Compiler Construction

Lecture 9: Syntactic Analysis IV ($LL(1)$ Parsing)

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

$LL(k)$ Grammars

Lemma (Characterization of $LL(k)$)

$G \in LL(k)$ iff for all leftmost derivations of the form

$$S \Rightarrow_l^* wA\alpha \begin{cases} \Rightarrow_l w\beta\alpha \\ \Rightarrow_l w\gamma\alpha \end{cases}$$

such that $\beta \neq \gamma$, it follows that $\text{first}_k(\beta\alpha) \cap \text{first}_k(\gamma\alpha) = \emptyset$.

Proof.

omitted □

Remarks:

- If $G \in LL(k)$, then the A-production is determined by the lookahead sets $\text{first}_k(\beta\alpha)$ (for every $A \to \beta \in P$).
- Problem: still infinitely many rightmost contexts α to be considered (if β/γ "too short", i.e., $\text{first}_k(\beta\alpha) \neq \text{first}_k(\beta)$).

Lookahead Sets

Definition (Lookahead set)

Given $\pi = A \to \beta \in P$,

$$\mathrm{la}(\pi) := \mathrm{fi}(\beta \cdot \mathrm{fo}(A)) \subseteq \Sigma_\varepsilon$$

is called the lookahead set of π (where $\mathrm{fi}(\Gamma) := \bigcup_{\gamma \in \Gamma} \mathrm{fi}(\gamma)$).

Corollary

1. *For all* $a \in \Sigma$,
$$a \in \mathrm{la}(A \to \beta) \text{ iff } a \in \mathrm{fi}(\beta) \text{ or } (\beta \Rightarrow^* \varepsilon \text{ and } a \in \mathrm{fo}(A))$$
2. $\varepsilon \in \mathrm{la}(A \to \beta)$ *iff* $\beta \Rightarrow^* \varepsilon$ *and* $\varepsilon \in \mathrm{fo}(A)$

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)

Computing Lookahead Sets II

Corollary

1. $A \to a\alpha \in P \implies a \in \mathrm{fi}(A)$
2. $A \to B\alpha \in P, a \in \mathrm{fi}(B) \implies a \in \mathrm{fi}(A)$
3. $A \to \varepsilon \in P \implies \varepsilon \in \mathrm{fi}(A)$
4. $\mathrm{fi}(\varepsilon) = \{\varepsilon\}$
5. $a \in \mathrm{fi}(A) \implies a \in \mathrm{fi}(A\alpha)$
6. $A \to \alpha B \in P, x \in \mathrm{fo}(A) \implies x \in \mathrm{fo}(B)$

Example

Grammar for arithmetic expressions (cf. Example 7.3):

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

- $F \to \texttt{a} \in P \implies \texttt{a} \in \mathrm{fi}(F)$
- $T \to F \in P, \texttt{a} \in \mathrm{fi}(F) \implies \texttt{a} \in \mathrm{fi}(T)$
- $\texttt{a} \in \mathrm{fi}(T)$
  $\implies \mathrm{la}(T \to T\texttt{*}F) = \mathrm{fi}(T\texttt{*}F \cdot \mathrm{fo}(T)) \ni \texttt{a}$
- $\texttt{a} \in \mathrm{fi}(F)$
  $\implies \mathrm{la}(T \to F) = \mathrm{fi}(F \cdot \mathrm{fo}(T)) \ni \texttt{a}$
- $\implies \texttt{a} \in \mathrm{la}(T \to T\texttt{*}F) \cap \mathrm{la}(T \to F) \neq \emptyset$
- $\implies G_{AE} \notin LL(1)$

Fixing the Problem

(general methods later)

Example 9.1 (continuing Example 8.12)

Restructuring (such that $L(G'_{AE}) = L(G_{AE})$):

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

$A \in N$	fi(A)	fo(A)
E	$\{\texttt{(}, \texttt{a}, \texttt{b}\}$	$\{\varepsilon, \texttt{)}\}$
E'	$\{\texttt{+}, \varepsilon\}$	$\{\varepsilon, \texttt{)}\}$
T	$\{\texttt{(}, \texttt{a}, \texttt{b}\}$	$\{\texttt{+}, \varepsilon, \texttt{)}\}$
T'	$\{\texttt{*}, \varepsilon\}$	$\{\texttt{+}, \varepsilon, \texttt{)}\}$
F	$\{\texttt{(}, \texttt{a}, \texttt{b}\}$	$\{\texttt{*}, \texttt{+}, \varepsilon, \texttt{)}\}$

$A \to \beta \in P$	la$(A \to \beta) =$ fi$(\beta \cdot$ fo$(A))$
$E \to TE'$	$\{\texttt{(}, \texttt{a}, \texttt{b}\}$
$E' \to \texttt{+}TE'$	$\{\texttt{+}\}$
$E' \to \varepsilon$	$\{\varepsilon, \texttt{)}\}$
$T \to FT'$	$\{\texttt{(}, \texttt{a}, \texttt{b}\}$
$T' \to \texttt{*}FT'$	$\{\texttt{*}\}$
$T' \to \varepsilon$	$\{\texttt{+}, \varepsilon, \texttt{)}\}$
$F \to \texttt{(}E\texttt{)}$	$\{\texttt{(}\}$
$F \to \texttt{a}$	$\{\texttt{a}\}$
$F \to \texttt{b}$	$\{\texttt{b}\}$

$\Longrightarrow$ $G'_{AE} \in LL(1)$

Outline

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 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 DTA(G)

Remarks:

- 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 $a \notin \bigcup_{A \to \beta \in P} \mathrm{la}(A \to \beta)$)

The Deterministic Top-Down Automaton I

Definition 9.2 (Deterministic top-down parsing automaton)

Let $G = \langle N, \Sigma, P, S \rangle \in LL(1)$. The deterministic top-down parsing automaton of G, $\mathrm{DTA}(G)$, 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 $\mathrm{NTA}(G)$)
- Action function

$$\begin{aligned} \mathrm{act} &: \Sigma_\varepsilon \times X_\varepsilon \to \{(\alpha, i) \mid \pi_i = A \to \alpha\} \cup \{\mathsf{pop}, \mathsf{accept}, \mathsf{error}\} \\ \text{with } \mathrm{act}(x, A) &:= (\alpha, i) \text{ if } \pi_i = A \to \alpha \text{ and } x \in \mathrm{la}(\pi_i) \\ \mathrm{act}(a, a) &:= \mathsf{pop} \\ \mathrm{act}(\varepsilon, \varepsilon) &:= \mathsf{accept} \\ \mathrm{act}(x, y) &:= \mathsf{error} \qquad \text{otherwise} \end{aligned}$$

- Transitions for $x \in \Sigma_\varepsilon$, $w \in \Sigma^*$, $Y \in X$, $\beta \in X^*$, and $z \in [p]^*$:

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

The Deterministic Top-Down Automaton II

Example 9.3 (cf. Example 9.1)

$$
\begin{array}{lll}
G'_{AE}: & E \to TE' & (1) \\
& E' \to \texttt{+}TE' \mid \varepsilon & (2,3) \\
& T \to FT' & (4) \\
& T' \to \texttt{*}FT' \mid \varepsilon & (5,6) \\
& F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b} & (7,8,9)
\end{array}
$$

$A \to \beta \in P$	$\mathrm{la}(A \to \beta)$
$E \to TE'$	{(, a, b}
$E' \to \texttt{+}TE'$	{+}
$E' \to \varepsilon$	{ε,)}
$T \to FT'$	{(, a, b}
$T' \to \texttt{*}FT'$	{*}
$T' \to \varepsilon$	{+, ε,)}
$F \to \texttt{(}E\texttt{)}$	{(}
$F \to \texttt{a}$	{a}
$F \to \texttt{b}$	{b}

$\mathrm{act} : \Sigma_\varepsilon \times X_\varepsilon \to \{(\alpha, i) \mid \pi_i = A \to \alpha\} \cup \{\mathsf{pop}, \mathsf{accept}, \mathsf{error}\}$ (empty = error)

act	E	E'	T	T'	F	a	b	(	)	*	+	ε
a	$(TE', 1)$		$(FT', 4)$		$(\texttt{a}, 8)$	pop						
b	$(TE', 1)$		$(FT', 4)$		$(\texttt{b}, 9)$		pop					
(	$(TE', 1)$		$(FT', 4)$		$(\texttt{(}E\texttt{)}, 7)$			pop				
)		$(\varepsilon, 3)$		$(\varepsilon, 6)$					pop			
*				$(\texttt{*}FT', 5)$						pop		
+		$(\texttt{+}TE', 2)$		$(\varepsilon, 6)$							pop	
ε		$(\varepsilon, 3)$		$(\varepsilon, 6)$								accept

The Deterministic Top-Down Automaton III

Example 9.3 (continued)

act	E	E'	T	T'	F	a	b	(	)	*	+	ε
a	$(TE', 1)$		$(FT', 4)$		$(\mathtt{a}, 8)$	pop						
b	$(TE', 1)$		$(FT', 4)$		$(\mathtt{b}, 9)$		pop					
(	$(TE', 1)$		$(FT', 4)$		$((E), 7)$			pop				
)		$(\varepsilon, 3)$		$(\varepsilon, 6)$					pop			
*				$(*FT', 5)$						pop		
+		$(+TE', 2)$		$(\varepsilon, 6)$							pop	
ε		$(\varepsilon, 3)$		$(\varepsilon, 6)$								accept

Leftmost analysis of `(a)*b`:

$$
\begin{array}{ll}
 & (\mathtt{(a)*b}, E, \varepsilon) \\
\vdash & (\mathtt{(a)*b}, TE', 1) \\
\vdash & (\mathtt{(a)*b}, FT'E', 14) \\
\vdash & (\mathtt{(a)*b}, \mathtt{(}E\mathtt{)}T'E', 147) \\
\vdash & (\mathtt{\ a)*b}, E\mathtt{)}T'E', 147) \\
\vdash & (\mathtt{\ a)*b}, TE'\mathtt{)}T'E', 1471) \\
\vdash & (\mathtt{\ a)*b}, FT'E'\mathtt{)}T'E', 14714) \\
\vdash & (\mathtt{\ a)*b}, \mathtt{a}T'E'\mathtt{)}T'E', 147148) \\
\vdash & (\mathtt{\ \ )*b}, T'E'\mathtt{)}T'E', 147148)
\end{array}
\qquad
\begin{array}{ll}
\vdash & (\mathtt{)*b}, E'\mathtt{)}T'E', 1471486) \\
\vdash & (\mathtt{)*b}, \mathtt{)}T'E', 14714863) \\
\vdash & (\mathtt{\ *b}, T'E', 14714863) \\
\vdash & (\mathtt{\ *b}, \mathtt{*}FT'E', 147148635) \\
\vdash & (\mathtt{\ \ b}, FT'E', 147148635) \\
\vdash & (\mathtt{\ \ b}, \mathtt{b}T'E', 1471486359) \\
\vdash & (\mathtt{\ \ b}, \varepsilon T'E', 1471486359) \\
\vdash & (\mathtt{\ \ b}, \varepsilon E', 14714863596) \\
\vdash & (\mathtt{\ \ b}, \varepsilon\varepsilon, 147148635963)
\end{array}
$$

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

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 9.4 (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 9.5

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 9.6

The grammar (cf. Example 7.3)

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

is left recursive, and in Example 8.12 it was shown that $G_{AE} \notin LL(1)$

Left Recursion II

Lemma 9.7

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 DTA(G) (Def. 9.2) starts with
$(vw, S, \varepsilon) \vdash^* (w, A\alpha, \ldots) \vdash^+ (w, A\beta\alpha, \ldots)$.
But in the last state the behaviour of DTA(G) is determined by the same input (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

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

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 9.8

G_{AE} : $E \to E\texttt{+}T \mid T$, $T \to T\texttt{*}F \mid F$, $F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b}$ is transformed into

G'_{AE} : $E \to TE'$, $E' \to \texttt{+}TE' \mid \varepsilon$, $T \to FT'$, $T' \to \texttt{*}FT' \mid \varepsilon$, $F \to \texttt{(}E\texttt{)} \mid \texttt{a} \mid \texttt{b}$ with $G'_{AE} \in LL(1)$ (see Example 9.1).

Removing Indirect Left Recursion

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

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

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

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

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 9.9

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

is transformed into

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

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)
- 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 9.10

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*

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

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 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 9.7, G is not left recursive), and hence $k \leq |N|$.
4. Altogether: $n \leq (|w| + 1) \cdot (|N| + 1)$.

□