangle^{0'} \\
\langle Sa\rangle & \to & b\,\langle Sa\rangle\,\langle a/S\rangle\,a^{0} \\
 & | & b\,S\,b\,\langle b/S\rangle\,a^{\,0'} \\
 & | & a\,a^{\,1} \mid a\,a\,c\,a^{\,2}
\end{array}
$$

LR(2) to LR(1)

Example 2 cont'd:

$[\,S \rightarrow a\,]$'s right context is now terminal a $\Rightarrow$ perform *Right-context-propagation*

$$
\begin{array}{lcl}
S & \rightarrow & b\,S\,a\,\langle a/S\rangle^{\,0} \\
 & | & b\,S\,b\,\langle b/S\rangle^{\,0'} \\
 & | & a^{\,1} \mid a\,a\,c^{\,2} \\
\langle a/S\rangle & \rightarrow & \epsilon^{0} \mid a\,c^{\,1} \\
\langle b/S\rangle & \rightarrow & S\,a\,\langle a/S\rangle^{0} \mid S\,b\,\langle b/S\rangle^{0'}
\end{array}
\quad \Rightarrow \quad
\begin{array}{lcl}
S & \rightarrow & b\,\langle Sa\rangle\,\langle a/S\rangle^{\,0} \\
 & | & b\,S\,b\,\langle b/S\rangle^{\,0'} \\
 & | & a^{\,1} \mid a\,a\,c^{\,2} \\
\langle a/S\rangle & \rightarrow & \epsilon^{0} \mid a\,c^{\,1} \\
\langle b/S\rangle & \rightarrow & \langle Sa\rangle\,\langle a/S\rangle^{0} \mid S\,b\,\langle b/S\rangle^{0'} \\
\langle Sa\rangle & \rightarrow & b\,\langle Sa\rangle\,\langle\langle a/S\rangle\,a\rangle^{0} \\
 & | & b\,S\,b\,\langle b/S\rangle\,a^{\,0'} \\
 & | & a\,a^{\,1} \mid a\,a\,c\,a^{\,2} \\
\langle\langle a/S\rangle a\rangle & \rightarrow & a^{0} \mid a\,c\,a^{\,1}
\end{array}
$$

LR(2) to LR(1)

Example 2 cont'd:

$[\,S \rightarrow a\,]$'s right context is now terminal a $\Rightarrow$ perform *Right-context-propagation*

$$
\begin{array}{lcl}
S & \rightarrow & b\,S\,a\,\langle a/S\rangle^{\,0} \\
 & | & b\,S\,b\,\langle b/S\rangle^{\,0'} \\
 & | & a^{\,1} \mid a\,a\,c^{\,2} \\
\langle a/S\rangle & \rightarrow & \epsilon^{0} \mid a\,c^{\,1} \\
\langle b/S\rangle & \rightarrow & S\,a\,\langle a/S\rangle^{0} \mid S\,b\,\langle b/S\rangle^{0'}
\end{array}
\quad\Rightarrow\quad
\begin{array}{lcl}
S & \rightarrow & b\,\langle Sa\rangle\,\langle a/S\rangle^{\,0} \\
 & | & b\,S\,b\,\langle b/S\rangle^{\,0'} \\
 & | & a^{\,1} \mid a\,a\,c^{\,2} \\
\langle a/S\rangle & \rightarrow & \epsilon^{0} \mid a\,c^{\,1} \\
\langle b/S\rangle & \rightarrow & \langle Sa\rangle\,\langle a/S\rangle^{0} \mid S\,b\,\langle b/S\rangle^{0'} \\
\langle Sa\rangle & \rightarrow & b\,\langle Sa\rangle\,\langle\langle a/S\rangle\,a\rangle^{0} \\
 & | & b\,S\,b\,\langle\langle b/S\rangle\,a\rangle^{\,0'} \\
 & | & a\,a^{\,1} \mid a\,a\,c\,a^{\,2} \\
\langle\langle a/S\rangle a\rangle & \rightarrow & a^{0} \mid a\,c\,a^{\,1} \\
\langle\langle b/S\rangle a\rangle & \rightarrow & \langle Sa\rangle\langle a/S\rangle\,a^{0} \mid S\,b\,\langle b/S\rangle\,a^{0'}
\end{array}
$$

LR(2) to LR(1)

Example 2 cont'd:

$[\, S \to a \,]$'s right context is now terminal a $\Rightarrow$ perform *Right-context-propagation*

$$
\begin{array}{lcl}
S & \to & b\, S\, a\, \langle a/S \rangle^{\,0} \\
 & | & b\, S\, b\, \langle b/S \rangle^{\,0'} \\
 & | & a^{\,1} \mid a\, a\, c^{\,2} \\
\langle a/S \rangle & \to & \epsilon^{0} \mid a\, c^{\,1} \\
\langle b/S \rangle & \to & S\, a\, \langle a/S \rangle^{0} \mid S\, b\, \langle b/S \rangle^{0'}
\end{array}
\quad \Rightarrow \quad
\begin{array}{lcl}
S & \to & b\, \langle Sa \rangle\, \langle a/S \rangle^{\,0} \\
 & | & b\, S\, b\, \langle b/S \rangle^{\,0'} \\
 & | & a^{\,1} \mid a\, a\, c^{\,2} \\
\langle a/S \rangle & \to & \epsilon^{0} \mid a\, c^{\,1} \\
\langle b/S \rangle & \to & \langle Sa \rangle\, \langle a/S \rangle^{0} \mid S\, b\, \langle b/S \rangle^{0'} \\
\langle Sa \rangle & \to & b\, \langle Sa \rangle\, \langle\langle a/S \rangle\, a \rangle^{0} \\
 & | & b\, S\, b\, \langle\langle b/S \rangle\, a \rangle^{\,0'} \\
 & | & a\, a^{\,1} \mid a\, a\, c\, a^{\,2} \\
\langle\langle a/S \rangle a \rangle & \to & a^{0} \mid a\, c\, a^{\,1} \\
\langle\langle b/S \rangle a \rangle & \to & \langle Sa \rangle \langle\langle a/S \rangle a \rangle^{0} \mid S\, b\, \langle\langle b/S \rangle a \rangle^{0'}
\end{array}
$$

LR(2) to LR(1)

Example 2 finished:

With fresh nonterminals we get the final grammar

$$S \rightarrow b\,S\,S\,^{0} \mid a\,^{1} \mid a\,a\,c\,^{2} \quad \Rightarrow$$

$$\begin{array}{lcl}
S & \rightarrow & b\,C\,A,^{0} \mid b\,S\,b\,B,^{1} \mid a\,^{2} \mid a\,a\,c\,^{3} \\
A & \rightarrow & \epsilon^{0} \mid a\,c\,^{1} \\
B & \rightarrow & C\,A^{0} \mid S\,b\,B^{1} \\
C & \rightarrow & b\,C\,D^{0} \mid b\,S\,b\,E\,^{1} \mid a\,a\,^{2} \mid a\,a\,c\,a\,^{3} \\
D & \rightarrow & a^{0} \mid a\,c\,a\,^{1} \\
E & \rightarrow & C\,D^{0} \mid S\,b\,E^{1}
\end{array}$$

Chapter 2:

Summary

Special LR(k)-Subclasses

Discussion:

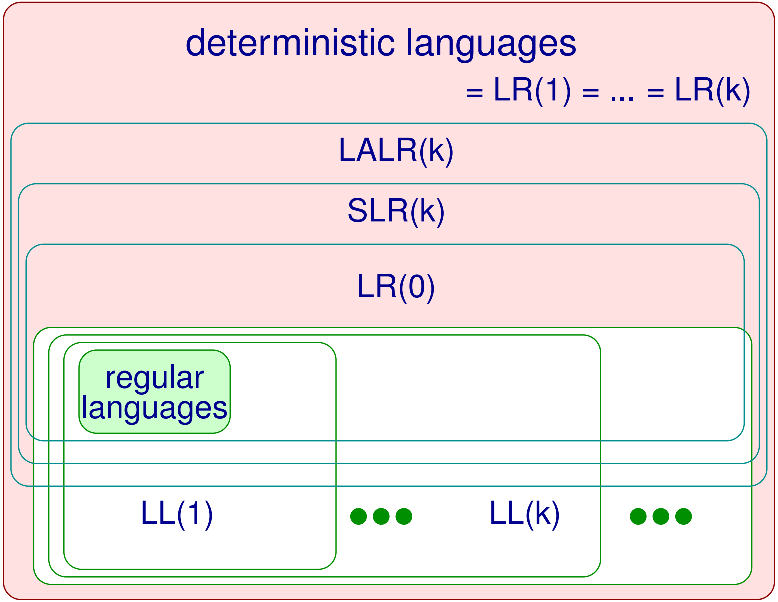
- Our examples mostly were $LR(1)$ – or could be transformed to $LR(1)$
- In general, the canonical $LR(k)$-automaton has much more states then $LR(G) = LR(G, 0)$
- Therefore in practice, subclasses of $LR(k)$-grammars are often considered, which only use $LR(G)$ ...

Special LR(k)-Subclasses

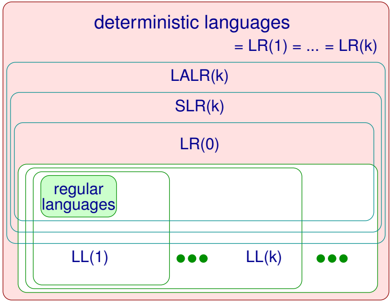
Discussion:

- Our examples mostly were $LR(1)$ – or could be transformed to $LR(1)$
- 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 $\Longrightarrow$ $LALR(k)$

Parsing Methods



Parsing Methods



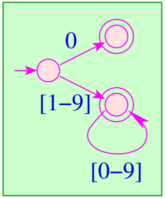
Discussion:

- 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.

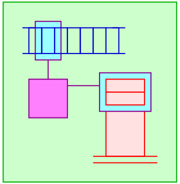
Lexical and Syntactical Analysis:

Concept of specification and implementation:

0 | [1-9][0-9]* → **Generator** →

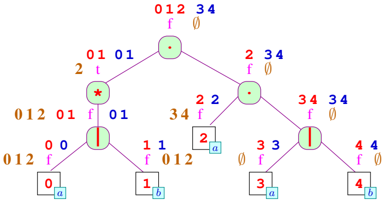


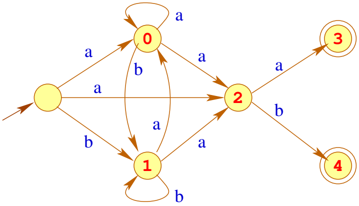
E→E{op}E → **Generator** →



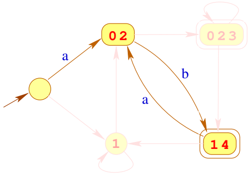
Lexical and Syntactical Analysis:

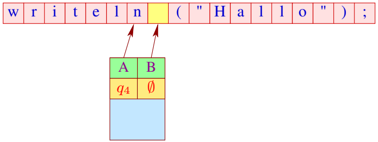
From Regular Expressions to Finite Automata





From Finite Automata to Scanners

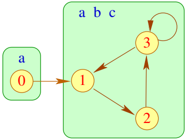


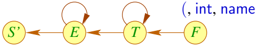


Lexical and Syntactical Analysis:

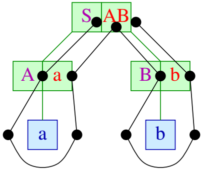
Computation of lookahead sets:

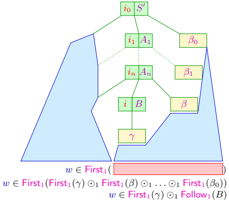
$$\begin{array}{lll@{\qquad}lll} F_\epsilon(S') & \supseteq & F_\epsilon(E) & F_\epsilon(E) & \supseteq & F_\epsilon(E) \\ F_\epsilon(E) & \supseteq & F_\epsilon(T) & F_\epsilon(T) & \supseteq & F_\epsilon(T) \\ F_\epsilon(T) & \supseteq & F_\epsilon(F) & F_\epsilon(F) & \supseteq & \{\,(\,,\text{name},\text{int}\} \end{array}$$

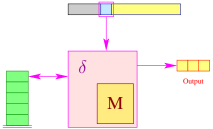




From Item-Pushdown Automata to LL(1)-Parsers:

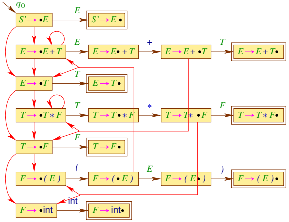


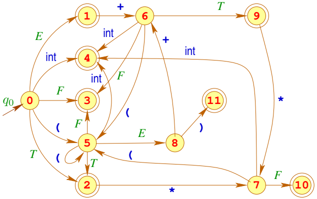




Lexical and Syntactical Analysis:

From characteristic to canonical Automata:





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