Syntactic Analysis

Chapter 2:
Basics of Pushdown Automata

Example:

<table>
<thead>
<tr>
<th>States:</th>
<th>0, 1, 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start state:</td>
<td>0</td>
</tr>
<tr>
<td>Final states:</td>
<td>0, 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>a</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>b</td>
<td>2</td>
</tr>
</tbody>
</table>

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
- $\delta \subseteq Q^+ \times (T \cup \{\varepsilon\}) \times Q^*$ a finite set of transitions

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
... for example:

| States:  | 0, 1, 2 | 0 | a | 11 |
| Start state: | 0 | 1 | a | 11 |
| Final states: | 0, 2 | 11 | b | 2 |
|       | 12 | b | 2 |

**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 $x \neq x'$ and $x \neq \epsilon \neq x'$ is valid.

---

**Pushdown Automata**

*CYK*

**Theorem:**

For each context free grammar $G = (N, T, P, S)$ a pushdown automaton $M$ with $L(G) = L(M)$ 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**

---

**Syntactic Analysis**

**Chapter 3:**

**Top-down Parsing**
Item Pushdown Automaton – Example

Our example:

\[ S \to AB \quad A \to a \quad B \to b \]

Item Pushdown Automaton

The item pushdown automaton \( M^L \) has three kinds of transitions:

**Expansions:**

\[
( [A \to \alpha \bullet B \beta], \epsilon, [A \to \alpha \bullet B \beta] \cdot [B \to \bullet \gamma]) \quad \text{for} \quad A \to \alpha B \beta, \ B \to \gamma \in P
\]

**Shifts:**

\[
( [A \to \alpha \bullet a \beta], [A \to \alpha a \bullet \beta]) \quad \text{for} \quad A \to \alpha a \beta \in P
\]

**Reduces:**

\[
( [A \to \alpha \bullet B \beta], B \to \gamma \bullet \epsilon, [A \to \alpha B \gamma \bullet \beta]) \quad \text{for} \quad A \to \alpha B \beta, \ B \to \gamma \in P
\]

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:**

Start state: \( [S' \to \bullet S \ ] \)

End state: \( [S' \to S \bullet \ ] \)

Transition relations:

\[
\begin{align*}
[S' \to \bullet S \ ] & \quad \epsilon \quad [S' \to \bullet S \ ] \quad [S \to \bullet AB] \\
[S \to \bullet AB] & \quad \epsilon \quad [S' \to \bullet S \ ] \quad [A \to \bullet a] \\
[A \to \bullet a] & \quad a \quad A \to \bullet a \\
[S \to \bullet AB] & \quad \epsilon \quad [S \to \bullet AB] \quad [A \to \bullet a] \\
[S \to \bullet AB] & \quad \epsilon \quad [S \to \bullet AB] \quad [B \to \bullet b] \\
[B \to \bullet b] & \quad b \quad B \to \bullet b \\
[S' \to \bullet S \ ] & \quad \epsilon \quad [S \to \bullet AB] \\
[S \to \bullet AB] & \quad \epsilon \quad [S' \to \bullet S \ ]
\end{align*}
\]

---

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**

**Example:**  
\[ S' \rightarrow S \$
\]
\[ S \rightarrow e | aSb \]

The transitions of the according Item Pushdown Automaton:

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[ S' \rightarrow \bullet S $ ]</td>
<td>ε</td>
</tr>
<tr>
<td>1</td>
<td>[ S' \rightarrow \bullet S $ ]</td>
<td>ε</td>
</tr>
<tr>
<td>2</td>
<td>[ S \rightarrow \bullet aSb ]</td>
<td>a</td>
</tr>
<tr>
<td>3</td>
<td>[ S \rightarrow \bullet aSb ]</td>
<td>ε</td>
</tr>
<tr>
<td>4</td>
<td>[ S \rightarrow \bullet aSb ]</td>
<td>ε</td>
</tr>
<tr>
<td>5</td>
<td>[ S \rightarrow \bullet aSb ]</td>
<td>ε</td>
</tr>
<tr>
<td>6</td>
<td>[ S \rightarrow aSb ]</td>
<td>ε</td>
</tr>
<tr>
<td>7</td>
<td>[ S \rightarrow aSb ]</td>
<td>b</td>
</tr>
<tr>
<td>8</td>
<td>[ S' \rightarrow \bullet S $ ]</td>
<td>ε</td>
</tr>
<tr>
<td>9</td>
<td>[ S' \rightarrow \bullet S $ ]</td>
<td>ε</td>
</tr>
</tbody>
</table>

**Structure of the \textit{LL(1)}-Parser:**

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

**Topdown Parsing**

**Problem:**

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

**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 \textit{LL(1)} if a unique choice is always possible

**Definition:**

A reduced grammar is called \textit{LL(1)}, if for each two distinct rules \( A \rightarrow \alpha, A \rightarrow \alpha' \in P \) and each derivation \( S \rightarrow_1^* u A [ \beta ] \) with \( u \in T^* \) the following is valid:

\[ \text{First}_1(\alpha \beta) \cap \text{First}_1(\alpha' \beta) = \emptyset \]
Topdown Parsing

Example 1:

\[
S \rightarrow \text{if ( E ) S else S} \mid E; \quad E \rightarrow \text{id}
\]

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

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:

\[
S \rightarrow \epsilon \mid a S b
\]

<table>
<thead>
<tr>
<th>First(_1(S))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\epsilon)</td>
</tr>
<tr>
<td>(a b)</td>
</tr>
<tr>
<td>(a a b b)</td>
</tr>
<tr>
<td>(a a a b b b)</td>
</tr>
<tr>
<td>\ldots</td>
</tr>
</tbody>
</table>