Compiler Construction

Lecture 10: Syntactic Analysis V ($LR(k)$ Grammars)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/cc10/

Winter semester 2010/11

Outline

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 10.1 (Arithmetic expressions; cf. Example ??)

```
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

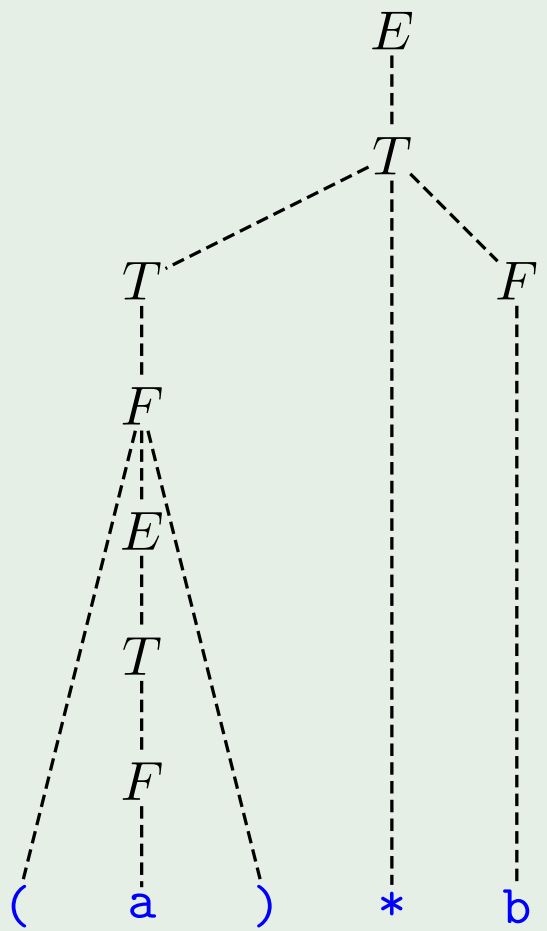
Repetition: Top-Down Parsing

Example 10.2

Grammar for arithmetic expressions:

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

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

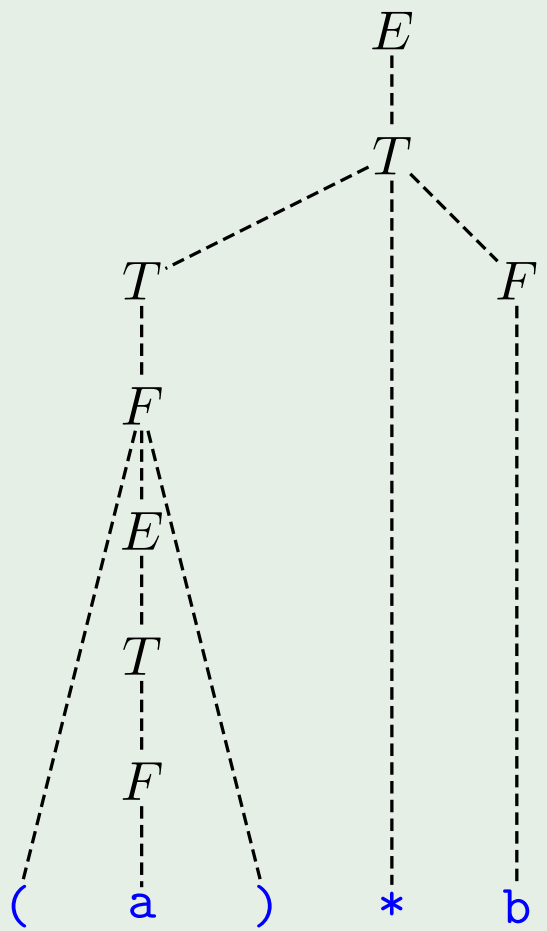


Example 10.3

Grammar for
arithmetic expressions:

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

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)
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 10.4 (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: $\{\varepsilon\} \times \{S\} \times [p]^*$

Nondeterministic Bottom-Up Automaton II

Example 10.5

Grammar for arithmetic expressions (cf. Example 7.3):

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

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 NBA(G)

Theorem 10.6 (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* $z \in [p]^*$,

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

Proof.

similar to the top-down case (Theorem 7.7) □

Nondeterminisn in NBA(G)

Remark: 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:

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

Resolving Termination Nondeterminism I

General assumption to avoid nondeterminism of last type:
every grammar is start separated

Definition 10.7 (Start separation)

A grammar $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ is called start separated if S only occurs in productions of the form $S \to A$ where $A \neq S$.

Remarks:

- Start separation always possible by adding $S' \to S$ with new start symbol S'
- From now on consider only reduced grammars of this form ($\pi_0 = S' \to S$)

Resolving Termination Nondeterminism II

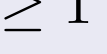
Start separation removes last form of nondeterminism (when to terminate parsing?):

Lemma 10.8

If $G \in CFG_\Sigma$ is start separated, then no successor of a final configuration (ε, S', z) in $\mathrm{NBA}(G)$ is again a final configuration. (Thus parsing should be stopped in the first final configuration.)

Proof.

- To (ε, S', z), only reductions by ε-productions can be applied:
$$(\varepsilon, S', z) \vdash (\varepsilon, S'A, zi) \quad \text{if } \pi_i = A \to \varepsilon$$
- Thereafter, only reductions by productions of the form $A_0 \to A_1 \dots A_n$ $(n \geq 0)$ can be applied
- Every resulting configuration is of the (non-final) form
$$(\varepsilon, S'B_1 \dots B_k, z) \quad \text{where } k \geq 1$$



Outline

$LR(k)$ Grammars I

Goal: resolve remaining nondeterminism of NBA(G) by supporting lookahead of $k \in \mathbb{N}$ symbols on the input

$\Longrightarrow$ $LR(k)$: reading of input from left to right with k-lookahead, computing a rightmost analysis

Definition 10.9 ($LR(k)$ grammar)

Let $G = \langle N, \Sigma, P, S \rangle \in CFG_\Sigma$ be start separated and $k \in \mathbb{N}$. Then G has the $LR(k)$ property (notation: $G \in LR(k)$) if for all rightmost derivations of the form

$$S \begin{cases} \Rightarrow_r^* \alpha A w \Rightarrow_r \alpha\beta w \\ \Rightarrow_r^* \gamma B x \Rightarrow_r \alpha\beta y \end{cases}$$

such that $\text{first}_k(w) = \text{first}_k(y)$, it follows that $\alpha = \gamma$, $A = B$, and $x = y$.

Remarks:

- If $G \in LR(k)$, then the reduction of $\alpha\beta w$ to $\alpha A w$ is already determined by $\text{first}_k(w)$.
- Therefore NBA(G) in configuration $(w, \alpha\beta, z)$ can decide to reduce and how to reduce.
- Computation of NBA(G) for $S \Rightarrow_r^* \alpha A w \Rightarrow_r \alpha\beta w$:

$$(w'w, \varepsilon, \varepsilon) \vdash^* (w, \alpha\beta, z) \overset{\text{red } i}{\vdash} (w, \alpha A, zi) \vdash \ldots$$

  where $\pi_i = A \to \beta$
- Computation of NBA(G) for $S \Rightarrow_r^* \gamma B x \Rightarrow_r \alpha\beta y$:
  - with direct reduction ($y = x$, $\alpha\beta = \gamma\delta$, $\pi_j = B \to \delta$):

$$(y'y, \varepsilon, \varepsilon) \vdash^* (y, \alpha\beta, z') = (x, \gamma\delta, z') \overset{\text{red } j}{\vdash} (x, \gamma B, z'j) \vdash \ldots$$

  - with previous shifts ($y = x'x$, $\alpha\beta x' = \gamma\delta$, $\pi_j = B \to \delta$):

$$\begin{aligned}(y'y, \varepsilon, \varepsilon) &\vdash^* (y, \alpha\beta, z') = (x'x, \alpha\beta, z') \\ &\overset{\text{shift}^*}{\vdash} (x, \alpha\beta x', z') = (x, \gamma\delta, z') \\ &\overset{\text{red } j}{\vdash} (x, \gamma B, z'j) \vdash \ldots\end{aligned}$$