LL(1) Parsing

Greta Yorsh Yu-Yang Lin Hou Julian Nagele

Tutorial 05

February 12 2018

ECS652: Compilers (Spring 2018)

Top-down Parsing

- traversal of the parse tree in preorder

Backtracking Parsers

- try different possibilities, backing up an arbitrary amount in the input if one possibility fails
- powerful but slow
- unsuitable for practical compilers

Predictive Parsers

- predict the next construction using one or more look-ahead tokens
- recursive-descent parsing = "hand-written" parsers
- LL(1) parsing = process input from left to right, trace out a leftmost derivation and use 1 look-ahead symbol

LL(1) Parsing

- use an explicit stack rather than recursive calls
- mark the bottom of the stack with EOF (e.g. a \$ sign)

Two Basic Actions

- *prediction*: replace a non-terminal A at the top of the stack by a string α using the grammar rule $A \rightarrow \alpha$; α pushed onto the stack in reversed order of symbols
- *match* a token on top of the stack with the next input token

Example

- grammar $S \rightarrow (\, S\,)\, S \mid \epsilon$
- input ()\$

LL(1) Parsing Table

Recall

context-free grammar G is 4-tuple $\langle V, T, P, S \rangle$ with

- V is set of non-terminals, T is set of terminals
- $S \in V$ is start symbol
- P is set of derivation rules of the form $N \to \alpha$ where $N \in V$ and $\alpha \in (V \cup T)^*$

- parsing table is 2-dimensional array indexed by N and T which contains rules to use when non-terminal is on top of the stack and terminal is next in the input
- grammar is LL(1) if table has at most one rule per entry

Example

table[N, T]	(	)	\$
S	$S \to (\, S \,)\, S$	$S \to \epsilon$	$S \to \epsilon$

First and Follow Sets

- how to build table?
- for a sequence α want terminal that begins strings derivable from α

Definition

let $t \in T$, $N \in V$, and $\alpha, \beta \in (V \cup T)^*$, then

- $\text{First}(t) = \{t\}$
- $\text{First}(N) = \{t \mid N \to^* t\beta\}$
- $\text{First}(\alpha) = \{t \mid \alpha \to^* t\beta\}$

Algorithm

whiteboard

Parsing Table Construction

for every rule $N \to \alpha$

for every $t \in \text{First}(\alpha)$, add $N \to \alpha$ to table$[N, t]$

Examples

- $S \to (\, S\,) \mid [\,]$
- $S \to (\, S\,)\, S \mid \epsilon$

Definition

let $t \in T$, $N \in V$, and $\alpha, \beta \in (V \cup T)^*$, then

- $\text{Follow}(N) = \{t \mid S \to^* \alpha N t \beta\}$
- $\text{Follow}(S)$ additionally contains \$

Algorithm

whiteboard

Parsing Table Construction

for every rule $N \to \alpha$
  for every $t \in \text{First}(\alpha)$, add $N \to \alpha$ to table$[N, t]$
  if $\alpha \to^* \epsilon$ then
    for every $t \in \text{Follow}(N)$, add $N \to \alpha$ to table$[N, t]$

Left-factoring

Examples

- $S \to (\, S\,) \mid (\,)$
- $\mathit{ifstmt} \to \text{if}\ (\ \mathit{exp}\ )\ \mathit{statement} \mid \text{if}\ (\ \mathit{exp}\ )\ \mathit{statement}\ \text{else}\ \mathit{statement}$

- required when rules share common prefix: $N \to \alpha\beta \mid \alpha\gamma$
- factor out α (pick longest common prefix of right-hand sides)
- $N \to \alpha N' \quad N' \to \beta \mid \gamma$

Example

- $\mathit{stmt} \to \mathit{assignstmt} \mid \mathit{callstmt} \mid \mathit{other} \qquad \mathit{assignstmt} \to \text{ident} := \mathit{exp}$
  $\mathit{callstmt} \to \text{ident}\ (\ \mathit{args}\ )$

- beware of "indirect" common prefixes – need to use substitution first

Left-recursion

Example

- $exp \rightarrow exp + term \mid term \quad term \rightarrow term * factor \mid factor$
  $factor \rightarrow (\, exp \,) \mid number$

- bounded look-ahead cannot deal with left recursion
- transform to right recursion
- $N \rightarrow N\alpha \mid \beta$
- $N \rightarrow \beta N' \quad N' \rightarrow \alpha N' \mid \epsilon$

Examples

- $stmtseq \rightarrow stmtseq \,;\, stmt \mid stmt$
- $stmtseq \rightarrow stmt \,;\, stmtseq \mid stmt$

Examples

- $S \to Bc \mid DB \quad B \to ab \mid cS \quad D \to d \mid \epsilon$
- $S \to AB \quad A \to Ca \mid \epsilon \quad B \to BaAC \mid c \quad C \to b \mid \epsilon$