Compiler Construction

Lecture 8: Syntax Analysis IV (More on $LL(1)$ & Bottom-Up Parsing)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH AACHEN UNIVERSITY

noll@cs.rwth-aachen.de

http://moves.rwth-aachen.de/teaching/ss-14/cc14/

Summer Semester 2014

Outline

Characterization of $LL(1)$

Theorem (Characterization of $LL(1)$)

$G \in LL(1)$ iff for all pairs of rules $A \to \beta \mid \gamma \in P$ (where $\beta \neq \gamma$):

$$\mathrm{la}(A \to \beta) \cap \mathrm{la}(A \to \gamma) = \emptyset.$$

Proof.

on the board □

Remark: the above theorem generally does not hold if $k > 1$ (cf. exercises)

Deterministic Top-Down Parsing

Approach: given $G \in CFG_\Sigma$,

1. Verify that $G \in LL(1)$ by computing the lookahead sets and checking alternatives for disjointness
2. Start with nondeterministic top-down parsing automaton $\mathrm{NTA}(G)$
3. Use 1-symbol lookahead to control the choice of expanding productions:
   - $(aw, A\alpha, z) \vdash (aw, \beta\alpha, zi)$
     if $\pi_i = A \to \beta$ and $a \in \mathrm{la}(\pi_i)$
   - $(\varepsilon, A\alpha, z) \vdash (\varepsilon, \beta\alpha, zi)$
     if $\pi_i = A \to \beta$ and $\varepsilon \in \mathrm{la}(\pi_i)$
   - [matching steps as before: $(aw, a\alpha, z) \vdash (w, \alpha, z)$]

$\Longrightarrow$ deterministic top-down parsing automaton $\mathrm{DTA}(G)$

Remarks:

- $\mathrm{DTA}(G)$ is actually not a pushdown automaton (a is read but not consumed). But: can be simulated using the finite control.
- Advantage of using lookahead is twofold:
  - Removal of nondeterminism
  - Earlier detection of syntax errors
    (in configurations $(aw, A\alpha, z)$ where $a \notin \bigcup_{A \to \beta \in P} \mathrm{la}(A \to \beta)$)

Outline

Transformation to $LL(1)$

Assume that $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma \setminus LL(1)$
(i.e., there exist $A \to \beta \mid \gamma \in P$ such that $\mathrm{la}(A \to \beta) \cap \mathrm{la}(A \to \gamma) \neq \emptyset$)

Transformation to $LL(1)$

Assume that $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma \setminus LL(1)$
(i.e., there exist $A \to \beta \mid \gamma \in P$ such that $\mathrm{la}(A \to \beta) \cap \mathrm{la}(A \to \gamma) \neq \emptyset$)

Two heuristics for transforming G into $G' \in LL(1)$:

1. Removal of left recursion
2. Left factorization

(used in parser-generating systems such as ANTLR)

Transformation to $LL(1)$

Assume that $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma \setminus LL(1)$
(i.e., there exist $A \to \beta \mid \gamma \in P$ such that $\mathrm{la}(A \to \beta) \cap \mathrm{la}(A \to \gamma) \neq \emptyset$)

Two heuristics for transforming G into $G' \in LL(1)$:

1. Removal of left recursion
2. Left factorization

(used in parser-generating systems such as ANTLR)

Remarks:

- Transformations generally preserve the semantics (= generated language) of CFGs but not the syntactic structure of words (different syntax trees).
- Transformations cannot always yield an $LL(1)$ grammar (since not every context-free language is generated by an LL grammar; details later).

Left Recursion I

Definition 8.1 (Left recursion)

A grammar $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ is called left recursive if there exist $A \in N$ and $\alpha \in X^*$ such that $A \Rightarrow^+ A\alpha$.

Left Recursion I

Definition 8.1 (Left recursion)

A grammar $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ is called left recursive if there exist $A \in N$ and $\alpha \in X^*$ such that $A \Rightarrow^+ A\alpha$.

Corollary 8.2

If $G \in CFG_\Sigma$ is left recursive with $A \Rightarrow^+ A\alpha$, then there exists $\beta \in X^*$ such that $A \Rightarrow_l^+ A\beta$.

Left Recursion I

Definition 8.1 (Left recursion)

A grammar $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ is called left recursive if there exist $A \in N$ and $\alpha \in X^*$ such that $A \Rightarrow^+ A\alpha$.

Corollary 8.2

If $G \in CFG_\Sigma$ is left recursive with $A \Rightarrow^+ A\alpha$, then there exists $\beta \in X^*$ such that $A \Rightarrow_l^+ A\beta$.

Example 8.3

The grammar (cf. Example 5.10)

$$G_{AE}: \quad \begin{aligned} E &\to E\texttt{+}T \mid T \\ T &\to T\texttt{*}F \mid F \\ F &\to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} \end{aligned}$$

is left recursive, and in Example 7.4 it was shown that $G_{AE} \notin LL(1)$

Left Recursion II

Lemma 8.4

If $G \in CFG_\Sigma$ *is left recursive, then* $G \notin \bigcup_{k \in \mathbb{N}} LL(k)$.

Left Recursion II

Lemma 8.4

If $G \in CFG_\Sigma$ *is left recursive, then* $G \notin \bigcup_{k \in \mathbb{N}} LL(k)$.

Proof.

(for $k = 1$) Assume that $G \in LL(1)$ is left recursive with $A \Rightarrow_l^+ A\beta$.
Together with the reducedness of G this implies that
$S \Rightarrow_l^* vA\alpha \Rightarrow_l^+ vA\beta\alpha \Rightarrow_l^+ vw$ for some $v, w \in \Sigma^*$ and $\alpha \in X^*$.

Left Recursion II

Lemma 8.4

If $G \in CFG_\Sigma$ is left recursive, then $G \notin \bigcup_{k \in \mathbb{N}} LL(k)$.

Proof.

(for $k = 1$) Assume that $G \in LL(1)$ is left recursive with $A \Rightarrow_l^+ A\beta$.
Together with the reducedness of G this implies that
$S \Rightarrow_l^* vA\alpha \Rightarrow_l^+ vA\beta\alpha \Rightarrow_l^+ vw$ for some $v, w \in \Sigma^*$ and $\alpha \in X^*$.
The corresponding computation of $\mathrm{DTA}(G)$ (Def. 7.6) starts with
$(vw, S, \varepsilon) \vdash^* (w, A\alpha, \ldots) \vdash^+ (w, A\beta\alpha, \ldots)$.

Left Recursion II

Lemma 8.4

If $G \in CFG_\Sigma$ is left recursive, then $G \notin \bigcup_{k \in \mathbb{N}} LL(k)$.

Proof.

(for $k = 1$) Assume that $G \in LL(1)$ is left recursive with $A \Rightarrow_l^+ A\beta$.
Together with the reducedness of G this implies that
$S \Rightarrow_l^* vA\alpha \Rightarrow_l^+ vA\beta\alpha \Rightarrow_l^+ vw$ for some $v, w \in \Sigma^*$ and $\alpha \in X^*$.
The corresponding computation of $\mathrm{DTA}(G)$ (Def. 7.6) starts with
$(vw, S, \varepsilon) \vdash^* (w, A\alpha, \ldots) \vdash^+ (w, A\beta\alpha, \ldots)$.
But in the last state the behaviour of $\mathrm{DTA}(G)$ is determined by the same input ($\mathrm{fi}(w)$) and stack symbol (A). Thus it enters a loop of the form
$(w, A\alpha, \ldots) \vdash^+ (w, A\beta\alpha, \ldots) \vdash^+ (w, A\beta\beta\alpha, \ldots) \vdash^+ \ldots$ and will never recognize w. Contradiction □

Removing Direct Left Recursion

Direct left recursion occurs in productions of the form

$A \to A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \ldots \mid \beta_n$ where $\alpha_i \neq \varepsilon$ and $\beta_j \neq A\ldots$

Removing Direct Left Recursion

Direct left recursion occurs in productions of the form

$$A \to A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \ldots \mid \beta_n \qquad \text{where } \alpha_i \neq \varepsilon \text{ and } \beta_j \neq A\ldots$$

Transformation: replacement by right recursion

$$\begin{aligned} A &\to \beta_1 A' \mid \ldots \mid \beta_n A' \\ A' &\to \alpha_1 A' \mid \ldots \mid \alpha_m A' \mid \varepsilon \end{aligned}$$

(with a new $A' \in N$) which preserves $L(G)$.

Removing Direct Left Recursion

Direct left recursion occurs in productions of the form

$$A \to A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \ldots \mid \beta_n \qquad \text{where } \alpha_i \neq \varepsilon \text{ and } \beta_j \neq A\ldots$$

Transformation: replacement by right recursion

$$\begin{aligned} A &\to \beta_1 A' \mid \ldots \mid \beta_n A' \\ A' &\to \alpha_1 A' \mid \ldots \mid \alpha_m A' \mid \varepsilon \end{aligned}$$

(with a new $A' \in N$) which preserves $L(G)$.

Example 8.5

$$G_{AE}: \quad \begin{aligned} E &\to E\texttt{+}T \mid T \\ T &\to T\texttt{*}F \mid F \\ F &\to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} \end{aligned}$$

is transformed into

$$G'_{AE}: \quad \begin{aligned} E &\to TE' \\ E' &\to \texttt{+}TE' \mid \varepsilon \\ T &\to FT' \\ T' &\to \texttt{*}FT' \mid \varepsilon \\ F &\to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} \end{aligned}$$

with $G'_{AE} \in LL(1)$ (see Example 7.5).

Removing Indirect Left Recursion

Indirect left recursion occurs in productions of the form ($n \geq 1$)

$$\begin{array}{ll} A & \to A_1\alpha_1 \mid \dots \\ A_1 & \to A_2\alpha_2 \mid \dots \\ & \vdots \\ A_{n-1} & \to A_n\alpha_n \mid \dots \\ A_n & \to A\beta \mid \dots \end{array}$$

Removing Indirect Left Recursion

Indirect left recursion occurs in productions of the form ($n \geq 1$)

$$\begin{array}{ll} A & \to A_1\alpha_1 \mid \ldots \\ A_1 & \to A_2\alpha_2 \mid \ldots \\ & \vdots \\ A_{n-1} & \to A_n\alpha_n \mid \ldots \\ A_n & \to A\beta \mid \ldots \end{array}$$

Transformation: into Greibach Normal Form with productions of the form

$A \to aB_1 \ldots B_n$ (where $n \in \mathbb{N}$ and each $B_i \neq S$) or
$S \to \varepsilon$

(cf. *Formale Systeme, Automaten, Prozesse*)

Left Factorization

Applies to productions of the form

$$A \rightarrow \alpha\beta \mid \alpha\gamma$$

which are problematic if α "at least as long as lookahead".

Left Factorization

Applies to productions of the form

$$A \to \alpha\beta \mid \alpha\gamma$$

which are problematic if α "at least as long as lookahead".

Transformation: delaying the decision by left factorization

$$\begin{aligned} A &\to \alpha A' \\ A' &\to \beta \mid \gamma \end{aligned}$$

(with a new $A' \in N$) which preserves $L(G)$.

Left Factorization

Applies to productions of the form

$$A \to \alpha\beta \mid \alpha\gamma$$

which are problematic if α "at least as long as lookahead".

Transformation: delaying the decision by left factorization

$$\begin{aligned} A &\to \alpha A' \\ A' &\to \beta \mid \gamma \end{aligned}$$

(with a new $A' \in N$) which preserves $L(G)$.

Example 8.6

$$\begin{aligned} \mathit{Statement} &\to \text{if } \mathit{Condition} \text{ then } \mathit{Statement} \text{ else } \mathit{Statement} \text{ fi} \\ &\mid \text{ if } \mathit{Condition} \text{ then } \mathit{Statement} \text{ fi} \end{aligned}$$

is transformed into

$$\begin{aligned} \mathit{Statement} &\to \text{if } \mathit{Condition} \text{ then } \mathit{Statement}\ S' \\ S' &\to \text{else } \mathit{Statement} \text{ fi} \mid \text{fi} \end{aligned}$$

Outline

The Complexity of $LL(1)$ Parsing I

- $LL(1)$ parsing has time (and hence space) complexity $\mathcal{O}(|w|)$ (where $w \in \Sigma^*$ is the input word)

The Complexity of $LL(1)$ Parsing I

- $LL(1)$ parsing has time (and hence space) complexity $\mathcal{O}(|w|)$ (where $w \in \Sigma^*$ is the input word)
- Here: proof for ε-free grammars (i.e., $A \to \alpha \in P \implies \alpha \neq \varepsilon$)

The Complexity of $LL(1)$ Parsing I

- $LL(1)$ parsing has time (and hence space) complexity $\mathcal{O}(|w|)$ (where $w \in \Sigma^*$ is the input word)
- Here: proof for ε-free grammars (i.e., $A \to \alpha \in P \implies \alpha \neq \varepsilon$)
- General case: see O. Mayer: *Syntaxanalyse*, p. 211ff

The Complexity of $LL(1)$ Parsing I

- $LL(1)$ parsing has time (and hence space) complexity $\mathcal{O}(|w|)$ (where $w \in \Sigma^*$ is the input word)
- Here: proof for ε-free grammars (i.e., $A \to \alpha \in P \implies \alpha \neq \varepsilon$)
- General case: see O. Mayer: *Syntaxanalyse*, p. 211ff

Lemma 8.7

Let $G = \langle N, \Sigma, P, S \rangle \in LL(1)$ *be* ε*-free. If*

$$(w, S, \varepsilon) \vdash^n (\varepsilon, \varepsilon, z)$$

in $\mathrm{DTA}(G)$, *then*

$$n \leq (|w| + 1) \cdot (|N| + 1).$$

Proof.

Let $(w, S, \varepsilon) \vdash^n (\varepsilon, \varepsilon, z)$ in $\mathrm{DTA}(G)$. To show: $n \leq (|w| + 1) \cdot (|N| + 1)$

1. Clear: the computation involves $|w|$ matching steps.

□

The Complexity of $LL(1)$ Parsing II

Proof.

Let $(w, S, \varepsilon) \vdash^n (\varepsilon, \varepsilon, z)$ in $\mathrm{DTA}(G)$. To show: $n \leq (|w| + 1) \cdot (|N| + 1)$

1. Clear: the computation involves $|w|$ matching steps.
2. Since G is ε-free, every matching step is preceded (and followed) by $k \geq 0$ expansion steps of the form

$$\begin{aligned}(av, A_1\alpha_1, \ldots) &\vdash (av, A_2\alpha_2\alpha_1, \ldots)\\ &\vdots\\ &\vdash (av, A_k\alpha_k \ldots \alpha_1, \ldots)\\ &\vdash (av, a\alpha_{k+1} \ldots \alpha_1, \ldots)\end{aligned}$$

where $A_i \to A_{i+1}\alpha_{i+1}$ for each $i \in [k-1]$ and $A_k \to a\alpha_{k+1}$.

□

Proof.

Let $(w, S, \varepsilon) \vdash^n (\varepsilon, \varepsilon, z)$ in $\mathrm{DTA}(G)$. To show: $n \leq (|w| + 1) \cdot (|N| + 1)$

1. Clear: the computation involves $|w|$ matching steps.
2. Since G is ε-free, every matching step is preceded (and followed) by $k \geq 0$ expansion steps of the form

$$\begin{aligned} (av, A_1\alpha_1, \ldots) &\vdash (av, A_2\alpha_2\alpha_1, \ldots) \\ &\vdots \\ &\vdash (av, A_k\alpha_k \ldots \alpha_1, \ldots) \\ &\vdash (av, a\alpha_{k+1} \ldots \alpha_1, \ldots) \end{aligned}$$

   where $A_i \to A_{i+1}\alpha_{i+1}$ for each $i \in [k-1]$ and $A_k \to a\alpha_{k+1}$.
3. This implies that $A_i \neq A_j$ for $i \neq j$ (by Lemma 8.4, G is not left recursive), and hence $k \leq |N|$.

□

The Complexity of $LL(1)$ Parsing II

Proof.

Let $(w, S, \varepsilon) \vdash^n (\varepsilon, \varepsilon, z)$ in $\mathrm{DTA}(G)$. To show: $n \leq (|w| + 1) \cdot (|N| + 1)$

1. Clear: the computation involves $|w|$ matching steps.
2. Since G is ε-free, every matching step is preceded (and followed) by $k \geq 0$ expansion steps of the form
$$\begin{aligned}(av, A_1\alpha_1, \ldots) &\vdash (av, A_2\alpha_2\alpha_1, \ldots)\\ &\vdots\\ &\vdash (av, A_k\alpha_k \ldots \alpha_1, \ldots)\\ &\vdash (av, a\alpha_{k+1} \ldots \alpha_1, \ldots)\end{aligned}$$
where $A_i \to A_{i+1}\alpha_{i+1}$ for each $i \in [k-1]$ and $A_k \to a\alpha_{k+1}$.
3. This implies that $A_i \neq A_j$ for $i \neq j$ (by Lemma 8.4, G is not left recursive), and hence $k \leq |N|$.
4. Altogether: $n \leq (|w| + 1) \cdot (|N| + 1)$.

□

Outline

Recursive-Descent Parsing I

Idea: avoid explicit use of pushdown store (as in $\mathrm{DTA}(G)$) by employing recursive procedures (with implicit runtime stack)

Recursive-Descent Parsing I

Idea: avoid explicit use of pushdown store (as in $\mathrm{DTA}(G)$) by employing recursive procedures (with implicit runtime stack)

Advantage: simple implementation

Recursive-Descent Parsing I

Idea: avoid explicit use of pushdown store (as in $\mathrm{DTA}(G)$) by employing recursive procedures (with implicit runtime stack)

Advantage: simple implementation

Ingredients:

- variable `token` for current token
- function `next()` for invoking the scanner
- procedure `print(i)` for displaying the leftmost analysis (or errors)

Recursive-Descent Parsing I

Idea: avoid explicit use of pushdown store (as in $\mathrm{DTA}(G)$) by employing recursive procedures (with implicit runtime stack)

Advantage: simple implementation

Ingredients:
- variable `token` for current token
- function `next()` for invoking the scanner
- procedure `print(i)` for displaying the leftmost analysis (or errors)

Method: to every $A \in N$ we assign a procedure `A()` which
- tests `token` with regard to the lookahead sets of the A-productions,
- prints the corresponding rule number and
- evaluates the corresponding right-hand side as follows:
  - for $a \in \Sigma$: match `token`; call `next()`
  - for $A \in N$: call `A()`

Recursive-Descent Parsing II

Example 8.8 (Arithmetic expressions; cf. Example 8.5)

```
proc main();
  token := next(); E()
proc E();    (* E → T E' *)
  if token in {'(','a','b'} then print(1); T(); E'()
  else print(error); stop fi
proc E'();   (* E' → + T E' | ε *)
  if token = '+' then print(2); token := next(); T(); E'()
  elsif token in {EOF, ')'} then print(3)
  else print(error); stop fi
proc T();    (* T → F T' *)
  if token in {'(','a','b'} then print(4); F(); T'()
  else print(error); stop fi
proc T'();   (* T' → * F T' | ε *)
  if token = '*' then print(5); token := next(); F(); T'()
  elsif token in {'+',EOF,')'} then print(6)
  else print(error); stop fi
proc F();    (* F → ( E ) | a | b *)
  if token = '(' then print(7); token := next(); E();
    if token = ')' then token := next() else print(error); stop fi
  elsif token = 'a' then print(8); token := next()
  elsif token = 'b' then print(9); token := next()
  else print(error); stop fi
```

Outline

Repetition: Top-Down Parsing

Example 8.9

Grammar for
arithmetic expressions:

$$
\begin{array}{llll}
G_{AE}: & E \to E\texttt{+}T \mid T & & (1,2) \\
 & T \to T\texttt{*}F \mid F & & (3,4) \\
 & F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7)
\end{array}
$$

Repetition: Top-Down Parsing

Example 8.9

E

Grammar for arithmetic expressions:

$$G_{AE}: \quad \begin{array}{lll} E \rightarrow E\texttt{+}T \mid T & (1,2) \\ T \rightarrow T\texttt{*}F \mid F & (3,4) \\ F \rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (5,6,7) \end{array}$$

Leftmost analysis of `(a)*b`:

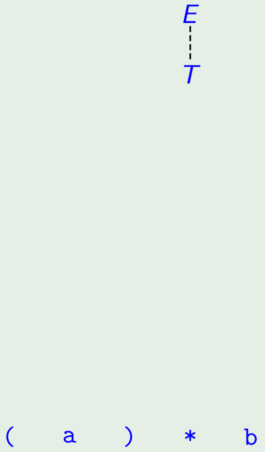
(a) * b

Example 8.9

Grammar for arithmetic expressions:

$$
\begin{array}{lllll}
G_{AE}: & E & \to & E\texttt{+}T \mid T & (1,2) \\
 & T & \to & T\texttt{*}F \mid F & (3,4) \\
 & F & \to & \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (5,6,7)
\end{array}
$$

Leftmost analysis of `(a)*b`:
2

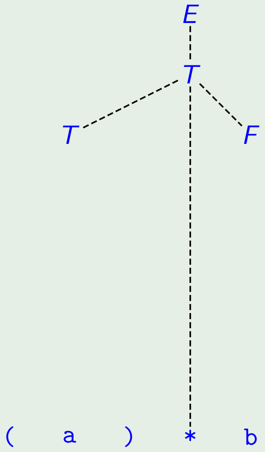


Example 8.9

Grammar for arithmetic expressions:

$$G_{AE}: \begin{array}{lll} E \to E\texttt{+}T \mid T & (1,2) \\ T \to T\texttt{*}F \mid F & (3,4) \\ F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (5,6,7) \end{array}$$

Leftmost analysis of `(a)*b`:
2 3

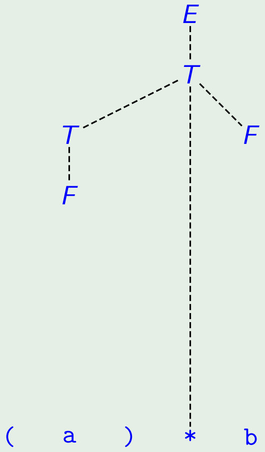


Example 8.9

Grammar for
arithmetic expressions:

$$G_{AE}: \quad \begin{array}{lll} E \rightarrow E{+}T \mid T & (1,2) \\ T \rightarrow T{*}F \mid F & (3,4) \\ F \rightarrow (E) \mid \texttt{a} \mid \texttt{b} & (5,6,7) \end{array}$$

Leftmost analysis of `(a)*b`:
2 3 4

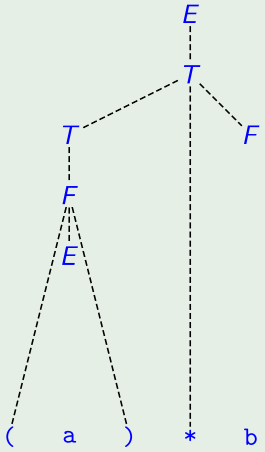


Example 8.9

Grammar for
arithmetic expressions:

$$
G_{AE}: \quad
\begin{array}{lll}
E \rightarrow E\texttt{+}T \mid T & & (1,2) \\
T \rightarrow T\texttt{*}F \mid F & & (3,4) \\
F \rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7)
\end{array}
$$

Leftmost analysis of `(a)*b`:
2 3 4 5

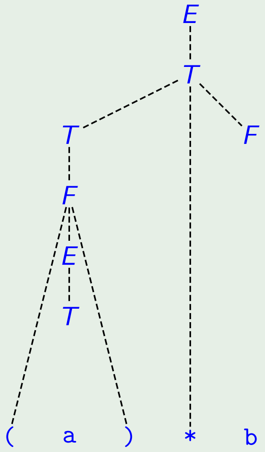


Example 8.9

Grammar for
arithmetic expressions:

$$G_{AE}: \quad \begin{array}{lll} E \rightarrow E{+}T \mid T & (1,2) \\ T \rightarrow T{*}F \mid F & (3,4) \\ F \rightarrow (E) \mid \mathtt{a} \mid \mathtt{b} & (5,6,7) \end{array}$$

Leftmost analysis of (a)*b:
2 3 4 5 2

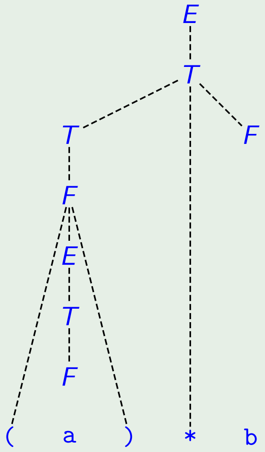


Example 8.9

Grammar for
arithmetic expressions:

$$G_{AE}: \begin{array}{lll} E \rightarrow E\texttt{+}T \mid T & (1,2) \\ T \rightarrow T\texttt{*}F \mid F & (3,4) \\ F \rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (5,6,7) \end{array}$$

Leftmost analysis of `(a)*b`:
2 3 4 5 2 4



Repetition: Top-Down Parsing

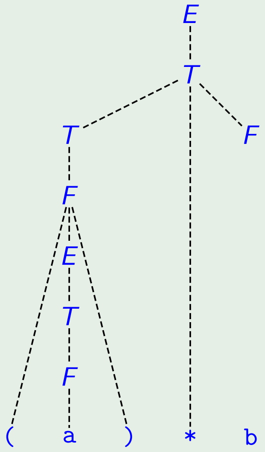
Example 8.9

Grammar for arithmetic expressions:

$$G_{AE}: \quad \begin{array}{lll} E \to E{+}T \mid T & (1,2) \\ T \to T{*}F \mid F & (3,4) \\ F \to (E) \mid \mathtt{a} \mid \mathtt{b} & (5,6,7) \end{array}$$

Leftmost analysis of `(a)*b`:

2 3 4 5 2 4 6



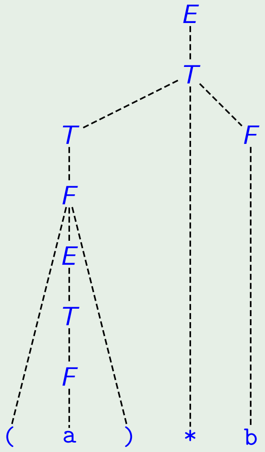
Repetition: Top-Down Parsing

Example 8.9

Grammar for arithmetic expressions:

$$
\begin{array}{llll}
G_{AE}: & E \rightarrow E\texttt{+}T \mid T & (1,2) \\
& T \rightarrow T\texttt{*}F \mid F & (3,4) \\
& F \rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (5,6,7)
\end{array}
$$

Leftmost analysis of `(a)*b`:
2 3 4 5 2 4 6 7



Example 8.10

Grammar for
arithmetic expressions:

$$
\begin{array}{llll}
G_{AE}: & E \rightarrow E\texttt{+}T \mid T & & (1,2) \\
& T \rightarrow T\texttt{*}F \mid F & & (3,4) \\
& F \rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7)
\end{array}
$$

Example 8.10

Grammar for arithmetic expressions:

$$
\begin{aligned}
G_{AE}: \quad & E \rightarrow E\texttt{+}T \mid T && (1,2)\\
& T \rightarrow T\texttt{*}F \mid F && (3,4)\\
& F \rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} && (5,6,7)
\end{aligned}
$$

Reversed rightmost analysis of `(a)*b`:

(a) * b

Example 8.10

Grammar for
arithmetic expressions:

$$
\begin{array}{llll}
G_{AE}: & E \rightarrow E{+}T \mid T & & (1,2) \\
 & T \rightarrow T{*}F \mid F & & (3,4) \\
 & F \rightarrow (E) \mid \mathtt{a} \mid \mathtt{b} & & (5,6,7)
\end{array}
$$

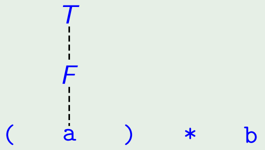
Reversed rightmost analysis
of (a)*b:
6

Example 8.10

Grammar for
arithmetic expressions:

$$G_{AE}: \quad \begin{array}{lll} E \to E{+}T \mid T & & (1,2) \\ T \to T{*}F \mid F & & (3,4) \\ F \to (E) \mid \mathtt{a} \mid \mathtt{b} & & (5,6,7) \end{array}$$

Reversed rightmost analysis
of (a)*b:
6 4



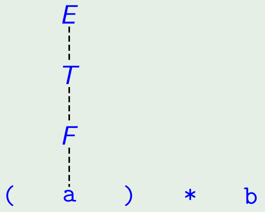
Example 8.10

Grammar for arithmetic expressions:

$$
\begin{array}{llll}
G_{AE}: & E \to E\texttt{+}T \mid T & & (1,2) \\
& T \to T\texttt{*}F \mid F & & (3,4) \\
& F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7)
\end{array}
$$

Reversed rightmost analysis of `(a)*b`:

6 4 2



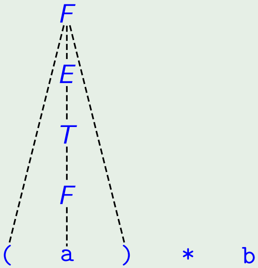
Example 8.10

Grammar for arithmetic expressions:

$$
\begin{array}{llll}
G_{AE}: & E \to E{+}T \mid T & (1,2) \\
& T \to T{*}F \mid F & (3,4) \\
& F \to (E) \mid \mathtt{a} \mid \mathtt{b} & (5,6,7)
\end{array}
$$

Reversed rightmost analysis of (a)*b:

6 4 2 5



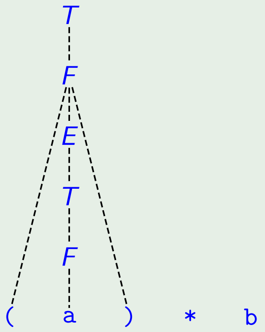
Example 8.10

Grammar for arithmetic expressions:

$$G_{AE}: \quad \begin{array}{lll} E \to E\texttt{+}T \mid T & & (1,2) \\ T \to T\texttt{*}F \mid F & & (3,4) \\ F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7) \end{array}$$

Reversed rightmost analysis of `(a)*b`:

6 4 2 5 4



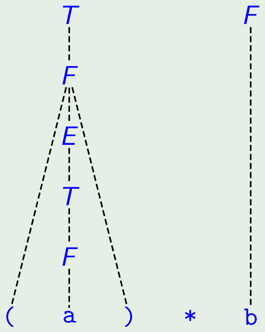
Example 8.10

Grammar for arithmetic expressions:

$$G_{AE}: \quad \begin{array}{lll} E \rightarrow E\texttt{+}T \mid T & & (1,2) \\ T \rightarrow T\texttt{*}F \mid F & & (3,4) \\ F \rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7) \end{array}$$

Reversed rightmost analysis of `(a)*b`:

6 4 2 5 4 7

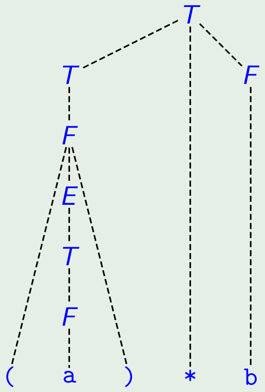


Example 8.10

Grammar for
arithmetic expressions:

$$G_{AE}: \quad \begin{aligned} E &\to E{+}T \mid T & (1,2)\\ T &\to T{*}F \mid F & (3,4)\\ F &\to (E) \mid a \mid b & (5,6,7)\end{aligned}$$

Reversed rightmost analysis
of (a)*b:
6 4 2 5 4 7 3



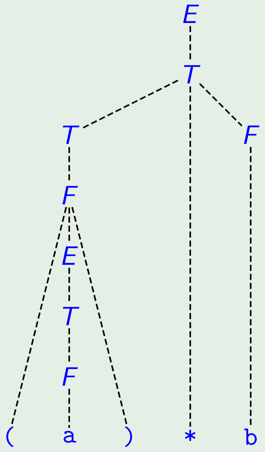
Example 8.10

Grammar for arithmetic expressions:

$$
\begin{array}{llll}
G_{AE}: & E \to E\texttt{+}T \mid T & & (1,2) \\
 & T \to T\texttt{*}F \mid F & & (3,4) \\
 & F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7)
\end{array}
$$

Reversed rightmost analysis of (a)*b:

6 4 2 5 4 7 3 2



Bottom-Up Parsing II

Approach:

1. Given $G \in CFG_\Sigma$, construct a nondeterministic bottom-up parsing automaton (NBA) which accepts $L(G)$ and which additionally computes corresponding (reversed) rightmost analyses
   - input alphabet: Σ
   - pushdown alphabet: X
   - output alphabet: $[p]$ (where $p := |P|$)
   - state set: omitted
   - transitions:

     shift: shifting input symbols onto the pushdown

     reduce: replacing the right-hand side of a production by its left-hand side (= inverse expansion steps)

Bottom-Up Parsing II

Approach:

1. Given $G \in CFG_\Sigma$, construct a nondeterministic bottom-up parsing automaton (NBA) which accepts $L(G)$ and which additionally computes corresponding (reversed) rightmost analyses
   - input alphabet: Σ
   - pushdown alphabet: X
   - output alphabet: $[p]$ (where $p := |P|$)
   - state set: omitted
   - transitions:

     shift: shifting input symbols onto the pushdown

     reduce: replacing the right-hand side of a production by its left-hand side (= inverse expansion steps)
2. Remove nondeterminism by allowing lookahead on the input: $G \in LR(k)$ iff $L(G)$ recognizable by deterministic bottom-up parsing automaton with lookahead of k symbols

Outline

Nondeterministic Bottom-Up Automaton I

Definition 8.11 (Nondeterministic bottom-up parsing automaton)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$. The nondeterministic bottom-up parsing automaton of G, $\mathrm{NBA}(G)$, is defined by the following components.

- Input alphabet: Σ
- Pushdown alphabet: X
- Output alphabet: $[p]$
- Configurations: $\Sigma^* \times X^* \times [p]^*$ (top of pushdown to the right)
- Transitions for $w \in \Sigma^*$, $\alpha \in X^*$, and $z \in [p]^*$:
  shifting steps: $(aw, \alpha, z) \vdash (w, \alpha a, z)$ if $a \in \Sigma$
  reduction steps: $(w, \alpha\beta, z) \vdash (w, \alpha A, zi)$ if $\pi_i = A \to \beta$
- Initial configuration for $w \in \Sigma^*$: $(w, \varepsilon, \varepsilon)$
- Final configurations: $\{\varepsilon\} \times \{S\} \times [p]^*$

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{aligned}
G_{AE}: \; E &\rightarrow E\texttt{+}T \mid T & (1,2)\\
T &\rightarrow T\texttt{*}F \mid F & (3,4)\\
F &\rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (5,6,7)
\end{aligned}
$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{aligned}
G_{AE}: E &\to E\texttt{+}T \mid T && (1,2)\\
T &\to T\texttt{*}F \mid F && (3,4)\\
F &\to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} && (5,6,7)
\end{aligned}
$$

Bottom-up parsing of `(a)*b`:

$(\texttt{(a)*b}, \varepsilon \quad, \varepsilon \quad)$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{aligned}
G_{AE}: \; E &\rightarrow E\texttt{+}T \mid T && (1,2)\\
T &\rightarrow T\texttt{*}F \mid F && (3,4)\\
F &\rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} && (5,6,7)
\end{aligned}
$$

Bottom-up parsing of `(a)*b`:

$$
\begin{aligned}
& (\texttt{(a)*b}, \varepsilon, \varepsilon)\\
\vdash\; & (\texttt{a)*b}, \texttt{(}, \varepsilon)
\end{aligned}
$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for
arithmetic expressions
(cf. Example 8.10):

$$
\begin{array}{llll}
G_{AE}: & E \to E\texttt{+}T \mid T & & (1,2) \\
& T \to T\texttt{*}F \mid F & & (3,4) \\
& F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7)
\end{array}
$$

Bottom-up parsing of `(a)*b`:

$$
\begin{array}{ll}
& (\texttt{(a)*b}, \varepsilon, \varepsilon) \\
\vdash & (\texttt{a)*b}, \texttt{(}, \varepsilon) \\
\vdash & (\texttt{)*b}, \texttt{(a}, \varepsilon)
\end{array}
$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{aligned}
G_{AE}: \; E &\rightarrow E\texttt{+}T \mid T & (1,2)\\
T &\rightarrow T\texttt{*}F \mid F & (3,4)\\
F &\rightarrow \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (5,6,7)
\end{aligned}
$$

Bottom-up parsing of `(a)*b`:

$$
\begin{aligned}
& (\texttt{(a)*b}, \varepsilon, \varepsilon)\\
\vdash\; & (\texttt{a)*b}, \texttt{(}, \varepsilon)\\
\vdash\; & (\texttt{)*b}, \texttt{(a}, \varepsilon)\\
\vdash\; & (\texttt{)*b}, \texttt{(}F, 6)
\end{aligned}
$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{aligned}
G_{AE}:\ & E \to E\texttt{+}T \mid T && (1,2)\\
& T \to T\texttt{*}F \mid F && (3,4)\\
& F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} && (5,6,7)
\end{aligned}
$$

Bottom-up parsing of `(a)*b`:

$$
\begin{aligned}
& (\texttt{(a)*b}, \varepsilon, \varepsilon)\\
\vdash\ & (\texttt{a)*b}, \texttt{(}, \varepsilon)\\
\vdash\ & (\texttt{)*b}, \texttt{(a}, \varepsilon)\\
\vdash\ & (\texttt{)*b}, \texttt{(}F, 6)\\
\vdash\ & (\texttt{)*b}, \texttt{(}T, 64)
\end{aligned}
$$

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$G_{AE}: \begin{array}{llll} E & \rightarrow & E\texttt{+}T \mid T & (1,2) \\ T & \rightarrow & T\texttt{*}F \mid F & (3,4) \\ F & \rightarrow & \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (5,6,7) \end{array}$$

Bottom-up parsing of $\texttt{(a)*b}$:

$$\begin{array}{ll} & (\texttt{(a)*b}, \varepsilon, \varepsilon) \\ \vdash & (\texttt{a)*b}, \texttt{(}, \varepsilon) \\ \vdash & (\texttt{)*b}, \texttt{(a}, \varepsilon) \\ \vdash & (\texttt{)*b}, \texttt{(}F, 6) \\ \vdash & (\texttt{)*b}, \texttt{(}T, 64) \\ \vdash & (\texttt{)*b}, \texttt{(}E, 642) \end{array}$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{aligned}
G_{AE}: \; & E \to E{+}T \mid T && (1,2)\\
& T \to T{*}F \mid F && (3,4)\\
& F \to (E) \mid \mathtt{a} \mid \mathtt{b} && (5,6,7)
\end{aligned}
$$

Bottom-up parsing of $\mathtt{(a)*b}$:

$$
\begin{aligned}
& (\mathtt{(a)*b}, \varepsilon, \varepsilon)\\
\vdash\; & (\mathtt{a)*b}, \mathtt{(}, \varepsilon)\\
\vdash\; & (\mathtt{)*b}, \mathtt{(a}, \varepsilon)\\
\vdash\; & (\mathtt{)*b}, \mathtt{(}F, 6)\\
\vdash\; & (\mathtt{)*b}, \mathtt{(}T, 64)\\
\vdash\; & (\mathtt{)*b}, \mathtt{(}E, 642)\\
\vdash\; & (\mathtt{*b}, \mathtt{(}E\mathtt{)}, 642)
\end{aligned}
$$

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{array}{llll}
G_{AE}: & E \to E{+}T \mid T & (1,2) \\
 & T \to T{*}F \mid F & (3,4) \\
 & F \to (E) \mid \mathtt{a} \mid \mathtt{b} & (5,6,7)
\end{array}
$$

Bottom-up parsing of `(a)*b`:

$$
\begin{array}{llll}
 & (\mathtt{(a)*b}, & \varepsilon & , \varepsilon) \\
\vdash & (\mathtt{a)*b}, & \mathtt{(} & , \varepsilon) \\
\vdash & (\mathtt{)*b}, & \mathtt{(a} & , \varepsilon) \\
\vdash & (\mathtt{)*b}, & \mathtt{(}F & , 6) \\
\vdash & (\mathtt{)*b}, & \mathtt{(}T & , 64) \\
\vdash & (\mathtt{)*b}, & \mathtt{(}E & , 642) \\
\vdash & (\mathtt{*b}, & \mathtt{(}E\mathtt{)} & , 642) \\
\vdash & (\mathtt{*b}, & F & , 6425)
\end{array}
$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{aligned}
G_{AE}: \; & E \to E{+}T \mid T & (1,2)\\
& T \to T{*}F \mid F & (3,4)\\
& F \to (E) \mid \mathtt{a} \mid \mathtt{b} & (5,6,7)
\end{aligned}
$$

Bottom-up parsing of (a)*b:

$$
\begin{aligned}
& (\mathtt{(a)*b}, \varepsilon, \varepsilon)\\
\vdash\; & (\mathtt{a)*b}, \mathtt{(}, \varepsilon)\\
\vdash\; & (\mathtt{)*b}, \mathtt{(a}, \varepsilon)\\
\vdash\; & (\mathtt{)*b}, \mathtt{(}F, 6)\\
\vdash\; & (\mathtt{)*b}, \mathtt{(}T, 64)\\
\vdash\; & (\mathtt{)*b}, \mathtt{(}E, 642)\\
\vdash\; & (\mathtt{*b}, \mathtt{(}E\mathtt{)}, 642)\\
\vdash\; & (\mathtt{*b}, F, 6425)\\
\vdash\; & (\mathtt{*b}, T, 64254)
\end{aligned}
$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{aligned}
G_{AE}: \; E &\to E\texttt{+}T \mid T && (1,2)\\
T &\to T\texttt{*}F \mid F && (3,4)\\
F &\to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} && (5,6,7)
\end{aligned}
$$

Bottom-up parsing of `(a)*b`:

$$
\begin{aligned}
& (\texttt{(a)*b}, \varepsilon, \varepsilon)\\
\vdash\; & (\texttt{a)*b}, \texttt{(}, \varepsilon)\\
\vdash\; & (\texttt{)*b}, \texttt{(a}, \varepsilon)\\
\vdash\; & (\texttt{)*b}, \texttt{(}F, 6)\\
\vdash\; & (\texttt{)*b}, \texttt{(}T, 64)\\
\vdash\; & (\texttt{)*b}, \texttt{(}E, 642)\\
\vdash\; & (\texttt{*b}, \texttt{(}E\texttt{)}, 642)\\
\vdash\; & (\texttt{*b}, F, 6425)\\
\vdash\; & (\texttt{*b}, T, 64254)\\
\vdash\; & (\texttt{b}, T\texttt{*}, 64254)
\end{aligned}
$$

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{array}{llll}
G_{AE}: & E \to E\texttt{+}T \mid T & (1,2) \\
& T \to T\texttt{*}F \mid F & (3,4) \\
& F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (5,6,7)
\end{array}
$$

Bottom-up parsing of `(a)*b`:

$$
\begin{array}{lllll}
& (& \texttt{(a)*b}, & \varepsilon & , \varepsilon &) \\
\vdash & (& \texttt{a)*b}, & \texttt{(} & , \varepsilon &) \\
\vdash & (& \texttt{)*b}, & \texttt{(a} & , \varepsilon &) \\
\vdash & (& \texttt{)*b}, & \texttt{(}F & , 6 &) \\
\vdash & (& \texttt{)*b}, & \texttt{(}T & , 64 &) \\
\vdash & (& \texttt{)*b}, & \texttt{(}E & , 642 &) \\
\vdash & (& \texttt{*b}, & \texttt{(}E\texttt{)} & , 642 &) \\
\vdash & (& \texttt{*b}, & F & , 6425 &) \\
\vdash & (& \texttt{*b}, & T & , 64254 &) \\
\vdash & (& \texttt{b}, & T\texttt{*} & , 64254 &) \\
\vdash & (& \varepsilon, & T\texttt{*b} & , 64254 &)
\end{array}
$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{aligned}
G_{AE}: \; & E \to E\texttt{+}T \mid T && (1,2) \\
& T \to T\texttt{*}F \mid F && (3,4) \\
& F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} && (5,6,7)
\end{aligned}
$$

Bottom-up parsing of `(a)*b`:

$$
\begin{array}{ll}
 & (\texttt{(a)*b},\ \varepsilon,\ \varepsilon) \\
\vdash & (\texttt{a)*b},\ \texttt{(},\ \varepsilon) \\
\vdash & (\texttt{)*b},\ \texttt{(a},\ \varepsilon) \\
\vdash & (\texttt{)*b},\ \texttt{(}F,\ 6) \\
\vdash & (\texttt{)*b},\ \texttt{(}T,\ 64) \\
\vdash & (\texttt{)*b},\ \texttt{(}E,\ 642) \\
\vdash & (\texttt{*b},\ \texttt{(}E\texttt{)},\ 642) \\
\vdash & (\texttt{*b},\ F,\ 6425) \\
\vdash & (\texttt{*b},\ T,\ 64254) \\
\vdash & (\texttt{b},\ T\texttt{*},\ 64254) \\
\vdash & (\varepsilon,\ T\texttt{*b},\ 64254) \\
\vdash & (\varepsilon,\ T\texttt{*}F,\ 642547)
\end{array}
$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{array}{llll}
G_{AE}: & E \to E\texttt{+}T \mid T & & (1,2) \\
& T \to T\texttt{*}F \mid F & & (3,4) \\
& F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7)
\end{array}
$$

Bottom-up parsing of $\texttt{(a)*b}$:

$$
\begin{array}{rlll}
& (\texttt{(a)*b}, & \varepsilon & , \varepsilon) \\
\vdash & (\texttt{a)*b}, & \texttt{(} & , \varepsilon) \\
\vdash & (\texttt{)*b}, & \texttt{(a} & , \varepsilon) \\
\vdash & (\texttt{)*b}, & \texttt{(}F & , 6) \\
\vdash & (\texttt{)*b}, & \texttt{(}T & , 64) \\
\vdash & (\texttt{)*b}, & \texttt{(}E & , 642) \\
\vdash & (\texttt{*b}, & \texttt{(}E\texttt{)} & , 642) \\
\vdash & (\texttt{*b}, & F & , 6425) \\
\vdash & (\texttt{*b}, & T & , 64254) \\
\vdash & (\texttt{b}, & T\texttt{*} & , 64254) \\
\vdash & (\varepsilon, & T\texttt{*b} & , 64254) \\
\vdash & (\varepsilon, & T\texttt{*}F & , 642547) \\
\vdash & (\varepsilon, & T & , 6425473)
\end{array}
$$

Nondeterministic Bottom-Up Automaton II

Example 8.12

Grammar for arithmetic expressions (cf. Example 8.10):

$$
\begin{array}{llll}
G_{AE}: & E \to E\texttt{+}T \mid T & & (1,2) \\
& T \to T\texttt{*}F \mid F & & (3,4) \\
& F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & & (5,6,7)
\end{array}
$$

Bottom-up parsing of (a)*b:

$$
\begin{array}{rl}
& (\texttt{(a)*b}, \varepsilon, \varepsilon) \\
\vdash & (\texttt{a)*b}, \texttt{(}, \varepsilon) \\
\vdash & (\texttt{)*b}, \texttt{(a}, \varepsilon) \\
\vdash & (\texttt{)*b}, \texttt{(}F, 6) \\
\vdash & (\texttt{)*b}, \texttt{(}T, 64) \\
\vdash & (\texttt{)*b}, \texttt{(}E, 642) \\
\vdash & (\texttt{*b}, \texttt{(}E\texttt{)}, 642) \\
\vdash & (\texttt{*b}, F, 6425) \\
\vdash & (\texttt{*b}, T, 64254) \\
\vdash & (\texttt{b}, T\texttt{*}, 64254) \\
\vdash & (\varepsilon, T\texttt{*b}, 64254) \\
\vdash & (\varepsilon, T\texttt{*}F, 642547) \\
\vdash & (\varepsilon, T, 6425473) \\
\vdash & (\varepsilon, E, 64254732)
\end{array}
$$

Correctness of $\mathrm{NBA}(G)$

Theorem 8.13 (Correctness of $\mathrm{NBA}(G)$)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ and $\mathrm{NBA}(G)$ as before. Then, for every $w \in \Sigma^*$ and $z \in [p]^*$,

$$(w, \varepsilon, \varepsilon) \vdash^* (\varepsilon, S, z) \quad \textit{iff} \quad \overleftarrow{z} \textit{ is a rightmost analysis of } w$$

Correctness of $\mathrm{NBA}(G)$

Theorem 8.13 (Correctness of $\mathrm{NBA}(G)$)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ *and* $\mathrm{NBA}(G)$ *as before. Then, for every* $w \in \Sigma^*$ *and* $z \in [p]^*$,

$$(w, \varepsilon, \varepsilon) \vdash^* (\varepsilon, S, z) \quad \textit{iff} \quad \overleftarrow{z} \text{ is a rightmost analysis of } w$$

Proof.

similar to the top-down case (Theorem 6.1) □

Nondeterminisn in $\mathrm{NBA}(G)$

Observation: $\mathrm{NBA}(G)$ is generally nondeterministic

- Shift or reduce? Example:

$$(bw, \alpha a, z) \vdash \begin{cases} (w, \alpha ab, z) \\ (bw, \alpha A, zi) \end{cases} \text{ if } \pi_i = A \to a$$

Nondeterminisn in $\mathrm{NBA}(G)$

Observation: $\mathrm{NBA}(G)$ is generally nondeterministic

- Shift or reduce? Example:

$$(bw, \alpha a, z) \vdash \begin{cases} (w, \alpha ab, z) \\ (bw, \alpha A, zi) \text{ if } \pi_i = A \to a \end{cases}$$

- If reduce: which "handle" β? Example:

$$(w, \alpha ab, z) \vdash \begin{cases} (w, \alpha A, zi) \\ (w, \alpha aB, zj) \end{cases} \text{ if } \pi_i = A \to ab \text{ and } \pi_j = B \to b$$

Nondeterminisn in $\mathrm{NBA}(G)$

Observation: $\mathrm{NBA}(G)$ is generally nondeterministic

- Shift or reduce? Example:

$$(bw, \alpha a, z) \vdash \begin{cases} (w, \alpha ab, z) \\ (bw, \alpha A, zi) \end{cases} \text{ if } \pi_i = A \to a$$

- If reduce: which "handle" β? Example:

$$(w, \alpha ab, z) \vdash \begin{cases} (w, \alpha A, zi) \\ (w, \alpha aB, zj) \end{cases} \text{ if } \pi_i = A \to ab \text{ and } \pi_j = B \to b$$

- If reduce β: which left-hand side A? Example:

$$(w, \alpha a, z) \vdash \begin{cases} (w, \alpha A, zi) \\ (w, \alpha B, zj) \end{cases} \text{ if } \pi_i = A \to a \text{ and } \pi_j = B \to a$$

Nondeterminisn in $\mathrm{NBA}(G)$

Observation: $\mathrm{NBA}(G)$ is generally nondeterministic

- Shift or reduce? Example:

$$(bw, \alpha a, z) \vdash \begin{cases} (w, \alpha ab, z) \\ (bw, \alpha A, zi) \end{cases} \text{ if } \pi_i = A \to a$$

- If reduce: which "handle" β? Example:

$$(w, \alpha ab, z) \vdash \begin{cases} (w, \alpha A, zi) \\ (w, \alpha aB, zj) \end{cases} \text{ if } \pi_i = A \to ab \text{ and } \pi_j = B \to b$$

- If reduce β: which left-hand side A? Example:

$$(w, \alpha a, z) \vdash \begin{cases} (w, \alpha A, zi) \\ (w, \alpha B, zj) \end{cases} \text{ if } \pi_i = A \to a \text{ and } \pi_j = B \to a$$

- When to terminate parsing? Example:

$$\underbrace{(\varepsilon, S, z)}_{\text{final}} \vdash (\varepsilon, A, zi) \text{ if } \pi_i = A \to S$$