



Compiler Construction

Lecture 8: Syntax Analysis IV ($LL(1)$) and Bottom-Up Parsing)

Winter Semester 2018/19

Thomas Noll

Software Modeling and Verification Group

RWTH Aachen University

<https://moves.rwth-aachen.de/teaching/ws-1819/cc/>

Recap: $LL(1)$ Parsing

Outline of Lecture 8

Recap: $LL(1)$ Parsing

Transformation to $LL(1)$

The Complexity of $LL(1)$ Parsing

Recursive-Descent Parsing

Bottom-Up Parsing

Nondeterministic Bottom-Up Parsing

Recap: $LL(1)$ Parsing

Characterisation of $LL(1)$

Theorem (Characterisation of $LL(1)$)

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

$$\text{la}(A \rightarrow \beta) \cap \text{la}(A \rightarrow \gamma) = \emptyset.$$

Proof.

later □

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

Recap: $LL(1)$ Parsing

The Deterministic Top-Down Automaton

Definition (Deterministic top-down parsing automaton)

Let $G = \langle N, \Sigma, P, S \rangle \in LL(1)$. The **deterministic top-down parsing automaton** of G , $DTA(G)$, is defined by the following components.

- **Input alphabet** Σ , **pushdown alphabet** X , **output alphabet** $[p]$
- **Configurations** $\Sigma^* \times X^* \times [p]^*$, **initial configuration** (w, S, ε) , **final configurations** $\{\varepsilon\} \times \{\varepsilon\} \times [p]^*$ (as $NTA(G)$)
- **Action function** $\text{act} : \Sigma_\varepsilon \times X_\varepsilon \rightarrow \{(\alpha, i) \mid \pi_i = A \rightarrow \alpha\} \cup \{\text{pop}, \text{accept}, \text{error}\}$
with $\text{act}(x, A) := (\alpha, i)$ if $\pi_i = A \rightarrow \alpha$ and $x \in \text{la}(\pi_i)$
 $\text{act}(a, a) := \text{pop}$
 $\text{act}(\varepsilon, \varepsilon) := \text{accept}$
 $\text{act}(x, y) := \text{error}$ otherwise
- **Transitions** for $x \in \Sigma_\varepsilon$, $w \in \Sigma^*$ (with $x \neq \varepsilon$ if $w \neq \varepsilon$), $Y \in X$, $\beta \in X^*$, and $z \in [p]^*$:

$$(xw, Y\beta, z) \vdash \begin{cases} (xw, \alpha\beta, zi) & \text{if } \text{act}(x, Y) = (\alpha, i) \\ (w, \beta, z) & \text{if } \text{act}(x, Y) = \text{pop} \end{cases}$$

Transformation to $LL(1)$

Outline of Lecture 8

Recap: $LL(1)$ Parsing

Transformation to $LL(1)$

The Complexity of $LL(1)$ Parsing

Recursive-Descent Parsing

Bottom-Up Parsing

Nondeterministic Bottom-Up Parsing

Transformation to $LL(1)$

Transformation to $LL(1)$

Goal: transformation of $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma} \setminus LL(1)$

(i.e., $la(A \rightarrow \beta) \cap la(A \rightarrow \gamma) \neq \emptyset$ for some $A \rightarrow \beta \mid \gamma \in P$) to $G' \in LL(1)$ with $L(G') = L(G)$

Transformation to $LL(1)$

Transformation to $LL(1)$

Goal: transformation of $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma} \setminus LL(1)$

(i.e., $la(A \rightarrow \beta) \cap la(A \rightarrow \gamma) \neq \emptyset$ for some $A \rightarrow \beta \mid \gamma \in P$) to $G' \in LL(1)$ with $L(G') = L(G)$

Problems

- Transformations generally **preserve the semantics** (= generated language) of CFGs but **not the syntactic structure** of words (different syntax trees).

Transformation to $LL(1)$

Transformation to $LL(1)$

Goal: transformation of $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma} \setminus LL(1)$

(i.e., $la(A \rightarrow \beta) \cap la(A \rightarrow \gamma) \neq \emptyset$ for some $A \rightarrow \beta \mid \gamma \in P$) to $G' \in LL(1)$ with $L(G') = L(G)$

Problems

- 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. Example of $L \notin \bigcup_{k \in \mathbb{N}} L(LL(k))$:
 $\{a^n 0 b^n \mid n \geq 1\} \cup \{a^n 1 b^{2^n} \mid n \geq 1\}$ (unbounded number of a 's required for decision)

Transformation to $LL(1)$

Transformation to $LL(1)$

Goal: transformation of $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma} \setminus LL(1)$

(i.e., $la(A \rightarrow \beta) \cap la(A \rightarrow \gamma) \neq \emptyset$ for some $A \rightarrow \beta \mid \gamma \in P$) to $G' \in LL(1)$ with $L(G') = L(G)$

Problems

- 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. Example of $L \notin \bigcup_{k \in \mathbb{N}} L(LL(k))$:
 $\{a^n 0 b^n \mid n \geq 1\} \cup \{a^n 1 b^{2^n} \mid n \geq 1\}$ (unbounded number of a 's required for decision)
- Even worse: given an arbitrary CFG it is **undecidable** whether there is $G \in LL(k)$ (for some k) which generates the same language (cf. D.J. Rosenkrantz, R.E. Stearns: *Properties of Deterministic Top Down Grammars*, Information and Control 17:226–256, 1970)

Transformation to $LL(1)$

Transformation to $LL(1)$

Goal: transformation of $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma} \setminus LL(1)$

(i.e., $la(A \rightarrow \beta) \cap la(A \rightarrow \gamma) \neq \emptyset$ for some $A \rightarrow \beta \mid \gamma \in P$) to $G' \in LL(1)$ with $L(G') = L(G)$

Problems

- 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. Example of $L \notin \bigcup_{k \in \mathbb{N}} L(LL(k))$:
 $\{a^n 0 b^n \mid n \geq 1\} \cup \{a^n 1 b^{2n} \mid n \geq 1\}$ (unbounded number of a 's required for decision)
- Even worse: given an arbitrary CFG it is **undecidable** whether there is $G \in LL(k)$ (for some k) which generates the same language (cf. D.J. Rosenkrantz, R.E. Stearns: *Properties of Deterministic Top Down Grammars*, Information and Control 17:226–256, 1970)

Heuristics for transforming G into $G' \in LL(1)$

1. Elimination of left recursion
2. Left factorization

(used in parser-generating systems such as ANTLR)

Transformation to $LL(1)$

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 $A \Rightarrow^+ A\alpha$.

Transformation to $LL(1)$

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 $A \Rightarrow_i^+ A\beta$.*

Transformation to $LL(1)$

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 $A \Rightarrow_i^+ A\beta$.*

Example 8.3

The grammar (cf. Example 5.11) $G_{AE} : \begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow (E) \mid a \mid b \end{array}$

is left recursive, and in Example 7.3 it was shown that $G_{AE} \notin LL(1)$

Left Recursion II

Lemma 8.4

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

Transformation to $LL(1)$

Left Recursion II

Lemma 8.4

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

Proof.

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

Transformation to $LL(1)$

Left Recursion II

Lemma 8.4

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

Proof.

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

The corresponding computation of $DTA(G)$ (Definition 7.5) starts with $(vw, S, \varepsilon) \vdash^* (w, A\alpha, \dots) \vdash^+ (w, A\beta\alpha, \dots)$.

Transformation to $LL(1)$

Left Recursion II

Lemma 8.4

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

Proof.

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

The corresponding computation of $DTA(G)$ (Definition 7.5) starts with $(vw, S, \varepsilon) \vdash^* (w, A\alpha, \dots) \vdash^+ (w, A\beta\alpha, \dots)$.

But in the last configuration the behaviour of $DTA(G)$ is determined by the same input ($\text{fi}(w)$) and stack symbol (A). Thus it enters an **infinite loop** of the form

$$(w, A\alpha, \dots) \vdash^+ (w, A\beta\alpha, \dots) \vdash^+ (w, A\beta\beta\alpha, \dots) \vdash^+ \dots$$

and will never recognize w . ⚡

□

Eliminating Direct Left Recursion

Direct left recursion occurs in productions of the form

$$A \rightarrow A\alpha_1 \mid \dots \mid A\alpha_m \mid \beta_1 \mid \dots \mid \beta_n \quad \text{where } \alpha_i \neq \varepsilon \text{ and } \beta_j \neq A\dots$$

Transformation to $LL(1)$

Eliminating Direct Left Recursion

Direct left recursion occurs in productions of the form

$$A \rightarrow A\alpha_1 \mid \dots \mid A\alpha_m \mid \beta_1 \mid \dots \mid \beta_n \quad \text{where } \alpha_i \neq \varepsilon \text{ and } \beta_j \neq A\dots$$

Transformation: replacement by **right recursion**

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

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

Transformation to $LL(1)$

Eliminating Direct Left Recursion

Direct left recursion occurs in productions of the form

$$A \rightarrow A\alpha_1 \mid \dots \mid A\alpha_m \mid \beta_1 \mid \dots \mid \beta_n \quad \text{where } \alpha_i \neq \varepsilon \text{ and } \beta_j \neq A\dots$$

Transformation: replacement by **right recursion**

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

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

Example 8.5

$$\begin{array}{l} G_{AE} : E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow (E) \mid a \mid b \end{array} \quad \text{is transformed into} \quad \begin{array}{l} G'_{AE} : E \rightarrow TE' \\ E' \rightarrow +TE' \mid \varepsilon \\ T \rightarrow FT' \\ T' \rightarrow *FT' \mid \varepsilon \\ F \rightarrow (E) \mid a \mid b \end{array}$$

with $G'_{AE} \in LL(1)$ (see Example 7.4).

Transformation to $LL(1)$

Eliminating Indirect Left Recursion I

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

$$A \rightarrow A_1\alpha_1 \quad A_1 \rightarrow A_2\alpha_2 \quad \dots \quad A_{k-1} \rightarrow A_k\alpha_k \quad A_k \rightarrow A\beta$$

Transformation to $LL(1)$

Eliminating Indirect Left Recursion I

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

$$A \rightarrow A_1\alpha_1 \quad A_1 \rightarrow A_2\alpha_2 \quad \dots \quad A_{k-1} \rightarrow A_k\alpha_k \quad A_k \rightarrow A\beta$$

Systematic elimination of left recursion from a grammar without cycles ($A \Rightarrow^+ A$) and ε -productions ($A \rightarrow \varepsilon$) (which can both be eliminated systematically):

Algorithm 8.6 (Elimination of left recursion)

Input: $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ with $N = \{B_1, \dots, B_n\}$

Transformation to $LL(1)$

Eliminating Indirect Left Recursion I

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

$$A \rightarrow A_1\alpha_1 \quad A_1 \rightarrow A_2\alpha_2 \quad \dots \quad A_{k-1} \rightarrow A_k\alpha_k \quad A_k \rightarrow A\beta$$

Systematic elimination of left recursion from a grammar without cycles ($A \Rightarrow^+ A$) and ε -productions ($A \rightarrow \varepsilon$) (which can both be eliminated systematically):

Algorithm 8.6 (Elimination of left recursion)

Input: $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ with $N = \{B_1, \dots, B_n\}$

Procedure: **for** $i := 1$ **to** n **do**

for $j := 1$ **to** $i - 1$ **do** % remove references to “previous” nonterminals

 replace each $B_i \rightarrow B_j\gamma$ by $B_i \rightarrow \delta_1\gamma \mid \dots \mid \delta_k\gamma$,

 where $B_j \rightarrow \delta_1 \mid \dots \mid \delta_k$ are all current B_j -productions

end

 eliminate direct left recursion among B_i -productions

end

Transformation to $LL(1)$

Eliminating Indirect Left Recursion I

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

$$A \rightarrow A_1\alpha_1 \quad A_1 \rightarrow A_2\alpha_2 \quad \dots \quad A_{k-1} \rightarrow A_k\alpha_k \quad A_k \rightarrow A\beta$$

Systematic elimination of left recursion from a grammar without cycles ($A \Rightarrow^+ A$) and ε -productions ($A \rightarrow \varepsilon$) (which can both be eliminated systematically):

Algorithm 8.6 (Elimination of left recursion)

Input: $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ with $N = \{B_1, \dots, B_n\}$

Procedure: **for** $i := 1$ **to** n **do**

for $j := 1$ **to** $i - 1$ **do** % remove references to “previous” nonterminals

 replace each $B_i \rightarrow B_j\gamma$ by $B_i \rightarrow \delta_1\gamma \mid \dots \mid \delta_k\gamma$,

 where $B_j \rightarrow \delta_1 \mid \dots \mid \delta_k$ are all current B_j -productions

end

 eliminate direct left recursion among B_i -productions

end

Output: equivalent grammar G' without left recursion (possibly with ε -productions)

Eliminating Indirect Left Recursion II

Example 8.7

1. Original grammar ($n = 2, B_1 = S, B_2 = A$):
- $$G : S \rightarrow Sa \mid Ab \mid c$$
- $$A \rightarrow Ad \mid Se \mid f$$

Transformation to $LL(1)$

Eliminating Indirect Left Recursion II

Example 8.7

1. Original grammar ($n = 2, B_1 = S, B_2 = A$):
$$G : S \rightarrow Sa \mid Ab \mid c$$
$$A \rightarrow Ad \mid Se \mid f$$
2. $i = 1$:
After B_1 -substitution (no $j < i$):
$$S \rightarrow Sa \mid Ab \mid c$$
$$A \rightarrow Ad \mid Se \mid f$$

After removal of direct left B_1 -recursion:
$$S \rightarrow AbS' \mid cS'$$
$$S' \rightarrow aS' \mid \varepsilon$$
$$A \rightarrow Ad \mid Se \mid f$$

Transformation to $LL(1)$

Eliminating Indirect Left Recursion II

Example 8.7

1. Original grammar ($n = 2, B_1 = S, B_2 = A$):
$$G : S \rightarrow Sa \mid Ab \mid c$$
$$A \rightarrow Ad \mid Se \mid f$$
2. $i = 1$:
After B_1 -substitution (no $j < i$):
$$S \rightarrow Sa \mid Ab \mid c$$
$$A \rightarrow Ad \mid Se \mid f$$

After removal of direct left B_1 -recursion:
$$S \rightarrow AbS' \mid cS'$$
$$S' \rightarrow aS' \mid \varepsilon$$
$$A \rightarrow Ad \mid Se \mid f$$
3. $i = 2$:
After B_2 -substitution ($j = 1$):
$$S \rightarrow AbS' \mid cS'$$
$$S' \rightarrow aS' \mid \varepsilon$$
$$A \rightarrow Ad \mid AbS'e \mid cS'e \mid f$$

After removal of direct left B_2 -recursion:
$$G' : S \rightarrow AbS' \mid cS'$$
$$S' \rightarrow aS' \mid \varepsilon$$
$$A \rightarrow cS'eA' \mid fA'$$
$$A' \rightarrow dA' \mid bS'eA' \mid \varepsilon$$

Transformation to $LL(1)$

Eliminating Indirect Left Recursion II

Example 8.7

1. Original grammar ($n = 2, B_1 = S, B_2 = A$):
 $G : S \rightarrow Sa \mid Ab \mid c$
 $A \rightarrow Ad \mid Se \mid f$
2. $i = 1$:
After B_1 -substitution (no $j < i$):
 $S \rightarrow Sa \mid Ab \mid c$
 $A \rightarrow Ad \mid Se \mid f$
After removal of direct left B_1 -recursion:
 $S \rightarrow AbS' \mid cS'$
 $S' \rightarrow aS' \mid \varepsilon$
 $A \rightarrow Ad \mid Se \mid f$
3. $i = 2$:
After B_2 -substitution ($j = 1$):
 $S \rightarrow AbS' \mid cS'$
 $S' \rightarrow aS' \mid \varepsilon$
 $A \rightarrow Ad \mid AbS'e \mid cS'e \mid f$
After removal of direct left B_2 -recursion:
 $G' : S \rightarrow AbS' \mid cS'$
 $S' \rightarrow aS' \mid \varepsilon$
 $A \rightarrow cS'eA' \mid fA'$
 $A' \rightarrow dA' \mid bS'eA' \mid \varepsilon$

(Observation: $G' \notin LL(1)$ since $\text{la}(S \rightarrow AbS') \cap \text{la}(S \rightarrow cS') = \{c\} \neq \emptyset$)

Transformation to $LL(1)$

Eliminating Indirect Left Recursion II

Example 8.7

1. Original grammar ($n = 2, B_1 = S, B_2 = A$):
 $G : S \rightarrow Sa \mid Ab \mid c$
 $A \rightarrow Ad \mid Se \mid f$
2. $i = 1$:
After B_1 -substitution (no $j < i$):
 $S \rightarrow Sa \mid Ab \mid c$
 $A \rightarrow Ad \mid Se \mid f$
After removal of direct left B_1 -recursion:
 $S \rightarrow AbS' \mid cS'$
 $S' \rightarrow aS' \mid \varepsilon$
 $A \rightarrow Ad \mid Se \mid f$
3. $i = 2$:
After B_2 -substitution ($j = 1$):
 $S \rightarrow AbS' \mid cS'$
 $S' \rightarrow aS' \mid \varepsilon$
 $A \rightarrow Ad \mid AbS'e \mid cS'e \mid f$
After removal of direct left B_2 -recursion:
 $G' : S \rightarrow AbS' \mid cS'$
 $S' \rightarrow aS' \mid \varepsilon$
 $A \rightarrow cS'eA' \mid fA'$
 $A' \rightarrow dA' \mid bS'eA' \mid \varepsilon$

(Observation: $G' \notin LL(1)$ since $\text{la}(S \rightarrow AbS') \cap \text{la}(S \rightarrow cS') = \{c\} \neq \emptyset$)

Remark: after applying Algorithm 8.6, **never** $G' \in LL(1)$ (cf. exercises)

Transformation to $LL(1)$

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”.

Transformation to $LL(1)$

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”.

Transformation: delaying the decision by **left factorization**

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

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

Transformation to $LL(1)$

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”.

Transformation: delaying the decision by **left factorization**

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

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

Example 8.8

Statement \rightarrow if *Condition* then *Statement* else *Statement* fi
| if *Condition* then *Statement* fi

is transformed into

Statement \rightarrow if *Condition* then *Statement* S'
 S' \rightarrow else *Statement* fi | fi

The Complexity of $LL(1)$ Parsing

Outline of Lecture 8

Recap: $LL(1)$ Parsing

Transformation to $LL(1)$

The Complexity of $LL(1)$ Parsing

Recursive-Descent Parsing

Bottom-Up Parsing

Nondeterministic Bottom-Up Parsing

The Complexity of $LL(1)$ Parsing

The Complexity of $LL(1)$ Parsing I

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

The Complexity of $LL(1)$ Parsing

The Complexity of $LL(1)$ Parsing I

- $LL(1)$ parsing has time (and hence space) **complexity** $\mathcal{O}(|w|)$ (for input word $w \in \Sigma^*$)
- Here: proof for **ε -free grammars** (i.e., without ε -productions)

The Complexity of $LL(1)$ Parsing

The Complexity of $LL(1)$ Parsing I

- $LL(1)$ parsing has time (and hence space) **complexity** $\mathcal{O}(|w|)$ (for input word $w \in \Sigma^*$)
- Here: proof for **ε -free grammars** (i.e., without ε -productions)
- General case: see O. Mayer: *Syntaxanalyse*, p. 211ff

The Complexity of $LL(1)$ Parsing

The Complexity of $LL(1)$ Parsing I

- $LL(1)$ parsing has time (and hence space) **complexity** $\mathcal{O}(|w|)$ (for input word $w \in \Sigma^*$)
- Here: proof for **ε -free grammars** (i.e., without ε -productions)
- General case: see O. Mayer: *Syntaxanalyse*, p. 211ff

Lemma 8.9

Let $G = \langle N, \Sigma, P, S \rangle \in LL(1)$ be without ε -productions. If

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

in $DTA(G)$, then

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

The Complexity of $LL(1)$ Parsing

The Complexity of $LL(1)$ Parsing II

Proof.

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

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



The Complexity of $LL(1)$ Parsing

The Complexity of $LL(1)$ Parsing II

Proof.

Let $(w, S, \varepsilon) \vdash^n (\varepsilon, \varepsilon, z)$ in $DTA(G)$. 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

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

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

□

The Complexity of $LL(1)$ Parsing

The Complexity of $LL(1)$ Parsing II

Proof.

Let $(w, S, \varepsilon) \vdash^n (\varepsilon, \varepsilon, z)$ in $DTA(G)$. 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

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

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

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

The Complexity of $LL(1)$ Parsing II

Proof.

Let $(w, S, \varepsilon) \vdash^n (\varepsilon, \varepsilon, z)$ in $DTA(G)$. 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

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

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

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



Recursive-Descent Parsing

Outline of Lecture 8

Recap: $LL(1)$ Parsing

Transformation to $LL(1)$

The Complexity of $LL(1)$ Parsing

Recursive-Descent Parsing

Bottom-Up Parsing

Nondeterministic Bottom-Up Parsing

Recursive-Descent Parsing

Recursive-Descent Parsing I

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

Recursive-Descent Parsing

Recursive-Descent Parsing I

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

Advantage: simple implementation

Recursive-Descent Parsing

Recursive-Descent Parsing I

Idea: avoid explicit use of pushdown store (as in $DTA(G)$) 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

Recursive-Descent Parsing I

Idea: avoid explicit use of pushdown store (as in $DTA(G)$) 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 8.10 (Arithmetic expressions; cf. Example 8.5)

```
proc main();
  token := next(); E()
proc E();   (*  $E \rightarrow T E'$  *)
  if token in {'(', 'a', 'b'} then print(1); T(); E'()
  else print(error); stop fi
proc E'();  (*  $E' \rightarrow + T E' \mid \varepsilon$  *)
  if token = '+' then print(2); token := next(); T(); E'()
  elsif token in {EOF, ')'} then print(3)
  else print(error); stop fi
proc T();   (*  $T \rightarrow F T'$  *)
  if token in {'(', 'a', 'b'} then print(4); F(); T'()
  else print(error); stop fi
proc T'();  (*  $T' \rightarrow * F T' \mid \varepsilon$  *)
  if token = '*' then print(5); token := next(); F(); T'()
  elsif token in {'+', EOF, ')'} then print(6)
  else print(error); stop fi
proc F();   (*  $F \rightarrow ( E ) \mid a \mid b$  *)
  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
```

Bottom-Up Parsing

Outline of Lecture 8

Recap: $LL(1)$ Parsing

Transformation to $LL(1)$

The Complexity of $LL(1)$ Parsing

Recursive-Descent Parsing

Bottom-Up Parsing

Nondeterministic Bottom-Up Parsing

Bottom-Up Parsing

Recap: Top-Down Parsing

Example 8.11

Grammar for
arithmetic expressions:

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

Bottom-Up Parsing

Recap: Top-Down Parsing

Example 8.11

E

Grammar for
arithmetic expressions:

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

Leftmost analysis of $(a)*b$:

(a) * b

Bottom-Up Parsing

Recap: Top-Down Parsing

Example 8.11

$$\begin{array}{c} E \\ | \\ T \end{array}$$

Grammar for
arithmetic expressions:

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

Leftmost analysis of $(a)*b$:

2

(a) * b

Bottom-Up Parsing

Recap: Top-Down Parsing

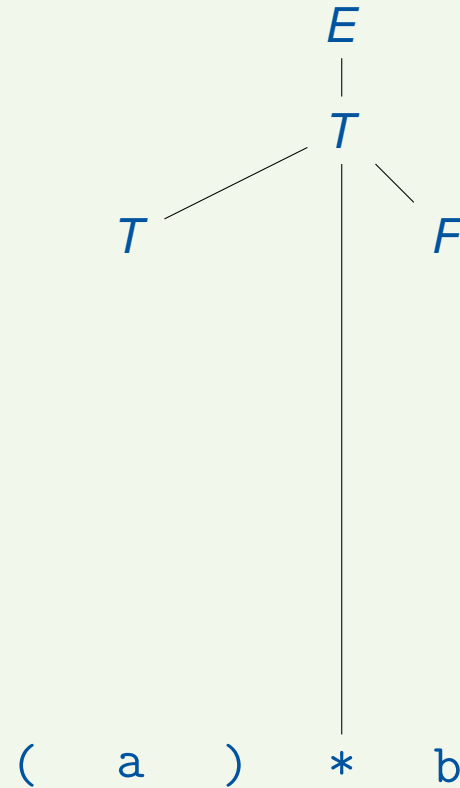
Example 8.11

Grammar for
arithmetic expressions:

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

Leftmost analysis of $(a)*b$:

2 3



Bottom-Up Parsing

Recap: Top-Down Parsing

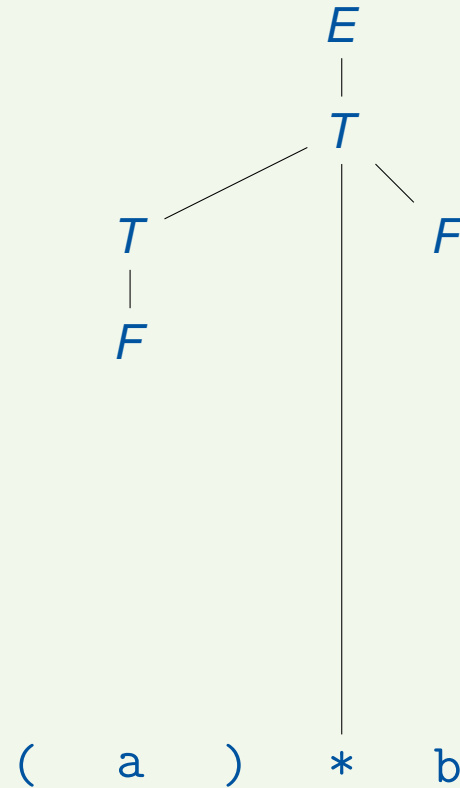
Example 8.11

Grammar for
arithmetic expressions:

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

Leftmost analysis of $(a)*b$:

2 3 4



Bottom-Up Parsing

Recap: Top-Down Parsing

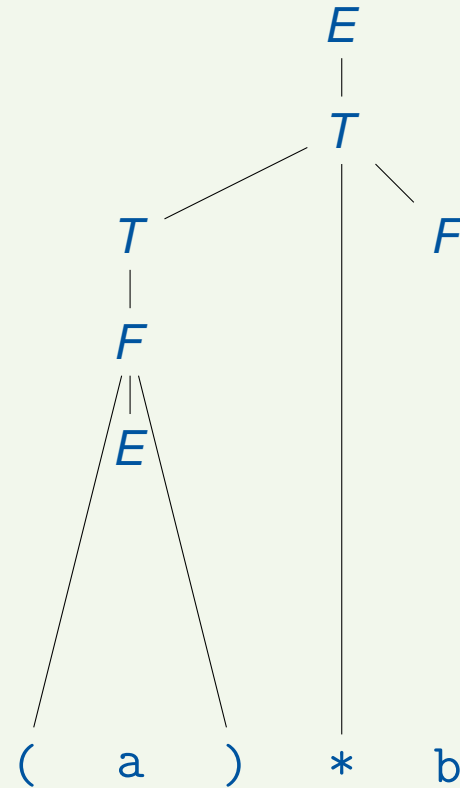
Example 8.11

Grammar for
arithmetic expressions:

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

Leftmost analysis of $(a)*b$:

2 3 4 5



Bottom-Up Parsing

Recap: Top-Down Parsing

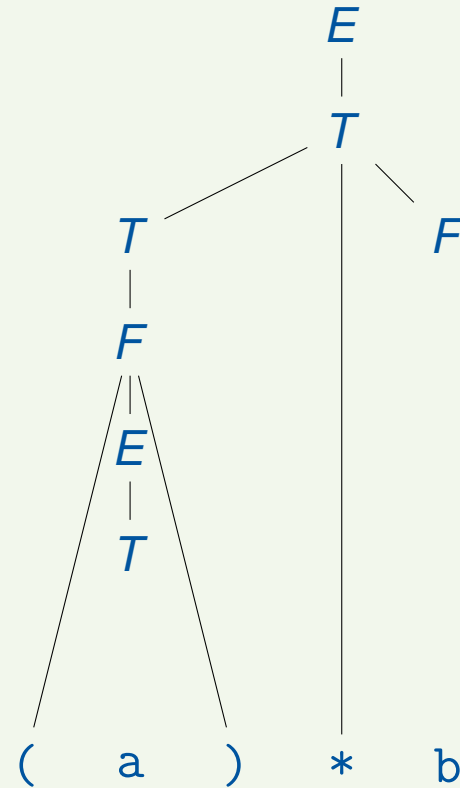
Example 8.11

Grammar for arithmetic expressions:

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

Leftmost analysis of $(a)*b$:

2 3 4 5 2



Bottom-Up Parsing

Recap: Top-Down Parsing

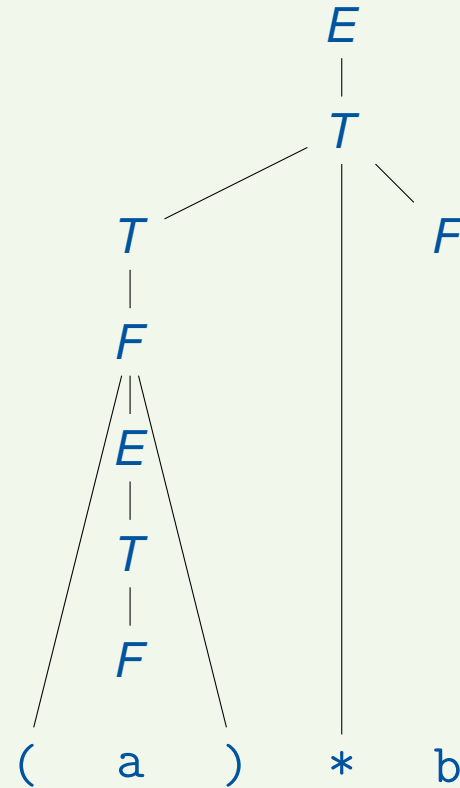
Example 8.11

Grammar for
arithmetic expressions:

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

Leftmost analysis of $(a)*b$:

2 3 4 5 2 4



Bottom-Up Parsing

Recap: Top-Down Parsing

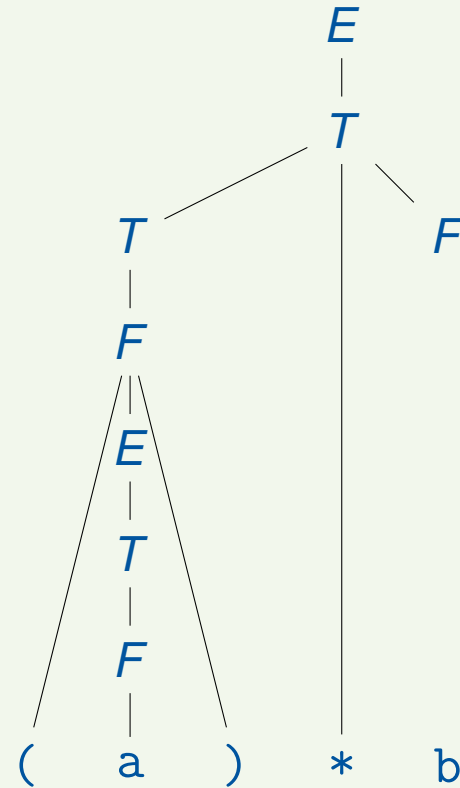
Example 8.11

Grammar for arithmetic expressions:

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

Leftmost analysis of $(a)*b$:

2 3 4 5 2 4 6



Bottom-Up Parsing

Recap: Top-Down Parsing

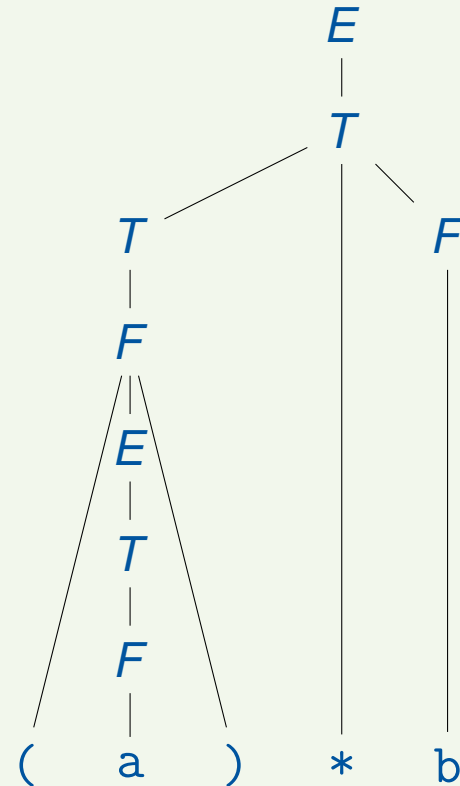
Example 8.11

Grammar for
arithmetic expressions:

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

Leftmost analysis of $(a)*b$:

2 3 4 5 2 4 6 7



Bottom-Up Parsing

Bottom-Up Parsing I

Example 8.12

Grammar for
arithmetic expressions:

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

Bottom-Up Parsing

Bottom-Up Parsing I

Example 8.12

Grammar for
arithmetic expressions:

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

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

(a) * b

Bottom-Up Parsing

Bottom-Up Parsing I

Example 8.12

Grammar for
arithmetic expressions:

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

Reversed rightmost analysis

of $(a)*b$:

6

F
|
(a) * b

Bottom-Up Parsing

Bottom-Up Parsing I

Example 8.12

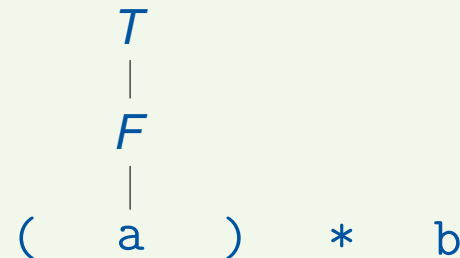
Grammar for
arithmetic expressions:

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

Reversed rightmost analysis

of $(a)*b$:

6 4



Bottom-Up Parsing

Bottom-Up Parsing I

Example 8.12

Grammar for
arithmetic expressions:

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

Reversed rightmost analysis

of $(a)*b$:

6 4 2



Bottom-Up Parsing

Bottom-Up Parsing I

Example 8.12

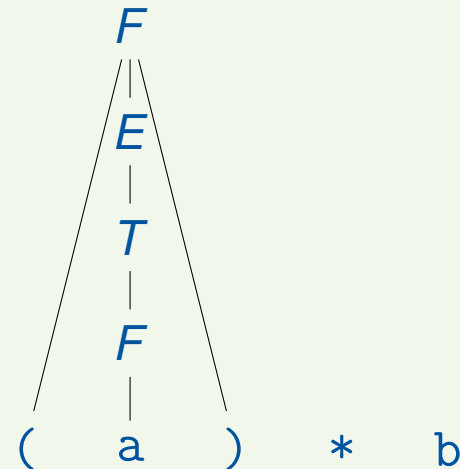
Grammar for
arithmetic expressions:

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

Reversed rightmost analysis

of $(a)*b$:

6 4 2 5



Bottom-Up Parsing

Bottom-Up Parsing I

Example 8.12

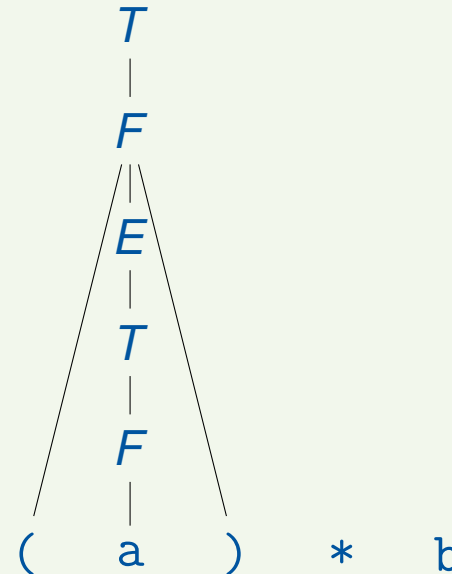
Grammar for
arithmetic expressions:

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

Reversed rightmost analysis

of $(a)*b$:

6 4 2 5 4



Bottom-Up Parsing

Bottom-Up Parsing I

Example 8.12

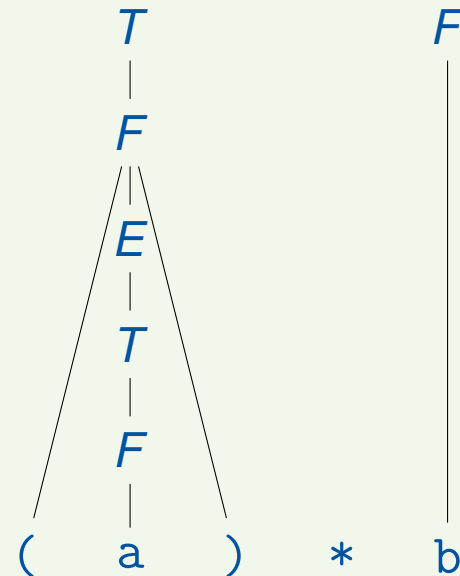
Grammar for
arithmetic expressions:

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

Reversed rightmost analysis

of $(a)*b$:

6 4 2 5 4 7



Bottom-Up Parsing

Bottom-Up Parsing I

Example 8.12

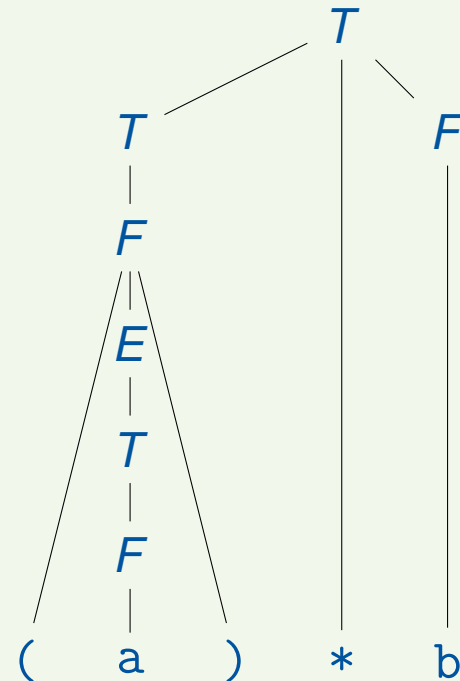
Grammar for
arithmetic expressions:

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

Reversed rightmost analysis

of $(a)*b$:

6 4 2 5 4 7 3



Bottom-Up Parsing

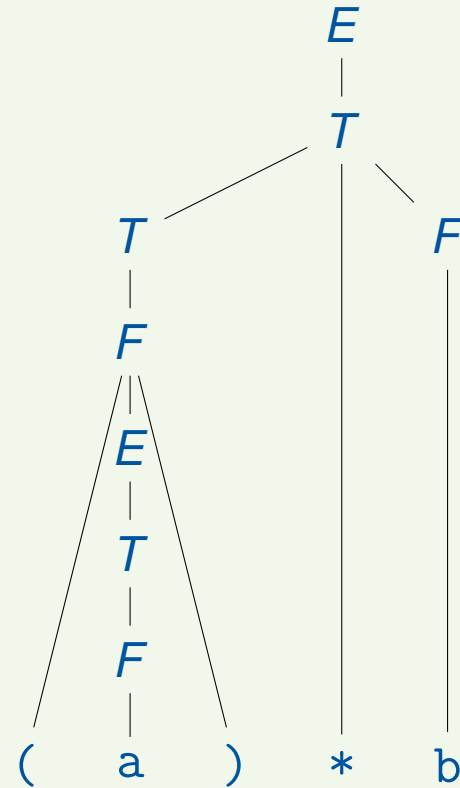
Bottom-Up Parsing I

Example 8.12

Grammar for
arithmetic expressions:

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

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 step)

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 step)
2. Remove nondeterminism by allowing **lookahead** on the input:
 $G \in LR(k)$ iff $L(G)$ recognisable by deterministic bottom-up parsing automaton with lookahead of k symbols

Nondeterministic Bottom-Up Parsing

Outline of Lecture 8

Recap: $LL(1)$ Parsing

Transformation to $LL(1)$

The Complexity of $LL(1)$ Parsing

Recursive-Descent Parsing

Bottom-Up Parsing

Nondeterministic Bottom-Up Parsing

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton I

Definition 8.13 (Nondeterministic bottom-up parsing automaton)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_{\Sigma}$. The **nondeterministic bottom-up parsing automaton** of G , $NBA(G)$, 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 $\pi_i = A \rightarrow \beta$
- **Initial configuration** for $w \in \Sigma^*$: $(w, \varepsilon, \varepsilon)$
- **Final configurations:** $\{\varepsilon\} \times \{S\} \times [p]^*$

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

$$((a)*b, \varepsilon, \varepsilon)$$

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

$\vdash ()*b, (F, 6)$

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

$\vdash () * b, (F, 6)$
 $\vdash () * b, (T, 64)$

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

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

$((a) * b, \varepsilon, \varepsilon)$
 $\vdash (a) * b, (, \varepsilon)$
 $\vdash () * b, (a, \varepsilon)$

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

$\vdash () * b, (F, 6)$
 $\vdash () * b, (T, 64)$
 $\vdash () * b, (E, 642)$

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

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

$((a) * b, \varepsilon, \varepsilon)$
 $\vdash (a) * b, (, \varepsilon)$
 $\vdash () * b, (a, \varepsilon)$

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

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

Nondeterministic Bottom-Up Parsing

The Nondeterministic Bottom-Up Automaton II

Example 8.14

Grammar for
arithmetic expressions
(cf. Example 8.12):

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

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

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

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

Nondeterministic Bottom-Up Parsing

Correctness of $NBA(G)$

Theorem 8.15 (Correctness of $NBA(G)$)

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

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

Nondeterministic Bottom-Up Parsing

Correctness of $NBA(G)$

Theorem 8.15 (Correctness of $NBA(G)$)

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

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

Proof.

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