Script **generated by TTT**

Title: Petter: Compiler Construction (14.05.2020) - 13: LL(1) Grammars

Date: Mon May 04 15:10:53 CEST 2020

Duration: 23:42 min

Pages: 9

Topdown Parsing

Problem:
Conflicts between the transitions prohibit an implementation of the item pushdown automaton as deterministic pushdown automaton.

Topdown Parsing

Problem:
Conflicts between the transitions prohibit an implementation of the item pushdown automaton as deterministic pushdown automaton.

Idea 1: GLL Parsing
For each conflict, we create a virtual copy of the complete configuration and continue computing in parallel.

Topdown Parsing

Problem:
Conflicts between the transitions prohibit an implementation of the item pushdown automaton as deterministic pushdown automaton.

Idea 1: GLL Parsing
For each conflict, we create a virtual copy of the complete configuration and continue computing in parallel.

Idea 2: Recursive Descent & Backtracking
Depth-first search for an appropriate derivation.

Topdown Parsing

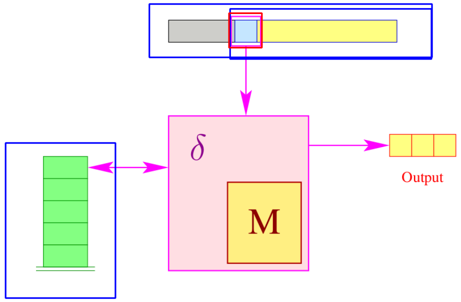
Problem:
Conflicts between the transitions prohibit an implementation of the item pushdown automaton as deterministic pushdown automaton.

Idea 1: GLL Parsing
For each conflict, we create a virtual copy of the complete configuration and continue computing in parallel.

Idea 2: Recursive Descent & Backtracking
Depth-first search for an appropriate derivation.

Idea 3: Recursive Descent & Lookahead
Conflicts are resolved by considering a lookup of the next input symbols.

Structure of the $LL(1)$-Parser:



- The parser accesses a frame of length 1 of the input;
- it corresponds to an item pushdown automaton, essentially;
- table $M[q, w]$ contains the rule of choice.

Topdown Parsing

Idea:

- Emanate from the item pushdown automaton
- Consider the next input symbol to determine the appropriate rule for the next expansion
- A grammar is called $LL(1)$ if a unique choice is always possible

Topdown Parsing

Idea:

- Emanate from the item pushdown automaton
- Consider the next input symbol to determine the appropriate rule for the next expansion
- A grammar is called $LL(1)$ if a unique choice is always possible

Philip Lewis

Richard Stearns

Definition:
A reduced grammar is called $LL(1)$, if for each two distinct rules $A \to \alpha\,,\ \ A \to \alpha' \in P$ and each derivation $S \to^*_L u\, A\, \beta$ with $u \in T^*$ the following is valid:

$$\text{First}_1(\alpha\,\beta)\ \cap\ \text{First}_1(\alpha'\,\beta) = \emptyset$$

Topdown Parsing

Example 1:

1st

$$\begin{array}{lll} S & \to & 1\ \text{if } (E) \; \boxed{S} \text{ else } \boxed{S} \mid \\ & & 2\ \text{while } (E) \; \boxed{S} \mid \\ & & E\,; \\ \boxed{E} & \to & 3\ \text{id} \end{array}$$

✓

is $LL(1)$, since ✓ $\text{First}_1(E) = \{\text{id}\}$

Topdown Parsing

Example 1:

$$\begin{array}{lll} S & \to & \text{if } (E) \; S \text{ else } S \mid \\ & & \text{while } (E) \; S \mid \\ & & E\,; \\ E & \to & \text{id} \end{array}$$

is $LL(1)$, since $\text{First}_1(E) = \{\text{id}\}$

Example 2:

1st 2nd 3rd 4th

$$\begin{array}{llll} S & \to & \text{if } (E) \; S \text{ else } S \mid & 1 \\ & & \text{if } (E) \; S \mid & 2 \\ & & \text{while } (E) \; S \mid & \\ & & E\,; & \\ E & \to & \text{id} & \end{array}$$

✗

... is not $LL(k)$ for any $k > 0$.