Script **generated by TTT**

Title: Petter: Compilerbau (18.05.2017)

Date: Thu May 18 14:16:50 CEST 2017

Duration: 99:49 min

Pages: 27

Example:

States: $0, 1, 2$
Start state: 0
Final states: $0, 2$

0	a	11
1	a	11
11	b	2
12	b	2

Example:

States: $0, 1, 2$
Start state: 0
Final states: $0, 2$

0	a	11
1	a	11
11	b	2
12	b	2

Conventions:

- We do not differentiate between pushdown symbols and states
- The rightmost / upper pushdown symbol represents the state
- Every transition consumes / modifies the upper part of the pushdown


Friedrich Bauer


Klaus Samelson

Definition: Pushdown Automaton

A pushdown automaton (PDA) is a tuple $M = (Q, T, \delta, q_0, F)$ with:

- Q a finite set of states;
- T an input alphabet;
- $q_0 \in Q$ the start state;
- $F \subseteq Q$ the set of final states and
- $\delta \subseteq Q^+ \times (T \cup \{\epsilon\}) \times Q^*$ a finite set of transitions

We define computations of pushdown automata with the help of transitions; a particular computation state (the current configuration) is a pair:

$$(\gamma, w) \in Q^* \times T^*$$

consisting of the pushdown content and the remaining input.

A computation step is characterized by the relation $\vdash \subseteq (Q^* \times T^*)^2$ with

$$(\alpha\,\gamma,\, x\,w) \vdash (\alpha\,\gamma',\, w) \quad \text{for} \quad (\gamma,\, x,\, \gamma') \in \delta$$

Remarks:

- The relation $\vdash$ depends on the pushdown automaton M
- The reflexive and transitive closure of $\vdash$ is denoted by $\vdash^*$
- Then, the language accepted by M is

$$\mathcal{L}(M) = \{w \in T^* \mid \exists\, f \in F : (q_0, w) \vdash^* (f, \epsilon)\}$$

Definition: Deterministic Pushdown Automaton

The pushdown automaton M is deterministic, if every configuration has maximally one successor configuration.

This is exactly the case if for distinct transitions (γ_1, x, γ_2), $(\gamma_1', x', \gamma_2') \in \delta$ we can assume:
Is γ_1 a suffix of γ_1', then $x \neq x' \wedge x \neq \epsilon \neq x'$ is valid.

... for example:

0	a	11
1	a	11
11	b	2
12	b	2

... this obviously holds

Pushdown Automata

M. Schützenberger A. Öttinger

Theorem:

For each context free grammar $G = (N, T, P, S)$ a pushdown automaton M with $\mathcal{L}(G) = \mathcal{L}(M)$ can be built.

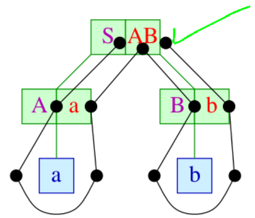
The theorem is so important for us, that we take a look at two constructions for automata, motivated by both of the special derivations:

- M_G^L to build Leftmost derivations
- M_G^R to build reverse Rightmost derivations

Item Pushdown Automaton – Example

Our example:

$$S \rightarrow AB \qquad A \rightarrow a \qquad B \rightarrow b$$



Item Pushdown Automaton – Example

We add another rule $S' \to S$ for initialising the construction:

Start state: $[S' \to \bullet\, S]$
End state: $[S' \to S\, \bullet]$
Transition relations:

$[S' \to \bullet\, S]$	ϵ	$[S' \to \bullet\, S]\,[S \to \bullet\, A\,B]$
$[S \to \bullet\, A\,B]$	ϵ	$[S \to \bullet\, A\,B]\,[A \to \bullet\, a]$
$[A \to \bullet\, a]$	a	$[A \to a\, \bullet]$
$[S \to \bullet\, A\,B]\,[A \to a\, \bullet]$	ϵ	$[S \to A\, \bullet\, B]$
$[S \to A\, \bullet\, B]$	ϵ	$[S \to A\, \bullet\, B]\,[B \to \bullet\, b]$
$[B \to \bullet\, b]$	b	$[B \to b\, \bullet]$
$[S \to A\, \bullet\, B]\,[B \to b\, \bullet]$	ϵ	$[S \to A\,B\, \bullet]$
$[S' \to \bullet\, S]\,[S \to A\,B\, \bullet]$	ϵ	$[S' \to S\, \bullet]$

Item Pushdown Automaton

The item pushdown automaton M_G^L has three kinds of transitions:

Expansions: $([A \to \alpha \bullet B\,\beta], \epsilon, [A \to \alpha \bullet B\,\beta]\,[B \to \bullet\,\gamma])$ for $A \to \alpha\,B\,\beta,\ B \to \gamma \in P$
Shifts: $([A \to \alpha \bullet a\,\beta], a, [A \to \alpha\,a \bullet \beta])$ for $A \to \alpha\,a\,\beta \in P$
Reduces: $([A \to \alpha \bullet B\,\beta]\,[B \to \gamma \bullet], \epsilon, [A \to \alpha\,B \bullet \beta])$ for $A \to \alpha\,B\,\beta,\ B \to \gamma \in P$

Items of the form: $[A \to \alpha \bullet]$ are also called complete
The item pushdown automaton shifts the bullet around the derivation tree ...

Item Pushdown Automaton

Discussion:

- The expansions of a computation form a leftmost derivation
- Unfortunately, the expansions are chosen nondeterministically
- For proving correctness of the construction, we show that for every Item $[A \to \alpha \bullet B\,\beta]$ the following holds:

$$([A \to \alpha \bullet B\,\beta],\ w) \vdash^* ([A \to \alpha\,B \bullet \beta],\ \epsilon) \quad \text{iff} \quad B \to^* w$$

- LL-Parsing is based on the item pushdown automaton and tries to make the expansions deterministic ...

Item Pushdown Automaton

The item pushdown automaton M_G^L has three kinds of transitions:

Expansions: $([A \to \alpha \bullet B\,\beta], \epsilon, [A \to \alpha \bullet B\,\beta]\,[B \to \bullet\,\gamma])$ for $A \to \alpha\,B\,\beta,\ B \to \gamma \in P$
Shifts: $([A \to \alpha \bullet a\,\beta], a, [A \to \alpha\,a \bullet \beta])$ for $A \to \alpha\,a\,\beta \in P$
Reduces: $([A \to \alpha \bullet B\,\beta]\,[B \to \gamma \bullet], \epsilon, [A \to \alpha\,B \bullet \beta])$ for $A \to \alpha\,B\,\beta,\ B \to \gamma \in P$

Items of the form: $[A \to \alpha \bullet]$ are also called complete
The item pushdown automaton shifts the bullet around the derivation tree ...

Item Pushdown Automaton

Discussion:

- The expansions of a computation form a leftmost derivation
- Unfortunately, the expansions are chosen nondeterministically

- For proving correctness of the construction, we show that for every Item $[A \to \alpha \bullet B\,\beta]$ the following holds:

$$([A \to \alpha \bullet B\,\beta],\ w) \vdash^* ([A \to \alpha\, B \bullet \beta],\ \epsilon) \qquad \text{iff} \qquad B \to^* w$$

- LL-Parsing is based on the item pushdown automaton and tries to make the expansions deterministic ...

Item Pushdown Automaton

Example: $S \to \epsilon \ \mid\ a\,S\,b$

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	$[S' \to S\bullet]$

Topdown Parsing

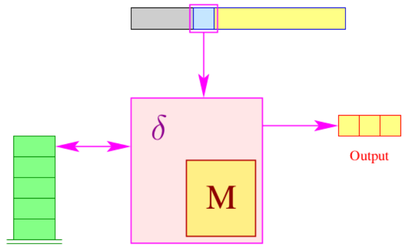
Problem:
Conflicts between the transitions prohibit an implementation of the item pushdown automaton as deterministic pushdown automaton.

Idea 1: GLL Parsing
For each conflict, we create a virtual copy of the complete stack and continue deriving 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 symbol.

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:

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

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
$a\,b$
$a\,a\,b\,b$
$a\,a\,a\,b\,b\,b$
$\ldots$

$\equiv$ the yield's prefix of length 1

Lookahead Sets

Arithmetics:

First$_1$(_) is compatible with union and concatenation:

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

$\odot$ being 1 – concatenation

Lookahead Sets

Arithmetics:

First$_1$(_) is compatible with union and concatenation:

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

$\odot$ being 1 – concatenation

Definition: 1-concatenation

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

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

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

Lookahead Sets

for example...

$$\begin{array}{lllll} E & \rightarrow & E+T & \mid & T \\ T & \rightarrow & T*F & \mid & F \\ F & \rightarrow & (E) & \mid & \text{name} \mid \text{int} \end{array}$$

with empty(E) = empty(T) = empty(F) = false

... we obtain:

$$\begin{array}{lll} F_\epsilon(S') \supseteq F_\epsilon(E) & & F_\epsilon(E) \supseteq F_\epsilon(E) \\ F_\epsilon(E) \supseteq F_\epsilon(T) & & F_\epsilon(T) \supseteq F_\epsilon(T) \\ F_\epsilon(T) \supseteq F_\epsilon(F) & & F_\epsilon(F) \supseteq \{\,(\,, \text{name}, \text{int}\} \end{array}$$

Fast Computation of Lookahead Sets

Observation:

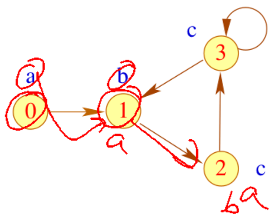
- The form of each inequality of these systems is:

  $x \sqsupseteq y$ resp. $x \sqsupseteq d$

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

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

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

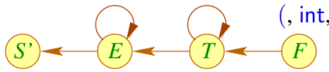


Fast Computation of Lookahead Sets

... for our example grammar:

abc

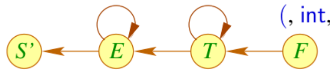
First_1 :



Fast Computation of Lookahead Sets

... for our example grammar:

First_1 :



Item Pushdown Automaton as LL(1)-Parser

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

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: $S \to \epsilon^{\,0} \mid a\,S\,b^{\,1}$

$\text{First}_1(\text{input})$	ϵ	a	b
S	?	?	?