Compilers

Nikos Papaspyrou Kostis Sagonas

nickie@softlab.ntua.gr kostis@cs.ntua.gr

National Technical University of Athens

School of Electrical and Computer Engineering

Software Engineering Laboratory

Polytechnioupoli, 15780 Zografou, Athens, Greece.

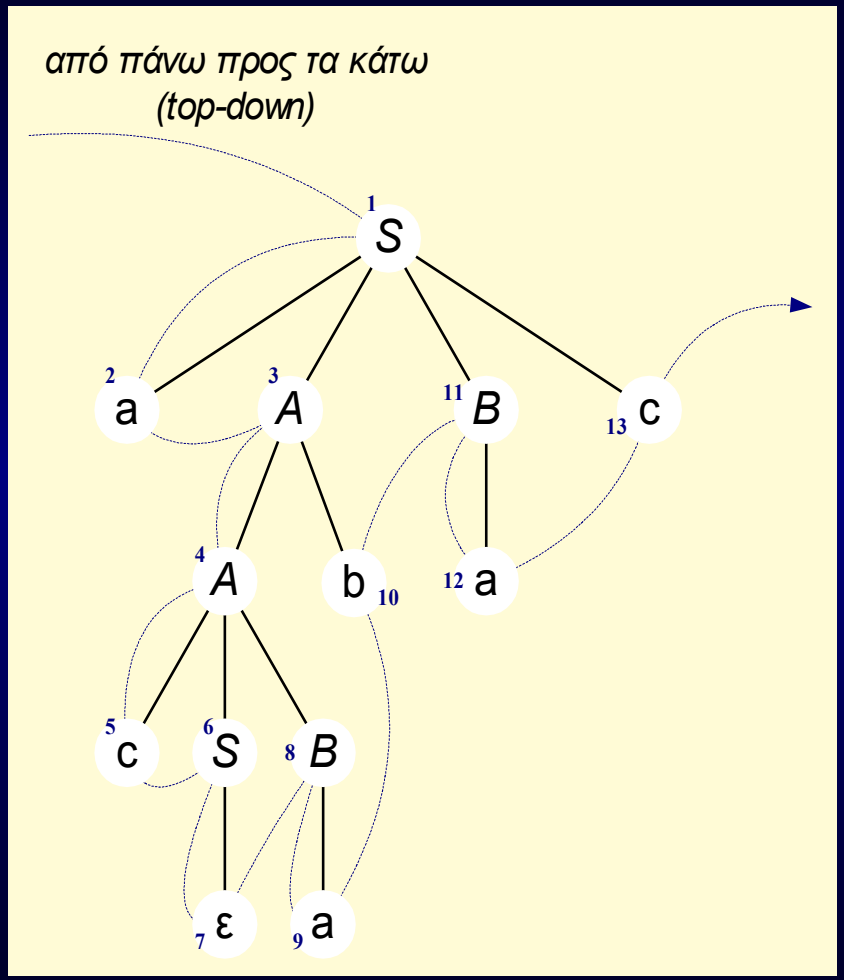
March 2017

Chapter 4:
Parsing

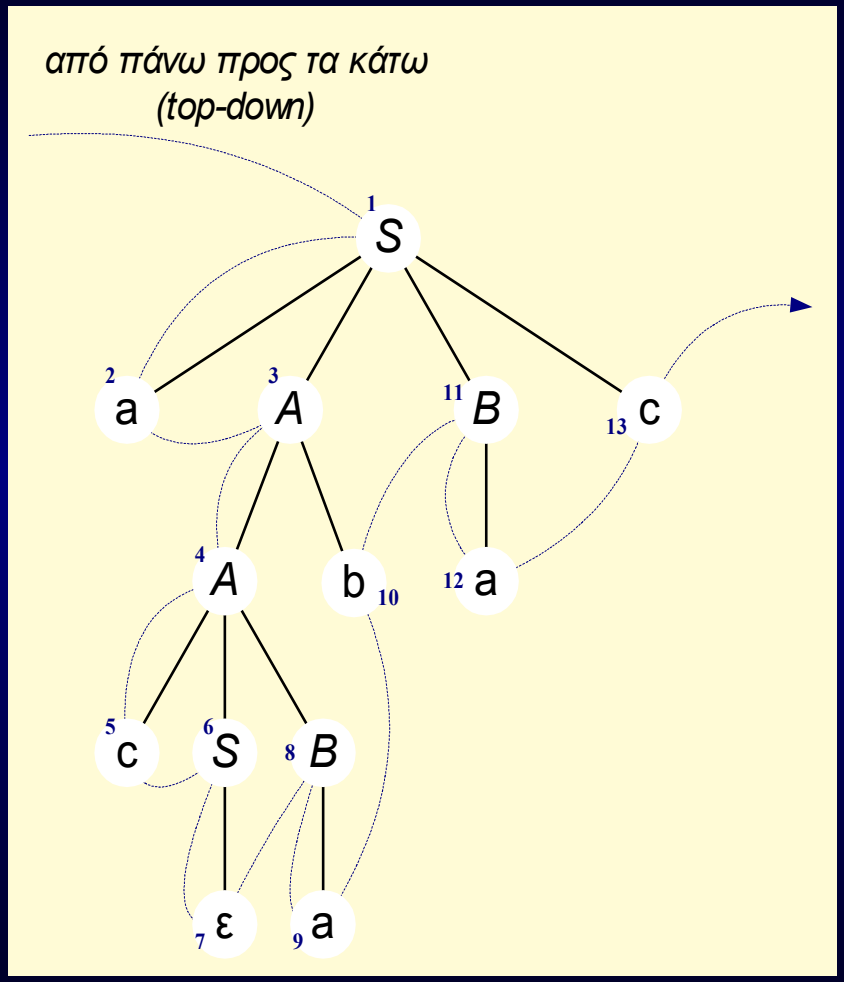
Parsing

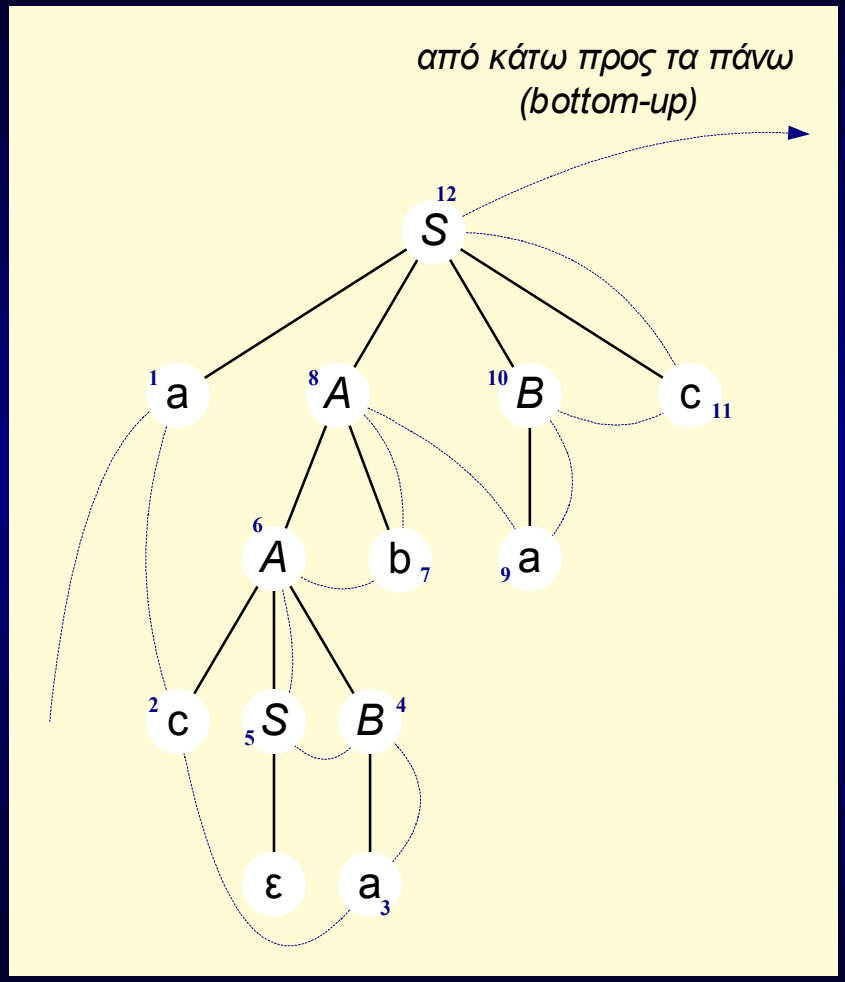
- Parse tree
- It can be built in two ways:
  - Top-down
    i.e., starting from the root and moving towards the leaves
  - Bottom-up
    i.e., starting from the leaves and moving towards the root

Top-down and bottom-up



Top-down and bottom-up





Auxiliaries (i)

- FIRST sets
  - Let $\alpha \in (T \cup N)^*$ be a string
  - The set FIRST$(\alpha) \subseteq T \cup \{ \epsilon \}$ contains the terminal symbols from which all strings that are produced by α are bound to start

Auxiliaries (i)

- FIRST sets
  - Let $\alpha \in (T \cup N)^*$ be a string
  - The set FIRST$(\alpha) \subseteq T \cup \{ \epsilon \}$ contains the terminal symbols from which all strings that are produced by α are bound to start

  - If $\alpha \Rightarrow^* a\beta$ then $a \in$ FIRST(α)
  - If $\alpha \Rightarrow^* \epsilon$ then $\epsilon \in$ FIRST(α)

Auxiliaries (ii)

- FOLLOW sets
  - Let A be a non-terminal symbol
  - The set FOLLOW$(A) \subseteq T \cup \{$ `EOF` $\}$ contains all terminal symbol that may follow A during a production from the start symbol S
  - If A can be the last symbol in a production, then `EOF` $\in$ FOLLOW(A)

Auxiliaries (ii)

- FOLLOW sets
  - Let A be a non-terminal symbol
  - The set FOLLOW$(A) \subseteq T \cup \{$ `EOF` $\}$ contains all terminal symbol that may follow A during a production from the start symbol S
  - If A can be the last symbol in a production, then `EOF` $\in$ FOLLOW(A)

  - If $S \Rightarrow^* \alpha A \mathrm{a} \beta$ then a $\in$ FOLLOW(A)
  - If $S \Rightarrow^* \alpha A$ then `EOF` $\in$ FOLLOW(A)

Calculating FIRST *(i)*

- $\text{FIRST}(\epsilon) = \{ \epsilon \}$
- $\text{FIRST}(\text{a}\beta) = \{ \text{a} \}$
- if $\epsilon \notin \text{FIRST}(A)$
  then $\text{FIRST}(A\beta) = \text{FIRST}(A)$
- if $\epsilon \in \text{FIRST}(A)$
  then $\text{FIRST}(A\beta) = (\text{FIRST}(A) - \{ \epsilon \}) \cup \text{FIRST}(\beta)$
- for each rule $A \rightarrow \alpha$, it must be $\text{FIRST}(\alpha) \subseteq \text{FIRST}(A)$

Calculating FIRST *(ii)*

- Example

  $\text{FIRST}(E) =$
  $\text{FIRST}(T) =$
  $\text{FIRST}(F) =$
  $\text{FIRST}(E') =$
  $\text{FIRST}(T') =$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FIRST *(ii)*

- Example

$$
\begin{aligned}
\text{FIRST}(E) &= \\
\text{FIRST}(T) &= \\
\text{FIRST}(F) &= \\
\text{FIRST}(E') &= \{+, \epsilon\} \\
\text{FIRST}(T') &= \\
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FIRST *(ii)*

- Example

$$
\begin{aligned}
\text{FIRST}(E) &= \\
\text{FIRST}(T) &= \\
\text{FIRST}(F) &= \\
\text{FIRST}(E') &= \{+, \epsilon\} \\
\text{FIRST}(T') &= \{*, \epsilon\}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FIRST *(ii)*

- Example

$$
\begin{aligned}
\text{FIRST}(E) &= \\
\text{FIRST}(T) &= \\
\text{FIRST}(F) &= \{\,\text{id}, (\,\} \\
\text{FIRST}(E') &= \{\,+, \epsilon\,\} \\
\text{FIRST}(T') &= \{\,*, \epsilon\,\}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FIRST *(ii)*

- Example

$$
\begin{aligned}
\text{FIRST}(E) &= \\
\text{FIRST}(T) &= \{\text{id}, (\,\} \\
\text{FIRST}(F) &= \{\text{id}, (\,\} \\
\text{FIRST}(E') &= \{+, \epsilon\} \\
\text{FIRST}(T') &= \{*, \epsilon\}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FIRST *(ii)*

- Example

$$
\begin{aligned}
\text{FIRST}(E) &= \{\text{id}, (\,\} \\
\text{FIRST}(T) &= \{\text{id}, (\,\} \\
\text{FIRST}(F) &= \{\text{id}, (\,\} \\
\text{FIRST}(E') &= \{+, \epsilon\} \\
\text{FIRST}(T') &= \{*, \epsilon\}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(i)*

- `EOF` $\in$ FOLLOW(S)
- for each rule $A \rightarrow \alpha B \beta$
  - (FIRST(β) $- \{ \epsilon \}$) $\subseteq$ FOLLOW(B)
  - if $\epsilon \in$ FIRST(β)
    then FOLLOW(A) $\subseteq$ FOLLOW(B)

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{ \\
\text{FOLLOW}(T) &= \{ \\
\text{FOLLOW}(F) &= \{ \\
\text{FOLLOW}(E') &= \{ \\
\text{FOLLOW}(T') &= \{
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{ +, \epsilon \} \\
\text{FIRST}(T') &= \{ *, \epsilon \}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{ \texttt{EOF} \\
\text{FOLLOW}(T) &= \{ \\
\text{FOLLOW}(F) &= \{ \\
\text{FOLLOW}(E') &= \{ \\
\text{FOLLOW}(T') &= \{
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{ +, \epsilon \} \\
\text{FIRST}(T') &= \{ *, \epsilon \}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{\texttt{EOF}, \texttt{)}\\
\text{FOLLOW}(T) &= \{\\
\text{FOLLOW}(F) &= \{\\
\text{FOLLOW}(E') &= \{\\
\text{FOLLOW}(T') &= \{
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{+, \epsilon\}\\
\text{FIRST}(T') &= \{*, \epsilon\}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E'\\
E' &\rightarrow \epsilon\\
E' &\rightarrow +\ T\ E'\\
T &\rightarrow F\ T'\\
T' &\rightarrow \epsilon\\
T' &\rightarrow *\ F\ T'\\
F &\rightarrow (\ E\ )\\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$\begin{aligned}
\text{FOLLOW}(E) &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T) &= \{ \\
\text{FOLLOW}(F) &= \{ \\
\text{FOLLOW}(E') &= \{ \\
\text{FOLLOW}(T') &= \{
\end{aligned}$$

$$\begin{aligned}
\text{FIRST}(E') &= \{+, \epsilon\} \\
\text{FIRST}(T') &= \{*, \epsilon\}
\end{aligned}$$

$$\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}$$

Calculating FOLLOW *(ii)*

- Example

$$\begin{aligned}
\text{FOLLOW}(E) &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T) &= \{+ \\
\text{FOLLOW}(F) &= \{ \\
\text{FOLLOW}(E') &= \{ \\
\text{FOLLOW}(T') &= \{
\end{aligned}$$

$$\begin{aligned}
\text{FIRST}(E') &= \{+, \epsilon\} \\
\text{FIRST}(T') &= \{*, \epsilon\}
\end{aligned}$$

$$\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{ \texttt{EOF}, \texttt{)} \} \\
\text{FOLLOW}(T) &= \{ \texttt{+}, \texttt{EOF}, \texttt{)} \\
\text{FOLLOW}(F) &= \{ \\
\text{FOLLOW}(E') &= \{ \\
\text{FOLLOW}(T') &= \{
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{ \texttt{+}, \epsilon \} \\
\text{FIRST}(T') &= \{ \texttt{*}, \epsilon \}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow +\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow *\ F\ T' \\
F &\rightarrow (\ E\ ) \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{ \texttt{EOF}, \texttt{)} \} \\
\text{FOLLOW}(T) &= \{ \texttt{+}, \texttt{EOF}, \texttt{)} \\
\text{FOLLOW}(F) &= \{ \texttt{*} \\
\text{FOLLOW}(E') &= \{ \\
\text{FOLLOW}(T') &= \{
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{ \texttt{+}, \epsilon \} \\
\text{FIRST}(T') &= \{ \texttt{*}, \epsilon \}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow \texttt{+}\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow \texttt{*}\ F\ T' \\
F &\rightarrow \texttt{(}\ E\ \texttt{)} \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{ \texttt{EOF}, \texttt{)} \} \\
\text{FOLLOW}(T) &= \{ \texttt{+}, \texttt{EOF}, \texttt{)} \\
\text{FOLLOW}(F) &= \{ \texttt{*}, \texttt{+}, \texttt{EOF}, \texttt{)} \\
\text{FOLLOW}(E') &= \{ \\
\text{FOLLOW}(T') &= \{
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{ \texttt{+}, \epsilon \} \\
\text{FIRST}(T') &= \{ \texttt{*}, \epsilon \}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow \texttt{+}\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow \texttt{*}\ F\ T' \\
F &\rightarrow \texttt{(}\ E\ \texttt{)} \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{ \texttt{EOF}, \texttt{)} \} \\
\text{FOLLOW}(T) &= \{ \texttt{+}, \texttt{EOF}, \texttt{)} \\
\text{FOLLOW}(F) &= \{ \texttt{*}, \texttt{+}, \texttt{EOF}, \texttt{)} \\
\text{FOLLOW}(E') &= \{ \texttt{EOF}, \texttt{)} \\
\text{FOLLOW}(T') &= \{ \\
\\
\text{FIRST}(E') &= \{ \texttt{+}, \epsilon \} \\
\text{FIRST}(T') &= \{ \texttt{*}, \epsilon \}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow \texttt{+}\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow \texttt{*}\ F\ T' \\
F &\rightarrow \texttt{(}\ E\ \texttt{)} \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{ \texttt{EOF}, \texttt{)} \} \\
\text{FOLLOW}(T) &= \{ \texttt{+}, \texttt{EOF}, \texttt{)} \\
\text{FOLLOW}(F) &= \{ \texttt{*}, \texttt{+}, \texttt{EOF}, \texttt{)} \\
\text{FOLLOW}(E') &= \{ \texttt{EOF}, \texttt{)} \} \\
\text{FOLLOW}(T') &= \{
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{ \texttt{+}, \epsilon \} \\
\text{FIRST}(T') &= \{ \texttt{*}, \epsilon \}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow \texttt{+}\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow \texttt{*}\ F\ T' \\
F &\rightarrow \texttt{(}\ E\ \texttt{)} \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T) &= \{\texttt{+}, \texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(F) &= \{\texttt{*}, \texttt{+}, \texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(E') &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T') &= \{\texttt{+}, \texttt{EOF}, \texttt{)}\}
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{\texttt{+}, \epsilon\} \\
\text{FIRST}(T') &= \{\texttt{*}, \epsilon\}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow \texttt{+}\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow \texttt{*}\ F\ T' \\
F &\rightarrow \texttt{(}\ E\ \texttt{)} \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T) &= \{\texttt{+}, \texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(F) &= \{\texttt{*}, \texttt{+}, \texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(E') &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T') &= \{\texttt{+}, \texttt{EOF}, \texttt{)}\}
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{\texttt{+}, \epsilon\} \\
\text{FIRST}(T') &= \{\texttt{*}, \epsilon\}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow \texttt{+}\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow \texttt{*}\ F\ T' \\
F &\rightarrow \texttt{(}\ E\ \texttt{)} \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T) &= \{\texttt{+}, \texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(F) &= \{\texttt{*}, \texttt{+}, \texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(E') &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T') &= \{\texttt{+}, \texttt{EOF}, \texttt{)}\}
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{\texttt{+}, \epsilon\} \\
\text{FIRST}(T') &= \{\texttt{*}, \epsilon\}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow \texttt{+}\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow \texttt{*}\ F\ T' \\
F &\rightarrow \texttt{(}\ E\ \texttt{)} \\
F &\rightarrow \text{id}
\end{aligned}
$$

Calculating FOLLOW *(ii)*

- Example

$$
\begin{aligned}
\text{FOLLOW}(E) &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T) &= \{\texttt{+}, \texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(F) &= \{\texttt{*}, \texttt{+}, \texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(E') &= \{\texttt{EOF}, \texttt{)}\} \\
\text{FOLLOW}(T') &= \{\texttt{+}, \texttt{EOF}, \texttt{)}\}
\end{aligned}
$$

$$
\begin{aligned}
\text{FIRST}(E') &= \{\texttt{+}, \epsilon\} \\
\text{FIRST}(T') &= \{\texttt{*}, \epsilon\}
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow \epsilon \\
E' &\rightarrow \texttt{+}\ T\ E' \\
T &\rightarrow F\ T' \\
T' &\rightarrow \epsilon \\
T' &\rightarrow \texttt{*}\ F\ T' \\
F &\rightarrow \texttt{(}\ E\ \texttt{)} \\
F &\rightarrow \text{id}
\end{aligned}
$$

Top-down parsing (i)

- Parsing starts from the root
- Every time, we seek:
  - the non-terminal symbol to replace
    - $\Rightarrow$ usually we pick the left-most
  - the production rule to apply
    - $\Rightarrow$ based on the next k tokens of the input string: LL(k)
- We repeat until the non-terminal symbols have been exhausted

LL(1) grammars

- Necessary conditions:
  - No left recursion (direct or indirect)
  - No common prefix in alternative rules
- Sometimes it is possible to transform a grammar into an equivalent LL(1)
  - $\Rightarrow$ left recursion elimination
  - $\Rightarrow$ left factorization

Transformation to LL(1)

- Substitution

$$\begin{array}{l} A \rightarrow \alpha_1 \mid \ldots \mid \alpha_n \\ B \rightarrow \beta_1 A \beta_2 \end{array} \Rightarrow \begin{array}{l} A \rightarrow \alpha_1 \mid \ldots \mid \alpha_n \\ B \rightarrow \beta_1 \alpha_1 \beta_2 \mid \ldots \mid \beta_1 \alpha_n \beta_2 \end{array}$$

- Left factorization

$$A \rightarrow \alpha\beta_1 \mid \ldots \mid \alpha\beta_n \Rightarrow \begin{array}{l} A \rightarrow \alpha B \\ B \rightarrow \beta_1 \mid \ldots \mid \beta_n \end{array}$$

- Direct left recursion elimination

$$A \rightarrow A\alpha_1 \mid \ldots \mid A\alpha_n \mid \beta_1 \mid \ldots \mid \beta_m$$

$$\Rightarrow \begin{array}{l} A \rightarrow \beta_1 B \mid \ldots \mid \beta_m B \\ B \rightarrow \alpha_1 B \mid \ldots \mid \alpha_n B \mid \epsilon \end{array}$$

Recursive descent parsing

$A \rightarrow \alpha_1 \mid \ldots \mid \alpha_n$

is transformed into a procedure whose body is of the form:

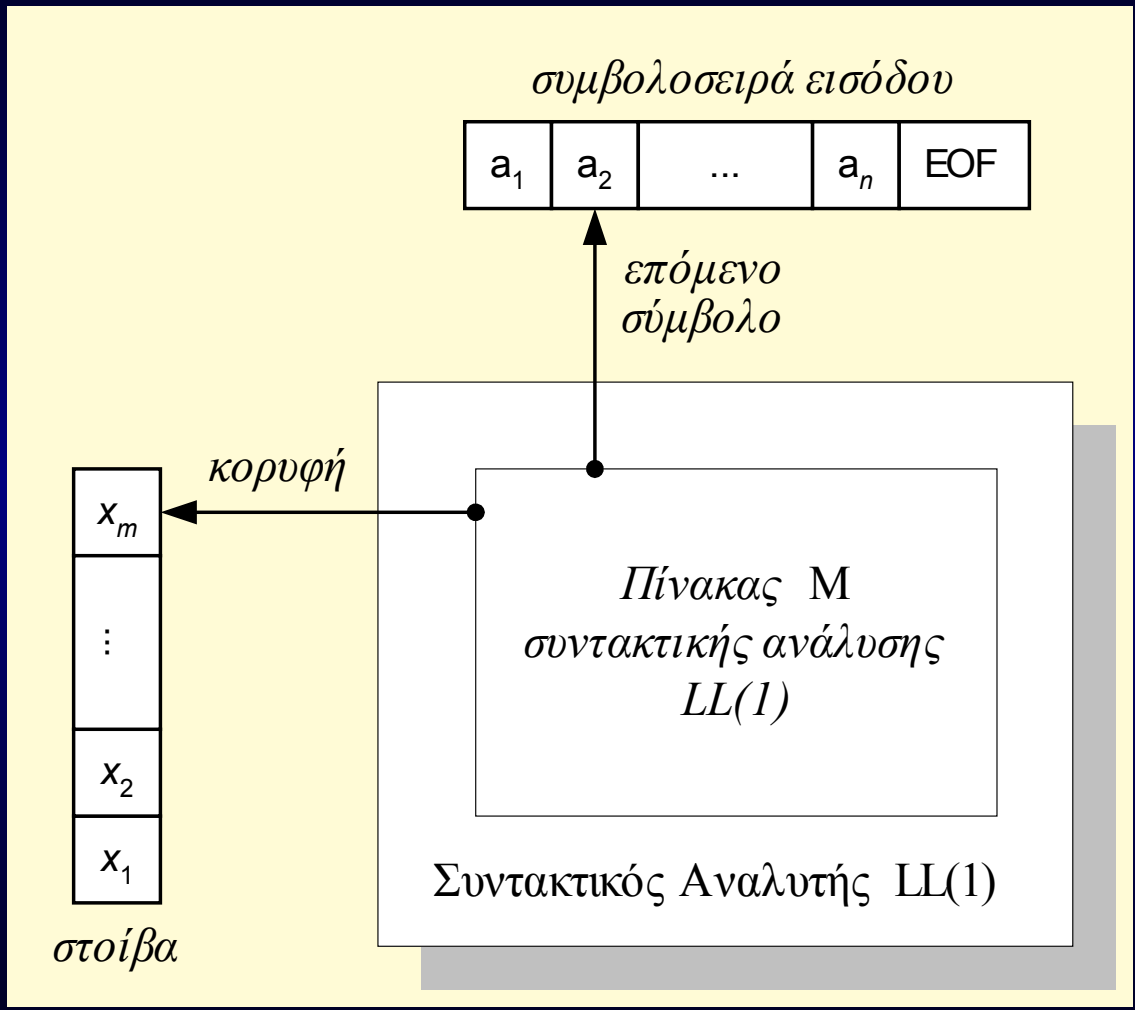
if *token* $\in$ FIRST(α_1) **then**
  code to recognize α_1
  ...
else if *token* $\in$ FIRST(α_n) **then**
  code to recognize α_n
else if $\epsilon \notin$ FIRST$(\alpha_1) \cup \ldots \cup$ FIRST(α_n) **then**
  syntax error
else if *token* $\notin$ FOLLOW(A) **then**
  syntax error
end if

LL(1) parsers (i)

- They use a stack where they put grammar symbols — initially just S
- Every time, they examine the top of the stack:
  - If it is a terminal symbol and it's the same as the next input symbol, then both are removed
  - If it is a non-terminal symbol, then they apply a rule depending on the next symbol of the input
- Success: both the stack and the input string are empty
  άδειες

LL(1) parsers (ii)

The algorithm for generating table M defines the class of LL(1) languages



Generating LL(1) parsers

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid \text{id}$

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{\text{id}, (\}$
$\text{FIRST}(E') = \{+, \epsilon\}$
$\text{FIRST}(T') = \{*, \epsilon\}$
$\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{), \texttt{EOF}\}$
$\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{+,), \texttt{EOF}\}$
$\text{FOLLOW}(F) = \{*, +,), \texttt{EOF}\}$

	id	+	*	(	)	EOF
E						
E'						
T						
T'						
F						

Generating LL(1) parsers

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid \text{id}$

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{\text{id}, (\}$
$\text{FIRST}(E') = \{+, \epsilon\}$
$\text{FIRST}(T') = \{*, \epsilon\}$
$\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{), \texttt{EOF}\}$
$\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{+,), \texttt{EOF}\}$
$\text{FOLLOW}(F) = \{*, +,), \texttt{EOF}\}$

	id	+	*	(	)	EOF
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'						
T						
T'						
F						

Generating LL(1) parsers

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid \text{id}$

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{\text{id}, (\}$
$\text{FIRST}(E') = \{+, \epsilon\}$
$\text{FIRST}(T') = \{*, \epsilon\}$
$\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{), \texttt{EOF}\}$
$\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{+,), \texttt{EOF}\}$
$\text{FOLLOW}(F) = \{*, +,), \texttt{EOF}\}$

	id	+	*	(	)	EOF
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$				
T						
T'						
F						

Generating LL(1) parsers

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid \text{id}$

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{\text{id}, (\}$
$\text{FIRST}(E') = \{+, \epsilon\}$
$\text{FIRST}(T') = \{*, \epsilon\}$
$\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{), \texttt{EOF}\}$
$\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{+,), \texttt{EOF}\}$
$\text{FOLLOW}(F) = \{*, +,), \texttt{EOF}\}$

	id	+	*	(	)	EOF
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T						
T'						
F						

Generating LL(1) parsers

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid \text{id}$

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{\text{id}, (\}$
$\text{FIRST}(E') = \{+, \epsilon\}$
$\text{FIRST}(T') = \{*, \epsilon\}$
$\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{), \texttt{EOF}\}$
$\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{+,), \texttt{EOF}\}$
$\text{FOLLOW}(F) = \{*, +,), \texttt{EOF}\}$

	id	+	*	(	)	EOF
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'						
F						

Generating LL(1) parsers

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \epsilon \\
F &\rightarrow (\ E\ ) \mid \text{id}
\end{aligned}
$$

$$
\begin{aligned}
&\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{\text{id}, (\} \\
&\text{FIRST}(E') = \{+, \epsilon\} \\
&\text{FIRST}(T') = \{*, \epsilon\} \\
&\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{), \texttt{EOF}\} \\
&\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{+,), \texttt{EOF}\} \\
&\text{FOLLOW}(F) = \{*, +,), \texttt{EOF}\}
\end{aligned}
$$

	id	+	*	(	)	EOF
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'			$T' \rightarrow *FT'$			
F						

Generating LL(1) parsers

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid \text{id}$

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{\text{id}, (\}$
$\text{FIRST}(E') = \{+, \epsilon\}$
$\text{FIRST}(T') = \{*, \epsilon\}$
$\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{), \texttt{EOF}\}$
$\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{+,), \texttt{EOF}\}$
$\text{FOLLOW}(F) = \{*, +,), \texttt{EOF}\}$

	id	+	*	(	)	EOF
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F						

Generating LL(1) parsers

$$
\begin{array}{lcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathrm{id}
\end{array}
$$

$$
\begin{array}{l}
\mathrm{FIRST}(E) = \mathrm{FIRST}(T) = \mathrm{FIRST}(F) = \{\mathrm{id}, (\} \\
\mathrm{FIRST}(E') = \{+, \epsilon\} \\
\mathrm{FIRST}(T') = \{*, \epsilon\} \\
\mathrm{FOLLOW}(E) = \mathrm{FOLLOW}(E') = \{), \texttt{EOF}\} \\
\mathrm{FOLLOW}(T) = \mathrm{FOLLOW}(T') = \{+,), \texttt{EOF}\} \\
\mathrm{FOLLOW}(F) = \{*, +,), \texttt{EOF}\}
\end{array}
$$

	id	+	*	(	)	EOF
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F				$F \rightarrow (E)$		

Generating LL(1) parsers

$E \rightarrow T\ E'$
$E' \rightarrow +\ T\ E' \mid \epsilon$
$T \rightarrow F\ T'$
$T' \rightarrow *\ F\ T' \mid \epsilon$
$F \rightarrow (\ E\ ) \mid \text{id}$

$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{\text{id}, (\}$
$\text{FIRST}(E') = \{+, \epsilon\}$
$\text{FIRST}(T') = \{*, \epsilon\}$
$\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{), \texttt{EOF}\}$
$\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{+,), \texttt{EOF}\}$
$\text{FOLLOW}(F) = \{*, +,), \texttt{EOF}\}$

	id	+	*	(	)	EOF
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow \text{id}$			$F \rightarrow (E)$		

Operation of a LL(1) parser

0	*E*	id + id * id `EOF`

Operation of a LL(1) parser

0	E	id + id * id `EOF`	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id `EOF`	

Operation of a LL(1) parser

0	E	id + id * id `EOF`	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id `EOF`	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id `EOF`	

Operation of a LL(1) parser

0	E	id + id * id `EOF`	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id `EOF`	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id `EOF`	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id `EOF`	

Operation of a LL(1) parser

0	E	id + id * id `EOF`	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id `EOF`	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id `EOF`	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id `EOF`	
4	$E'\ T'$	+ id * id `EOF`	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	

Operation of a LL(1) parser

0	E	id + id * id `EOF`	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id `EOF`	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id `EOF`	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id `EOF`	
4	$E'\ T'$	+ id * id `EOF`	$T' \rightarrow \epsilon$
5	E'	+ id * id `EOF`	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id `EOF`	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	$F \rightarrow$ id
9	$E'\ T'$ id	id * id EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	$F \rightarrow$ id
9	$E'\ T'$ id	id * id EOF	
10	$E'\ T'$	* id EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	$F \rightarrow$ id
9	$E'\ T'$ id	id * id EOF	
10	$E'\ T'$	* id EOF	$T' \rightarrow *\ F\ T'$
11	$E'\ T'\ F$ *	* id EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	$F \rightarrow$ id
9	$E'\ T'$ id	id * id EOF	
10	$E'\ T'$	* id EOF	$T' \rightarrow *\ F\ T'$
11	$E'\ T'\ F$ *	* id EOF	
12	$E'\ T'\ F$	id EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	$F \rightarrow$ id
9	$E'\ T'$ id	id * id EOF	
10	$E'\ T'$	* id EOF	$T' \rightarrow *\ F\ T'$
11	$E'\ T'\ F$ *	* id EOF	
12	$E'\ T'\ F$	id EOF	$F \rightarrow$ id
13	$E'\ T'$ id	id EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	$F \rightarrow$ id
9	$E'\ T'$ id	id * id EOF	
10	$E'\ T'$	* id EOF	$T' \rightarrow *\ F\ T'$
11	$E'\ T'\ F$ *	* id EOF	
12	$E'\ T'\ F$	id EOF	$F \rightarrow$ id
13	$E'\ T'$ id	id EOF	
14	$E'\ T'$	EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	$F \rightarrow$ id
9	$E'\ T'$ id	id * id EOF	
10	$E'\ T'$	* id EOF	$T' \rightarrow *\ F\ T'$
11	$E'\ T'\ F$ *	* id EOF	
12	$E'\ T'\ F$	id EOF	$F \rightarrow$ id
13	$E'\ T'$ id	id EOF	
14	$E'\ T'$	EOF	$T' \rightarrow \epsilon$
15	E'	EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	$F \rightarrow$ id
9	$E'\ T'$ id	id * id EOF	
10	$E'\ T'$	* id EOF	$T' \rightarrow *\ F\ T'$
11	$E'\ T'\ F$ *	* id EOF	
12	$E'\ T'\ F$	id EOF	$F \rightarrow$ id
13	$E'\ T'$ id	id EOF	
14	$E'\ T'$	EOF	$T' \rightarrow \epsilon$
15	E'	EOF	$E' \rightarrow \epsilon$
16	ϵ	EOF	

Operation of a LL(1) parser

0	E	id + id * id EOF	$E \rightarrow T\ E'$
1	$E'\ T$	id + id * id EOF	$T \rightarrow F\ T'$
2	$E'\ T'\ F$	id + id * id EOF	$F \rightarrow$ id
3	$E'\ T'$ id	id + id * id EOF	
4	$E'\ T'$	+ id * id EOF	$T' \rightarrow \epsilon$
5	E'	+ id * id EOF	$E' \rightarrow +\ T\ E'$
6	$E'\ T$ +	+ id * id EOF	
7	$E'\ T$	id * id EOF	$T \rightarrow F\ T'$
8	$E'\ T'\ F$	id * id EOF	$F \rightarrow$ id
9	$E'\ T'$ id	id * id EOF	
10	$E'\ T'$	* id EOF	$T' \rightarrow *\ F\ T'$
11	$E'\ T'\ F$ *	* id EOF	
12	$E'\ T'\ F$	id EOF	$F \rightarrow$ id
13	$E'\ T'$ id	id EOF	
14	$E'\ T'$	EOF	$T' \rightarrow \epsilon$
15	E'	EOF	$E' \rightarrow \epsilon$
16	ϵ	EOF	success

Bottom-up parsing (i)

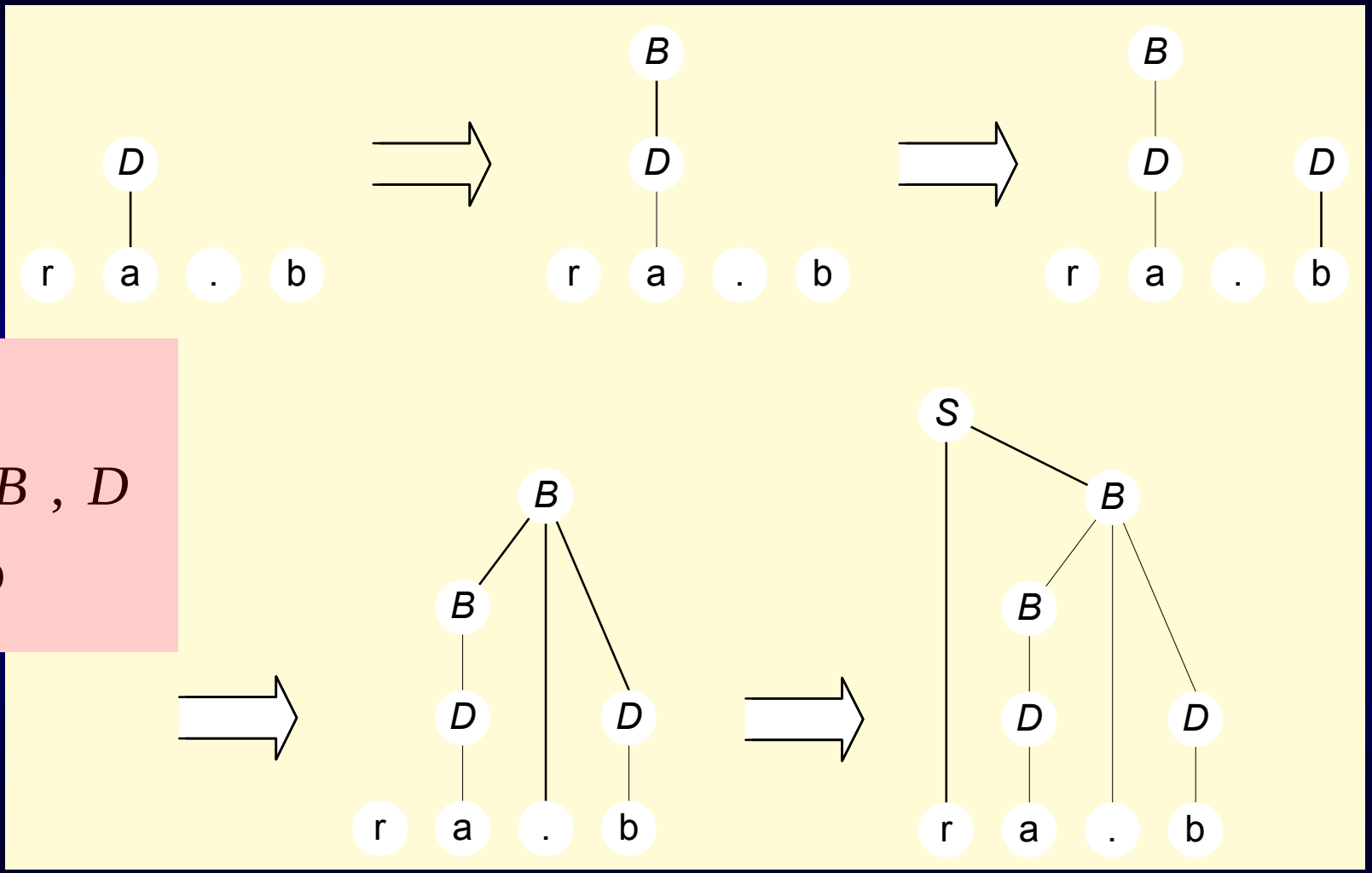
- Parsing starts from the leaves
- Every time, we seek:
  - the left-most node of the tree
  - that hasn't yet been built
  - but all its children have been built
- We repeat until the root is built
- Reducing: the selection of the nodes that will become the children of a new node

Bottom-up parsing (ii)

$S \rightarrow \text{r}\ B$

$B \rightarrow D \mid B\ ,\ D$

$D \rightarrow \text{a} \mid \text{b}$



Bottom-up parsing (iii)

- Shift-reduce parsers
  - They use an initially empty stack where they put symbols of the grammar
  - Shift: move the next input symbol to the top of the stack
  - Reduce: remove from the top of the stack the RHS of some rule and add the corresponding LHS
  - Success: the stack contains only S and all input symbols have been exhausted

Bottom-up parsing (iv)

$$S \rightarrow \text{r } B$$
$$B \rightarrow D \mid B \text{ , } D$$
$$D \rightarrow \text{a} \mid \text{b}$$

step	stack	input	action
0	ϵ	r a , b	

Bottom-up parsing (iv)

$$S \rightarrow \text{r } B$$
$$B \rightarrow D \mid B \text{ , } D$$
$$D \rightarrow \text{a} \mid \text{b}$$

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	

Bottom-up parsing (iv)

$$S \rightarrow \text{r } B$$
$$B \rightarrow D \mid B \text{ , } D$$
$$D \rightarrow \text{a} \mid \text{b}$$

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	shift
2	r a	, b	

Bottom-up parsing (iv)

$$S \rightarrow \text{r } B$$
$$B \rightarrow D \mid B \text{ , } D$$
$$D \rightarrow \text{a} \mid \text{b}$$

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	shift
2	r a	, b	reduce with $D \rightarrow$ a
3	r D	, b	

Bottom-up parsing (iv)

$S \rightarrow \text{r}\ B$

$B \rightarrow D \mid B\ ,\ D$

$D \rightarrow \text{a} \mid \text{b}$

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	shift
2	r a	, b	reduce with $D \rightarrow$ a
3	r D	, b	reduce with $B \rightarrow D$
4	r B	, b	

Bottom-up parsing (iv)

$S \rightarrow \text{r } B$

$B \rightarrow D \mid B \text{ , } D$

$D \rightarrow \text{a} \mid \text{b}$

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	shift
2	r a	, b	reduce with $D \rightarrow$ a
3	r D	, b	reduce with $B \rightarrow D$
4	r B	, b	shift
			(not reduce with $S \rightarrow$ r B)
5	r B ,	b	

Bottom-up parsing (iv)

$S \rightarrow \text{r } B$

$B \rightarrow D \mid B \text{ , } D$

$D \rightarrow \text{a} \mid \text{b}$

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	shift
2	r a	, b	reduce with $D \rightarrow$ a
3	r D	, b	reduce with $B \rightarrow D$
4	r B	, b	shift
			(not reduce with $S \rightarrow$ r B)
5	r B ,	b	shift
6	r B , b	ϵ	

Bottom-up parsing (iv)

$S \rightarrow \text{r } B$

$B \rightarrow D \mid B \text{ , } D$

$D \rightarrow \text{a} \mid \text{b}$

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	shift
2	r a	, b	reduce with $D \rightarrow$ a
3	r D	, b	reduce with $B \rightarrow D$
4	r B	, b	shift
			(not reduce with $S \rightarrow$ r B)
5	r B ,	b	shift
6	r B , b	ϵ	reduce with $D \rightarrow$ b
7	r B , D	ϵ	

Bottom-up parsing (iv)

$S \rightarrow$ r B

$B \rightarrow D \mid B$, D

$D \rightarrow$ a $\mid$ b

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	shift
2	r a	, b	reduce with $D \rightarrow$ a
3	r D	, b	reduce with $B \rightarrow D$
4	r B	, b	shift
			(not reduce with $S \rightarrow$ r B)
5	r B ,	b	shift
6	r B , b	ϵ	reduce with $D \rightarrow$ b
7	r B , D	ϵ	reduce with $B \rightarrow B$, D
			(not reduce with $B \rightarrow D$)
8	r B	ϵ	

Bottom-up parsing (iv)

$S \rightarrow \text{r } B$

$B \rightarrow D \mid B \text{ , } D$

$D \rightarrow \text{a} \mid \text{b}$

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	shift
2	r a	, b	reduce with $D \rightarrow$ a
3	r D	, b	reduce with $B \rightarrow D$
4	r B	, b	shift
			(not reduce with $S \rightarrow$ r B)
5	r B ,	b	shift
6	r B , b	ϵ	reduce with $D \rightarrow$ b
7	r B , D	ϵ	reduce with $B \rightarrow B$, D
			(not reduce with $B \rightarrow D$)
8	r B	ϵ	reduce with $S \rightarrow$ r B
9	S	ϵ	

Bottom-up parsing (iv)

$S \rightarrow$ r B

$B \rightarrow D \mid B$, D

$D \rightarrow$ a $\mid$ b

step	stack	input	action
0	ϵ	r a , b	shift
1	r	a , b	shift
2	r a	, b	reduce with $D \rightarrow$ a
3	r D	, b	reduce with $B \rightarrow D$
4	r B	, b	shift
			(not reduce with $S \rightarrow$ r B)
5	r B ,	b	shift
6	r B , b	ϵ	reduce with $D \rightarrow$ b
7	r B , D	ϵ	reduce with $B \rightarrow B$, D
			(not reduce with $B \rightarrow D$)
8	r B	ϵ	reduce with $S \rightarrow$ r B
9	S	ϵ	success

Bottom-up parsing (v)

- LR(k)
- LR(0)
- SLR(1)
- LALR(1)
- LR(1)