Script generated by TTT

Title: Petter: Compilerbau (17.05.2018)

Date: Thu May 17 14:17:29 CEST 2018

Duration: 30:50 min

Pages: 21

Topdown Parsing

Example 1:

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

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

Example 2:

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

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

Lookahead Sets

Definition: First_1-Sets

For a set $L \subseteq T^*$ we define:

$$\text{First}_1(L) \;=\; \{\epsilon \mid \epsilon \in L\} \cup \{u \in T \mid \exists\, v \in T^* :\; uv \in L\}$$

Example: $S \to \epsilon \quad | \quad a\,S\,b$

$\text{First}_1(S)$
ϵ
$a\,b$
$a\,a\,b\,b$
$a\,a\,a\,b\,b\,b$
$\ldots$

Lookahead Sets

Arithmetics:

$\text{First}_1(\_)$ is distributive with union and concatenation:

$$
\begin{array}{lcl}
\text{First}_1(\emptyset) & = & \emptyset \\
\text{First}_1(L_1 \cup L_2) & = & \text{First}_1(L_1) \cup \text{First}_1(L_2) \\
\text{First}_1(L_1 \cdot L_2) & = & \text{First}_1(\text{First}_1(L_1) \cdot \text{First}_1(L_2)) \\
 & := & \text{First}_1(L_1) \odot_1 \text{First}_1(L_2)
\end{array}
$$

$\odot_1$ being $1-$concatenation

Lookahead Sets

Arithmetics:

$\mathsf{First}_1(\_)$ is distributive with union and concatenation:

$$\begin{array}{lcl} \mathsf{First}_1(\emptyset) & = & \emptyset \\ \mathsf{First}_1(L_1 \cup L_2) & = & \mathsf{First}_1(L_1) \cup \mathsf{First}_1(L_2) \\ \mathsf{First}_1(L_1 \cdot L_2) & = & \mathsf{First}_1(\mathsf{First}_1(L_1) \cdot \mathsf{First}_1(L_2)) \\ & := & \mathsf{First}_1(L_1) \odot_1 \mathsf{First}_1(L_2) \end{array}$$

$\odot_1$ being $1-$concatenation

Definition: 1-concatenation

Let $L_1, L_2 \subseteq T \cup \{\epsilon\}$ with $L_1 \neq \emptyset \neq L_2$. Then:

$$L_1 \odot_1 L_2 = \begin{cases} L_1 & \text{if } \epsilon \notin L_1 \\ (L_1 \backslash \{\epsilon\}) \cup L_2 & \text{otherwise} \end{cases}$$

If all rules of G are productive, then all sets $\mathsf{First}_1(A)$ are non-empty.

Lookahead Sets

For $\alpha \in (N \cup T)^*$ we are interested in the set:

$$\mathsf{First}_1(\alpha) = \mathsf{First}_1(\{w \in T^* \mid \alpha \to^* w\})$$

Idea: Treat ϵ separately: $\mathsf{First}_1(A) = F_\epsilon(A) \cup \{\epsilon \mid A \to^* \epsilon\}$

- Let $\mathsf{empty}(X) = \mathsf{true}$ iff $X \to^* \epsilon$.
- $F_\epsilon(X_1 \ldots X_m) = \bigcup_{i=1}^{j} F_\epsilon(X_i)$ if $\bigwedge_{i=1}^{j-1} \mathsf{empty}(X_i) \wedge \neg\mathsf{empty}(X_j)$

Lookahead Sets

for example...

$$\begin{array}{lcl} E & \to & E + T \mid T \\ T & \to & T * F \mid F \\ F & \to & (\ E\ ) \mid \mathsf{name} \mid \mathsf{int} \end{array}$$

with $\mathsf{empty}(E) = \mathsf{empty}(T) = \mathsf{empty}(F) = \mathsf{false}$

Fast Computation of Lookahead Sets

Observation:

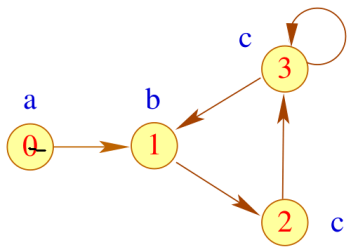
- The form of each inequality of these systems is:

  $$x \sqsupseteq y \quad \text{resp.} \quad x \sqsupseteq d$$

  for variables x, y und $d \in \mathrm{D}$.
- Such systems are called pure unification problems
- Such problems can be solved in linear space/time.

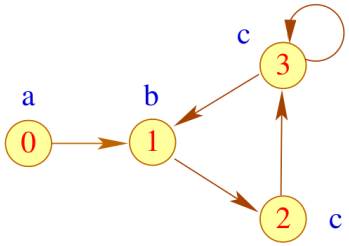
for example: $\mathrm{D} = 2^{\{a,b,c\}}$

$$\begin{array}{lll} x_0 \supseteq \{a\} & & \\ x_1 \supseteq \{b\} & x_1 \supseteq x_0 & x_1 \supseteq x_3 \\ x_2 \supseteq \{c\} & x_2 \supseteq x_1 & \\ x_3 \supseteq \{c\} & x_3 \supseteq x_2 & x_3 \supseteq x_3 \end{array}$$



Fast Computation of Lookahead Sets

Frank DeRemer
& Tom Pennello



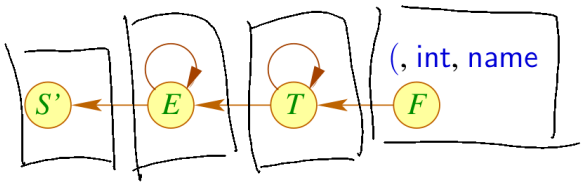
Proceeding:

- Create the Variable Dependency Graph for the inequality system.

Fast Computation of Lookahead Sets

... for our example grammar:

First_1 :



Item Pushdown Automaton as LL(1)-Parser

back to the example: $S' \to S\ \$ \qquad S \to \epsilon \mid a\ S\ b$

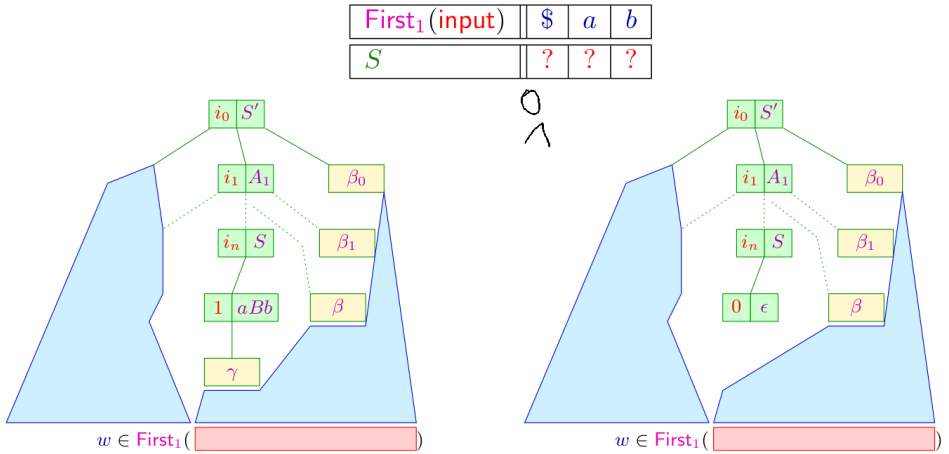
The transitions in the according Item Pushdown Automaton:

0	$[S' \to \bullet\ S\ \$]$	ϵ	$[S' \to \bullet\ S\ \$]\ [S \to \bullet]$
1	$[S' \to \bullet\ S\ \$]$	ϵ	$[S' \to \bullet\ S\ \$]\ [S \to \bullet\ a\ S\ b]$
2	$[S \to \bullet\ a\ S\ b]$	a	$[S \to a \bullet\ S\ b]$
3	$[S \to a \bullet\ S\ b]$	ϵ	$[S \to a \bullet\ S\ b]\ [S \to \bullet]$
4	$[S \to a \bullet\ S\ b]$	ϵ	$[S \to a \bullet\ S\ b]\ [S \to \bullet\ a\ S\ b]$
5	$[S \to a \bullet\ S\ b]\ [S \to \bullet]$	ϵ	$[S \to a\ S \bullet\ b]$
6	$[S \to a \bullet\ S\ b]\ [S \to a\ S\ b \bullet]$	ϵ	$[S \to a\ S \bullet\ b]$
7	$[S \to a\ S \bullet\ b]$	b	$[S \to a\ S\ b \bullet]$
8	$[S' \to \bullet\ S\ \$]\ [S \to \bullet]$	ϵ	$[S' \to S \bullet\ \$]$
9	$[S' \to \bullet\ S\ \$]\ [S \to a\ S\ b \bullet]$	ϵ	$[S' \to S \bullet\ \$]$

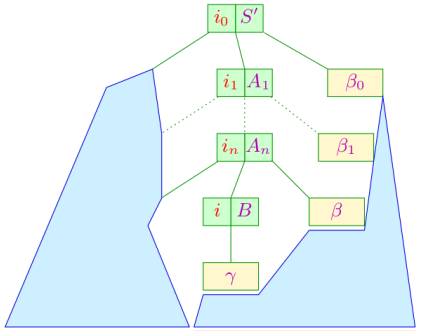
Conflicts arise between transations $(0, 1)$ or $(3, 4)$ resp..

Item Pushdown Automaton as LL(1)-Parser

... in detail: $S' \to S\ \$ \qquad S \to \epsilon^{\,0} \mid a\ S\ b^{\,1}$



Item Pushdown Automaton as LL(1)-Parser



$w \in \mathrm{First}_1($ $)$

$w \in \mathrm{First}_1(\mathrm{First}_1(\gamma) \odot_1 \mathrm{First}_1(\beta) \odot_1 \ldots \odot_1 \mathrm{First}_1(\beta_0))$

$w \in \mathrm{First}_1(\gamma) \odot_1 \mathrm{Follow}_1(B)$

Item Pushdown Automaton as LL(1)-Parser

Is G an $LL(1)$-grammar, we can index a lookahead-table with items and nonterminals:

LL(1)-Lookahead Table

We set $M[B,\ w]\ =\ i$ with $B \to \gamma\ ^{i}$ if $w \in \mathrm{First}_1(\gamma)\ \odot_1\ \mathrm{Follow}_1(B)$

... for example: $S' \to S\ \$$ $S \to \epsilon\ ^{0}$ | $a\ S\ b\ ^{1}$

Item Pushdown Automaton as LL(1)-Parser

For example: $S' \to S\ \$$ $S \to \epsilon\ ^{0}$ | $a\ S\ b\ ^{1}$

The transitions of the according Item Pushdown Automaton:

0	$[S' \to \bullet\ S\ \$]$	ϵ	$[S' \to \bullet\ S\ \$]\ [S \to \bullet]$
1	$[S' \to \bullet\ S\ \$]$	ϵ	$[S' \to \bullet\ S\ \$]\ [S \to \bullet\ a\ S\ b]$
2	$[S \to \bullet\ a\ S\ b]$	a	$[S \to a \bullet\ S\ b]$
3	$[S \to a \bullet\ S\ b]$	ϵ	$[S \to a \bullet\ S\ b]\ [S \to \bullet]$
4	$[S \to a \bullet\ S\ b]$	ϵ	$[S \to a \bullet\ S\ b]\ [S \to \bullet\ a\ S\ b]$
5	$[S \to a \bullet\ S\ b]\ [S \to \bullet]$	ϵ	$[S \to a\ S \bullet\ b]$
6	$[S \to a \bullet\ S\ b]\ [S \to a\ S\ b\bullet]$	ϵ	$[S \to a\ S \bullet\ b]$
7	$[S \to a\ S \bullet\ b]$	b	$[S \to a\ S\ b\bullet]$
8	$[S' \to \bullet\ S\ \$]\ [S \to \bullet]$	ϵ	$[S' \to S \bullet\ \$]$
9	$[S' \to \bullet\ S\ \$]\ [S \to a\ S\ b\bullet]$	ϵ	$[S' \to S \bullet\ \$]$

Lookahead table:

	$\$$	a	b
S	0	1	0

Left Recursion

Attention:

Many grammars are not $LL(k)$!

A reason for that is:

Definition

Grammar G is called left-recursive, if

$$A \to^{+} A\,\beta \qquad \text{for an} \quad A \in N,\ \beta \in (T \cup N)^{*}$$

Left Recursion

Theorem:

Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

Let wlog. $A \to A\,\beta \,|\, \alpha \quad \in P$
and A be reachable from S

Assumption: G is $LL(k)$

$\Rightarrow \text{First}_k(\alpha\,\beta^n\,\gamma) \cap \text{First}_k(\alpha\,\beta^{n+1}\,\gamma) = \emptyset$

Case 1: $\beta \to^* \epsilon$ — Contradiction !!!
Case 2: $\beta \to^* w \neq \epsilon \Longrightarrow \text{First}_k(\alpha\,w^k\,\gamma) \cap \text{First}_k(\alpha\,w^{k+1}\,\gamma) \neq \emptyset$

Left Recursion

Theorem:

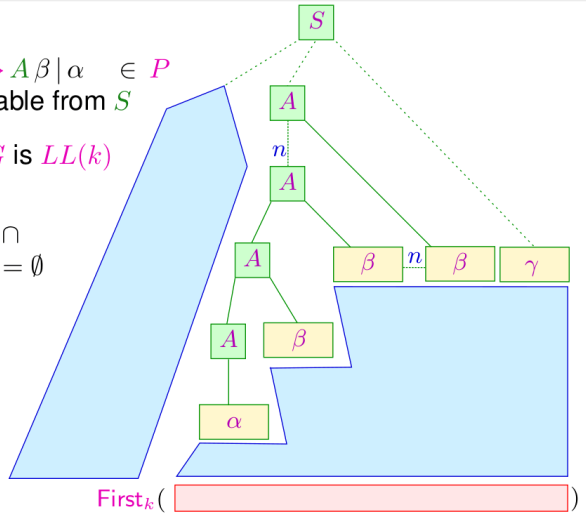
Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

Let wlog. $A \to A\,\beta \,|\, \alpha \quad \in P$
and A be reachable from S

Assumption: G is $LL(k)$

$\Rightarrow \text{First}_k(\alpha\,\beta^n\,\gamma) \cap \text{First}_k(\alpha\,\beta^{n+1}\,\gamma) = \emptyset$



Left Recursion

Theorem:

Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

Let wlog. $A \to A\,\beta \,|\, \alpha \quad \in P$
and A be reachable from S

Assumption: G is $LL(k)$

$\Rightarrow \text{First}_k(\alpha\,\beta^n\,\gamma) \cap \text{First}_k(\alpha\,\beta^{n+1}\,\gamma) = \emptyset$

Case 1: $\beta \to^* \epsilon$ — Contradiction !!!
Case 2: $\beta \to^* w \neq \epsilon \Longrightarrow \text{First}_k(\alpha\,w^k\,\gamma) \cap \text{First}_k(\alpha\,w^{k+1}\,\gamma) \neq \emptyset$

Left Recursion

Theorem:

Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

Let wlog. $A \to A\,\beta \,|\, \alpha \quad \in P$
and A be reachable from S

Assumption: G is $LL(k)$

$\Rightarrow \text{First}_k(\alpha\,\beta^n\,\gamma) \cap \text{First}_k(\alpha\,\beta^{n+1}\,\gamma) = \emptyset$

Case 1: $\beta \to^* \epsilon$ — Contradiction !!!
Case 2: $\beta \to^* w \neq \epsilon \Longrightarrow \text{First}_k(\alpha\,w^k\,\gamma) \cap \text{First}_k(\alpha\,w^{k+1}\,\gamma) \neq \emptyset$

Right-Regular Context-Free Parsing

Recurring scheme in programming languages: Lists of sth...

$S \to b \quad | \quad S\,a\,b$

Alternative idea: Regular Expressions

$S \to (\ b\ a\ )^*\ b$

Left Recursion

Theorem:

Let a grammar G be reduced and left-recursive, then G is not $LL(k)$ for any k.

Proof:

Let wlog. $A \to A\,\beta \,|\, \alpha \quad \in P$ and A be reachable from S

Assumption: G is $LL(k)$

$\Rightarrow \text{First}_k(\alpha\,\beta^n\,\gamma) \cap \text{First}_k(\alpha\,\beta^{n+1}\,\gamma) = \emptyset$