Parsing

Deterministic Top-Down Parsing: LL(k) Parsing

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Winter 2017/18

Table of contents

Introduction (1)

Top-Down Parsing: Scan and Predict with

$$\text{Predict: } \frac{[A\alpha, i]}{[\gamma\alpha, i]} \; A \rightarrow \gamma \in P$$

Problem: in general highly non-deterministic.
Better if grammar in GNF but still non-deterministic.

Goal: find grammars that allow for deterministic top-down parsing.

Introduction (2)

Idea: Use the next terminal symbol(s) as lookahead to determine which production to predict.

Example: 1 lookahead

Productions $A \to X\beta$ and $A \to Y\gamma$ such that $X \stackrel{*}{\Rightarrow} b\beta'$ and $Y \stackrel{*}{\Rightarrow} c\gamma'$.

Then:

stack	input
$A\Gamma$	$b\ldots$
$X\beta\Gamma$	$b\ldots$

stack	remaining input
$A\Gamma$	$c\ldots$
$Y\gamma\Gamma$	$c\ldots$

Deterministic, if neither $X \stackrel{*}{\Rightarrow} c\ldots$ nor $Y \stackrel{*}{\Rightarrow} b\ldots$

LL(1) grammars (1)

Intuition: A CFG is LL(1) if it allows for a deterministic top-down parsing with 1 lookahead. In order to define LL(1), we define First and Follow.

First and Follow

Let $\alpha \in (N \cup T)^*$.

$$First(\alpha) = \{a \mid \alpha \overset{*}{\Rightarrow} a\beta, a \in T, \beta \in (N \cup T)^*\} \cup \{\epsilon \mid \alpha \overset{*}{\Rightarrow} \epsilon\}$$

Let $A \in N$.

$$Follow(A) = \{a \mid S \overset{*}{\Rightarrow} \alpha A a \beta, a \in T, \alpha, \beta \in (N \cup T)^*\}$$

$$\cup \{\$ \mid S \overset{*}{\Rightarrow} \alpha A, \alpha \in (N \cup T)^*\}$$

where \$ is a new symbol marking the end of the input.

LL(1) grammars (2)

Examples

1. G_1: $S \rightarrow ab \mid aSb$
   $First(ab) = First(aSb) = \{a\}$
   $Follow(S) = \{b, \$\}$
2. G_2: $S \rightarrow aB \mid bA, A \rightarrow a \mid aS \mid bAA, B \rightarrow b \mid bS \mid aBB$
   $First(aB) = \{a\}, First(bA) = \{b\}$
   $First(a) = First(aS) = \{a\}, First(bAA) = \{b\}$
   $Follow(S) = \{a, b, \$\}$
3. G_3: $S \rightarrow aT, T \rightarrow b \mid Sb$
   $First(S) = First(aT) = \{a\}, First(b) = \{b\}, First(Sb) = \{a\}$

LL(1) grammars (3)

LL(1)-grammar

A CFG G is a **LL(1)-grammar** if for all $A \in N$:
Let $A \to \alpha_1 | \ldots | \alpha_n$ be all A-productions in G. then

- $\mathit{First}(\alpha_1), \ldots, \mathit{First}(\alpha_n)$ are pairwise disjoint, and
- if $\epsilon \in \mathit{First}(\alpha_j)$ for some $j \in [1..n]$, then $\mathit{Follow}(A) \cap \mathit{First}(\alpha_i) = \emptyset$ for all $1 \leq i \leq n, j \neq i$.

G_1 and G_2 are not LL(1), G_3 is LL(1).

There are CFLs that cannot be generated by a LL(1)-grammar.
Example: $\{a^n c b^n \mid n \geq 0\} \cup \{a^n d b^{2n} \mid n \geq 0\}$

LL(1) grammars (4)

Transformations that can help to obtain an equivalent LL(1) grammar:

- Elimination of left-recursion.
- Left-factoring: elimination of A-productions whose rhs have the same prefix:
  Replace $A \rightarrow \alpha\beta_1, \ldots, A \rightarrow \alpha\beta_n$ ($\alpha \in (N \cup T)^+$) with $A \rightarrow \alpha A'$, $A' \rightarrow \beta_1, \ldots, A' \rightarrow \beta_n$ where A' is a new non-terminal.

  Example: Transformation from G_1 to G_3.

Computing First and Follow (1)

First computation

- Computing *First* sets for single non-terminals:
  1. For all $X \in N \cup T$: $First(X) = \emptyset$.
     If $X \in T$, then add X to $First(X)$.
     If $X \to \epsilon \in P$, then add ϵ to $First(X)$.
  2. Do the following repeatedly until the *First*-sets do not change any more:
     For each production $X \to X_1 \ldots X_n$ with $n \geq 1$, add $a \in T$ to $First(X)$ if there is an $i \in [1..n]$ such that
     (i) $a \in First(X_i)$, and
     (ii) $\epsilon \in First(X_j)$ for all $1 \leq j < i$.
     If $\epsilon \in First(X_j)$ for all $1 \leq j \leq n$, then add ϵ to $First(X)$.
- For all $\alpha \in (N \cup T)^+$: Add a new nonterminal X_α and a production $X_\alpha \to \alpha$ and then compute $First(\alpha) = First(X_\alpha)$.
- $First(\epsilon) = \{\epsilon\}$.

Computing First and Follow (2)

First computation with deduction rules

Computing items $[X, t]$ with $X \in N \cup T$, $t \in T \cup \{\varepsilon\}$ such that $[X, t]$ iff $t \in \mathit{First}(X)$

Terminals: $\dfrac{}{[a, a]}\ a \in T$

ε-productions: $\dfrac{}{[A, \varepsilon]}\ A \to \varepsilon \in P$

Bottom-up propagation:

$$\frac{[B, X], [X_1, \varepsilon], \ldots, [X_k, \varepsilon]}{[A, X]}\ A \to X_1 \cdots X_k B\beta \in P, X \neq \varepsilon \text{ or } \beta = \varepsilon$$

Computing First and Follow (3)

Computing *Follow*

Let \$ be a new symbol (the end marker).

- For every $A \in N$: $\mathit{Follow}(A) = \emptyset$.
- Add \$ to $\mathit{Follow}(S)$.
- Do the following until the *Follow*-sets do not change any more:
  For each $A \rightarrow \alpha B \beta \in P$ with $\alpha, \beta \in (N \cup T)^*, B \in N$:
  - add $\mathit{First}(\beta) \cap T$ to $\mathit{Follow}(B)$.
  - if $\epsilon \in \mathit{First}(\beta)$, then add $\mathit{Follow}(A)$ to $\mathit{Follow}(B)$.

(We assume all $A \in N$ to be reachable.)

Computing First and Follow (4)

Computing *Follow* with deduction rules

Computing items $[A, t]$ with $A \in N$, $t \in T \cup \{\$\}$ such that $[A, t]$ iff $t \in \mathit{Follow}(A)$

Axiom: $\dfrac{}{[S, \$]}$

Right-to-left propagation:

$$\frac{}{[B, a]} \quad A \to \alpha B X_1 \dots X_k C \beta \in P, [X_1, \varepsilon], \dots, [X_k, \varepsilon], [C, a] \in \mathit{First}$$

Top-down propagation:

$$\frac{[A, X]}{[B, X]} \quad A \to \alpha B X_1 \dots X_k \in P, [X_1, \varepsilon], \dots, [X_k, \varepsilon] \in \mathit{First}$$

LL(1) parsing (1)

If a CFG is a LL(1) grammar, then it allows for a deterministic top-down parsing where the next input symbol as lookahead determines the predict step to take.

We construct a parsing table that tells us, depending on

- the topmost stack symbol and
- the next input symbol,

which production we have to predict.

LL(1) parsing (2)

Example

G_3: $S \to aT,\ T \to b \mid Sb$

$First(aT) = a$, $First(b) = b$, $First(Sb) = a$

parse table:

	S	T
a	$S \to aT$	$T \to Sb$
b	–	$T \to b$

Stack	remaining input
S	**a**abb
aT	aabb
T	**a**bb
Sb	**a**bb
aTb	abb
Tb	**b**b
bb	bb
b	b
–	–

LL(1) parsing (3)

Construction of the parsing table M

For each production $A \rightarrow \alpha$:

- For every $a \in T$ with $a \in \mathit{First}(\alpha)$: $M(A, a) = A \rightarrow \alpha$.
- If $\epsilon \in \mathit{First}(\alpha)$, then for each $b \in \mathit{Follow}(A)$: $M(A, b) = A \rightarrow \alpha$.

Example: LL(1) parsing table construction

$S \rightarrow ABC,\ A \rightarrow aA \mid \varepsilon,\ B \rightarrow cB \mid bB \mid \varepsilon,\ C \rightarrow d$

Parsing table:

	S	A	B	C
a	$S \rightarrow ABC$	$A \rightarrow aA$	–	–
b	$S \rightarrow ABC$	$A \rightarrow \varepsilon$	$B \rightarrow bB$	–
c	$S \rightarrow ABC$	$A \rightarrow \varepsilon$	$B \rightarrow cB$	–
d	$S \rightarrow ABC$	$A \rightarrow \varepsilon$	$B \rightarrow \varepsilon$	$C \rightarrow d$

LL(k) parsing

If more than one symbol as lookahead is used, namely up to k symbols, the technique is called LL(k) parsing.

The definitions of *First* and *Follow* must be extended to contain terminal strings of up to k symbols.

The parse table gets much larger of course.

A CFG is LL(k) if it allows for deterministic top-down parsing with k lookahead symbols.

Conclusion

- LL(1) grammars allow for a deterministic top-down parsing.
- The next terminal in the remaining input (the lookahead) determines the predict step to take.
- *First* and *Follow* and the parse table can be precompiled.
- The set of languages generated by LL(1) grammars is a proper subset of CFL.