**Script** generated by TTT

Title: Petter: Compilerbau (09.05.2019)

Date: Thu May 09 14:17:11 CEST 2019

Duration: 89:05 min

Pages: 22

## 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:** $0, 2$

| | | |
|---|---|---|
| $0$ | $a$ | $11$ |
| $1$ | $a$ | $11$ |
| $11$ | $b$ | $2$ |
| $12$ | $b$ | $2$ |

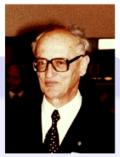
### Conventions:

- 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



### Definition: Pushdown Automaton


Friedrich Bauer


Klaus Samelson

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 $(\gamma_1, x, \gamma_2)\,,\ (\gamma_1', x', \gamma_2') \in \delta$ we can assume:
Is $\gamma_1$ a suffix of $\gamma_1'$, then $x \neq x' \ \wedge\ x \neq \epsilon \neq x'$ is valid.

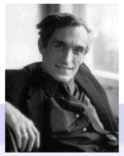
### ... for example:

| | | |
|---|---|---|
| $0$ | $a$ | $11$ |
| $1$ | $a$ | $11$ |
| $11$ | $b$ | $2$ |
| $12$ | $b$ | $2$ |

... 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:

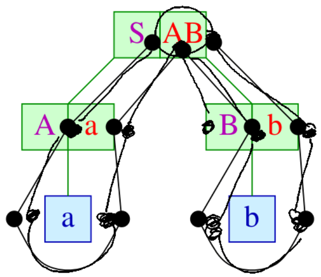
- $M_G^L$ to build Leftmost derivations
- $M_G^R$ to build reverse Rightmost derivations



## Item Pushdown Automaton – Example

### Our example:

$$S \to A\,B \qquad A \to a \qquad B \to b$$



## 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:**

| | | |
|---|---|---|
| $[S' \to \bullet\, S\,\$]$ | $\epsilon$ | $[S' \to \bullet\, S\,\$]\,[S \to \bullet\, A\,B]$ |
| $[S \to \bullet\, A\,B]$ | $\epsilon$ | $[S \to \bullet\, A\,B]\,[A \to \bullet\, a]$ |
| $[A \to \bullet\, a]$ | $a$ | $[A \to a\,\bullet]$ |
| $[S \to \bullet\, A\,B]\,[A \to a\,\bullet]$ | $\epsilon$ | $[S \to A\,\bullet\,B]$ |
| $[S \to A\,\bullet\,B]$ | $\epsilon$ | $[S \to A\,\bullet\,B]\,[B \to \bullet\, b]$ |
| $[B \to \bullet\, b]$ | $b$ | $[B \to b\,\bullet]$ |
| $[S \to A\,\bullet\,B]\,[B \to b\,\bullet]$ | $\epsilon$ | $[S \to A\,B\,\bullet]$ |
| $[S' \to \bullet\, S\,\$]\,[S \to A\,B\,\bullet]$ | $\epsilon$ | $[S' \to S\,\bullet\,\$]$ |



## 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 – 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:**

| | | |
|---|---|---|
| $[S' \to \bullet\, S\,\$]$ | $\epsilon$ | $[S' \to \bullet\, S\,\$]\,[S \to \bullet\, A\,B]$ |
| $[S \to \bullet\, A\,B]$ | $\epsilon$ | $[S \to \bullet\, A\,B]\,[A \to \bullet\, a]$ |
| $[A \to \bullet\, a]$ | $a$ | $[A \to a\,\bullet]$ |
| $[S \to \bullet\, A\,B]\,[A \to a\,\bullet]$ | $\epsilon$ | $[S \to A\,\bullet\,B]$ |
| $[S \to A\,\bullet\,B]$ | $\epsilon$ | $[S \to A\,\bullet\,B]\,[B \to \bullet\, b]$ |
| $[B \to \bullet\, b]$ | $b$ | $[B \to b\,\bullet]$ |
| $[S \to A\,\bullet\,B]\,[B \to b\,\bullet]$ | $\epsilon$ | $[S \to A\,B\,\bullet]$ |
| $[S' \to \bullet\, S\,\$]\,[S \to A\,B\,\bullet]$ | $\epsilon$ | $[S' \to S\,\bullet\,\$]$ |



## 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:

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

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



## Item Pushdown Automaton

### Example: $S' \to S\,\$ \qquad S \to \epsilon \mid a\, S\, b$

The transitions of the according Item Pushdown Automaton:

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



## Topdown Parsing

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

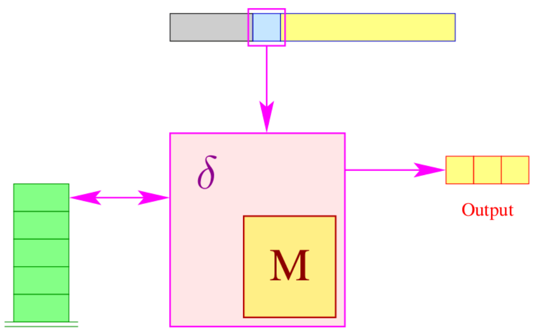
**Idea 1: GLL Parsing**
For each conflict, we create a virtual copy of the complete stack and continue deriving in parallel.

**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:

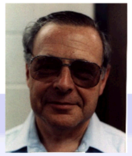
- Emanate from the item pushdown automaton
- Consider the next input symbol to determine the appropriate rule for the next expansion
- A grammar is called $LL(1)$ if a unique choice is always possible



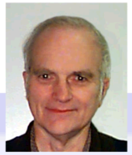
## Topdown Parsing

### 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:

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

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

### Example 2:

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

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



## Lookahead Sets

**Definition: $\text{First}_1$-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: $S \to \epsilon \ \ | \ \ a\, S\, b$

| $\text{First}_1(S)$ |
|---|
| $\epsilon$ |
| $a\, b$ |
| $a\, a\, b\, b$ |
| $a\, a\, a\, b\, b\, b$ |
| $\ldots$ |

## Lookahead Sets

### Arithmetics:

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

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

$\odot_1$ being $1-$concatenation



## Lookahead Sets

### Arithmetics:

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

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

$\odot_1$ 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 First$_1(A)$ are non-empty.



## Lookahead Sets

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

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

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

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