Introduction to the Theory of Computation

Lecture 16: Push-down automata

Tiago Cogumbreiro
Today we will learn...

- Pushdown automata (PDA)
- Formalizing PDAs
- Union of PDAs
- Examples

Section 2.2
Intuition

Define an automata family $\leftrightarrow$ CFG
NFA recap

Each transition performs one input operations: read/skip an input

Examples

- **Read one input:** $q_1 \xrightarrow{a} q_2$
- **Skip one input:** $q_1 \xrightarrow{\epsilon} q_2$
Nondeterministic **PushDown Automata (PDA)**

- Extend NFAs with an *unbounded stack*
- Recognizes the same language as CFGs

**PDA Execution**

Each transition:
- input op, pre-stack op, post-stack op
- Format: $q \xrightarrow{\text{input op, pre-stack op, post-stack op}} q'$

**Example**

$q_a \xrightarrow{\text{READ } a, \text{SKIP} \rightarrow \text{PUSH } a} q_a$

**Possible operations**

<table>
<thead>
<tr>
<th>$\text{INPUT}$</th>
<th>$\text{PRE}$</th>
<th>$\text{POST}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ $n$</td>
<td>POP $n$</td>
<td>PUSH $n$</td>
</tr>
<tr>
<td>SKIP ($\epsilon$)</td>
<td>SKIP</td>
<td>SKIP</td>
</tr>
</tbody>
</table>
Nondeterministic PushDown Automata (PDA)

- Extend NFAs with an *unbounded stack*
- Recognizes the same language as CFGs

### PDA Execution

Each transition:
- **input op, pre-stack op, post-stack op**
- Format: $q \xrightarrow{\text{READ, POP, PUSH}} q'$

### Possible operations

<table>
<thead>
<tr>
<th>$\text{INPUT}$</th>
<th>$\text{PRE}$</th>
<th>$\text{POST}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ $n$</td>
<td>POP $n$</td>
<td>PUSH $n$</td>
</tr>
<tr>
<td>SKIP ($\epsilon$)</td>
<td>SKIP</td>
<td>SKIP</td>
</tr>
</tbody>
</table>

### Example

$q_a \xrightarrow{\text{READ }a, \text{SKIP} \rightarrow \text{PUSH }a} q_a$

### Attention!

The comma does not denote parallel edges. Instead, we stack multiple transitions **vertically**.
PDA example (intuition)

Give a PDA that recognizes \( \{ a^n b^n \mid n \geq 0 \} \)

1. \( q_{\text{init}} \) \( \xrightarrow{\text{SKIP,SKIP} \rightarrow \text{PUSH EMPTY?}} \) \( q_a \)
2. \( q_a \) \( \xrightarrow{\text{READ } a, \text{SKIP} \rightarrow \text{PUSH } a} \) \( q_a \)
3. \( q_a \) \( \xrightarrow{\text{SKIP,SKIP} \rightarrow \text{SKIP}} \) \( q_b \)
4. \( q_a \) \( \xrightarrow{\text{READ } b, \text{POP } a \rightarrow \text{SKIP}} \) \( q_b \)
5. \( q_b \) \( \xrightarrow{\text{SKIP,EMPTY?} \rightarrow \text{SKIP}} \) \( q_F \)
Exercising transitions
Writing transitions

Possible operations

<table>
<thead>
<tr>
<th>$INPUT</th>
<th>$PRE</th>
<th>$POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ $n$</td>
<td>POP $n$</td>
<td>PUSH $n$</td>
</tr>
<tr>
<td>SKIP ($\epsilon$)</td>
<td>SKIP</td>
<td>SKIP</td>
</tr>
</tbody>
</table>

Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel EMPTY?):
Writing transitions

Possible operations

<table>
<thead>
<tr>
<th>$INPUT$</th>
<th>$PRE$</th>
<th>$POST$</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ $n$</td>
<td>POP $n$</td>
<td>PUSH $n$</td>
</tr>
<tr>
<td>SKIP ($\varepsilon$)</td>
<td>SKIP</td>
<td>SKIP</td>
</tr>
</tbody>
</table>

Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel $\text{EMPTY}$):
   
   $$\text{READ 0, EMPTY?} \rightarrow \text{SKIP}$$

2. Test if stack is empty:
### Writing transitions

#### Possible operations

<table>
<thead>
<tr>
<th>$\text{INPUT}$</th>
<th>$\text{PRE}$</th>
<th>$\text{POST}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ $n$</td>
<td>POP $n$</td>
<td>PUSH $n$</td>
</tr>
<tr>
<td>SKIP ($\epsilon$)</td>
<td>SKIP</td>
<td>SKIP</td>
</tr>
</tbody>
</table>

#### Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel EMPTY?):
   
   \[
   \text{READ 0, EMPTY?} \rightarrow \text{SKIP}
   \]

2. Test if stack is empty:
   
   \[
   \text{SKIP, EMPTY?} \rightarrow \text{SKIP}
   \]

3. Test if a is on top and leave stack untouched:
Writing transitions

Possible operations

<table>
<thead>
<tr>
<th>$INPUT</th>
<th>$PRE</th>
<th>$POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ $n$</td>
<td>POP $n$</td>
<td>PUSH $n$</td>
</tr>
<tr>
<td>SKIP ($\epsilon$)</td>
<td>SKIP</td>
<td>SKIP</td>
</tr>
</tbody>
</table>

Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel EMPTY?):
   
   \[
   \text{READ 0, EMPTY? } \rightarrow \text{ SKIP}
   \]

2. Test if stack is empty:
   
   \[
   \text{SKIP, EMPTY? } \rightarrow \text{ SKIP}
   \]

3. Test if a is on top and leave stack untouched:
   
   \[
   \text{SKIP, POP a } \rightarrow \text{ PUSH a}
   \]

4. Read b and leave stack untouched:
Writing transitions

Possible operations

<table>
<thead>
<tr>
<th>$INPUT$</th>
<th>$PRE$</th>
<th>$POST$</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ $n$</td>
<td>POP $n$</td>
<td>PUSH $n$</td>
</tr>
<tr>
<td>SKIP ($\epsilon$)</td>
<td>SKIP</td>
<td>SKIP</td>
</tr>
</tbody>
</table>

Exercises

1. Test if read 0 and stack is empty (assuming we initialize the stack with a sentinel EMPTY?):
   \[
   \text{READ } 0, \text{EMPTY?} \rightarrow \text{SKIP}
   \]

2. Test if stack is empty:
   \[
   \text{SKIP,EMPTY?} \rightarrow \text{SKIP}
   \]

3. Test if a is on top and leave stack untouched:
   \[
   \text{SKIP,PUSH } a \rightarrow \text{PUSH } a
   \]

4. Read b and leave stack untouched:
   \[
   \text{READ } b, \text{SKIP} \rightarrow \text{SKIP}
   \]
Simplifying the notation
Simplifying the notation

We can replace SKIP by $\epsilon$
Simplifying the notation

We can replace \texttt{SKIP} by $\epsilon$
Simplifying the notation

We can replace `SKIP` by $\epsilon$

We can omit `READ`
Simplifying the notation

We can replace \text{SKIP} by \(\epsilon\)

We can omit \text{READ}

Since read always appears in the same position, we can omit it, as we do in regular DFAs/NFAs.
Simplifying the notation

We can omit \text{PUSH/POP}
Simplifying the notation

We can omit \textbf{PUSH/POP}

We can replace sentinel \texttt{EMPTY?} by $\$$

Since push/pop always appear in the same position, we can omit them.
Since push/pop always appear in the same position, we can omit them.

We can omit PUSH/POP

We can replace sentinel EMPTY? by $.
Exercising transitions

(abbreviated notation)
Writing transitions

Possible operations

<table>
<thead>
<tr>
<th>$INPUT</th>
<th>$PRE</th>
<th>$POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ ($n$)</td>
<td>POP ($n$)</td>
<td>PUSH ($n$)</td>
</tr>
<tr>
<td>SKIP ($\epsilon$)</td>
<td>SKIP ($\epsilon$)</td>
<td>SKIP ($\epsilon$)</td>
</tr>
</tbody>
</table>

Exercises

1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $\$$)
Writing transitions

Possible operations

<table>
<thead>
<tr>
<th>$INPUT</th>
<th>$PRE</th>
<th>$POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ ((n))</td>
<td>POP ((n))</td>
<td>PUSH ((n))</td>
</tr>
<tr>
<td>SKIP ((\epsilon))</td>
<td>SKIP ((\epsilon))</td>
<td>SKIP ((\epsilon))</td>
</tr>
</tbody>
</table>

Exercises

1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $)$
   \[0, \text{$$} \rightarrow \text{$$}\]

2. Test if stack is empty while leaving the stack unchanged (assume sentinel $)$
Writing transitions

Possible operations

<table>
<thead>
<tr>
<th>INPUT</th>
<th>PRE</th>
<th>POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ (n)</td>
<td>POP (n)</td>
<td>PUSH (n)</td>
</tr>
<tr>
<td>SKIP (ε)</td>
<td>SKIP (ε)</td>
<td>SKIP (ε)</td>
</tr>
</tbody>
</table>

Exercises

1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $)
   
   $0, \$ \rightarrow \$

2. Test if stack is empty while leaving the stack unchanged (assume sentinel $)
   
   $\epsilon, \$ \rightarrow \$

3. Test if 0 is on top of the stack and replace it by 1:
Writing transitions

Possible operations

<table>
<thead>
<tr>
<th>$INPUT$</th>
<th>$PRE$</th>
<th>$POST$</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ ($n$)</td>
<td>POP ($n$)</td>
<td>PUSH ($n$)</td>
</tr>
<tr>
<td>SKIP ($\epsilon$)</td>
<td>SKIP ($\epsilon$)</td>
<td>SKIP ($\epsilon$)</td>
</tr>
</tbody>
</table>

Exercises

1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $\$$)
   
   $0, \$$ \rightarrow \$$

2. Test if stack is empty while leaving the stack unchanged (assume sentinel $\$$)
   
   $\epsilon, \$$ \rightarrow \$$

3. Test if 0 is on top of the stack and replace it by 1:
   
   $\epsilon, 0 \rightarrow 1$

4. Read 2, leave stack untouched
Writing transitions

Possible operations

<table>
<thead>
<tr>
<th>$INPUT$</th>
<th>$PRE$</th>
<th>$POST$</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ ($n$)</td>
<td>POP ($n$)</td>
<td>PUSH ($n$)</td>
</tr>
<tr>
<td>SKIP ($\epsilon$)</td>
<td>SKIP ($\epsilon$)</td>
<td>SKIP ($\epsilon$)</td>
</tr>
</tbody>
</table>

Exercises

1. Test if read 0 and stack is empty, leaving stack unchanged (assume a sentinel $\$$)
   \[ 0, \$$ \rightarrow \$$ \]

2. Test if stack is empty while leaving the stack unchanged (assume sentinel $\$$)
   \[ \epsilon, \$$ \rightarrow \$$ \]

3. Test if 0 is on top of the stack and replace it by 1:
   \[ \epsilon, 0 \rightarrow 1 \]

4. Read 2, leave stack untouched
   \[ 2, \epsilon \rightarrow \epsilon \]
Acceptance example
Acceptance example

Accepting $[\varepsilon aabb]$
Acceptance example

Accepting \([aabb]\)
Acceptance example

Accepting $[a\epsilon a bb]$
Acceptance example

Accepting \([aabb]\)
Acceptance example

Accepting \([aa\epsilon bb]\)
Acceptance example

Accepting [aa\textit{bb}]

\[
\begin{array}{c}
\text{start} \rightarrow q_{init} \\
\epsilon, \epsilon \rightarrow \$
\end{array}
\quad
\begin{array}{c}
q_a \\
a, \epsilon \rightarrow a
\end{array}
\quad
\begin{array}{c}
q_b \\
b, a \rightarrow \epsilon
\end{array}
\quad
\begin{array}{c}
q_F \\
\epsilon, \$ \rightarrow \epsilon
\end{array}
\]
Acceptance example

Accepting \([aab\varepsilon b]\)
Acceptance example

Accepting [aab\textit{b}]
Acceptance example

Accepting \([\text{aabb}\varepsilon]\)
Acceptance example

Accepting: bb

![Diagram of a push-down automaton with transitions (a, ε → a), (b, a → ε), (ε, ε → $), (ε, ε → ε), (ε, $ → ε)]
Acceptance example

Accepting: bb
Acceptance example

Accepting: $\epsilon$

Diagram:

- Start state: $q_{init}$
- Transitions:
  - $\epsilon, \epsilon \rightarrow \$ to $q_{a}$
  - $a, \epsilon \rightarrow a$ from $q_{a}$
  - $b, a \rightarrow \epsilon$ from $q_{a}$
  - $\epsilon, \epsilon \rightarrow \epsilon$ from $q_{a}$
  - $\epsilon, \$ \rightarrow \epsilon$ from $q_{b}$
- Accepting state: $q_{F}$
Acceptance example

Accepting: $\epsilon$

$$\begin{align*}
\text{start} & \rightarrow q_{\text{init}} \quad \epsilon, \epsilon \rightarrow \$ \\
q_{\text{init}} & \rightarrow q_a \quad a, \epsilon \rightarrow a \quad \epsilon, \epsilon \rightarrow \epsilon \\
q_a & \rightarrow q_b \quad b, a \rightarrow \epsilon \quad \epsilon, \$ \rightarrow \epsilon \\
q_b & \rightarrow q_F
\end{align*}$$

$$\begin{align*}
q_{\text{init}} & \rightarrow q_a \quad \epsilon \\
q_a & \rightarrow \$ \quad \epsilon \\
q_b & \rightarrow \$ \quad \epsilon \\
q_F & \rightarrow q_F
\end{align*}$$
Formalizing a PDA
Formalizing a PDA

Definition 2.13

A pushdown automaton is a 6-tuple \((Q, \Sigma, \Gamma, \delta, q_0, F)\) where

1. \(Q\) is a finite set called **states**
2. \(\Sigma\) is a finite set called **input alphabet**
3. \(\Gamma\) is a finite set called **stack alphabet**
4. \(\delta: Q \times \Sigma \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma)\) is the **transition function**
5. \(q_0 \in Q\) is the **start state**
6. \(F \subseteq Q\) is the set of **accepted states**
Let \((Q, \Sigma, \Gamma, \delta, q_1, \{q_F\})\) be defined as:

1. \(Q = \{q_{init}, q_a, q_b, q_F\}\)
2. \(\Sigma = \{a, b\}\)
3. \(\Gamma = \{a, \$\}\)

where \(\delta\) is defined by branches

\[
\begin{align*}
\delta(q_{init}, \epsilon, \epsilon) &= \{(q_a, \$)\} \\
\delta(q_a, a, \epsilon) &= \{(q_a, a)\} \\
\delta(q_a, \epsilon, \epsilon) &= \{(q_b, \epsilon)\} \\
\delta(q_b, b, a) &= \{(q_b, \epsilon)\} \\
\delta(q_b, \epsilon, \$) &= \{(q_F, \$)\} \\
\delta(q, c, s) &= \{\} \quad \text{otherwise}
\end{align*}
\]
Exercise
Give a PDA for the following grammar

Balanced parenthesis

\[ C \rightarrow \epsilon \mid Cc \mid CC \]
Give a PDA for the following grammar

Balanced parenthesis

\[ C \rightarrow o \ C \ c \mid CC \mid \epsilon \]
Acceptance

Acceptance: 0C
Acceptance

Acceptance: OC
Acceptance

Acceptance: $\epsilon$
Acceptance

Acceptance: $\epsilon$

CS420  ♦  Push-down automata  ♦  Lecture 16  ♦  Tiago Cogumbleiro
Acceptance

Acceptance: OOCOCC
Acceptance

Acceptance: 00C0CC
Formalization
Formalizing stack operation

Let $S(o_1, o_2, s)$ be defined as follows, where $S : \Gamma_\epsilon \times \Gamma_\epsilon \times \text{Stack}(\Gamma) \rightarrow \text{Stack}(\Gamma)$ and $\text{Stack}(\Gamma) = \text{List}(\Gamma)$:

**Pop operation**

\[
S(\epsilon) = s \\
S(\Gamma \cdot n) = S(n \cdot s)
\]

**Push operation**

\[
S(\epsilon) = s \\
S(\Gamma \cdot n) = S(n \cdot s)
\]

**Examples**

- $[0, 1] \triangleright \epsilon = [0, 1]$
- $[0, 1] \triangleright \$ \text{ is undefined!}$
- $[0, 1] \triangleright 0 = [1]$
- $[0, 1] \triangleright 1 \text{ is undefined!}$

- $[0, 1] \triangleleft \epsilon = [0, 1]$
- $[0, 1] \triangleleft \$ = [0, 1, \$]
- $[\cdot] \triangleleft \$ = [\$]
- $[0, 1] \triangleleft 0 = [0, 0, 1]$
- $[0, 1] \triangleleft 1 = [1, 0, 1]$
Stack operation exercises

Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ab ▹ c =$</td>
<td></td>
</tr>
<tr>
<td>$ab ▲ c =$</td>
<td></td>
</tr>
<tr>
<td>$ab ▹ a =$</td>
<td></td>
</tr>
<tr>
<td>$ab ▲ a =$</td>
<td></td>
</tr>
<tr>
<td>$ab ▹$</td>
<td></td>
</tr>
<tr>
<td>$ab ▲$</td>
<td></td>
</tr>
<tr>
<td>$\epsilon ▹$</td>
<td></td>
</tr>
<tr>
<td>$\epsilon ▲$</td>
<td></td>
</tr>
<tr>
<td>$\epsilon ▹ a$</td>
<td></td>
</tr>
<tr>
<td>$\epsilon ▲ a$</td>
<td></td>
</tr>
</tbody>
</table>
## Stack operation exercises

### Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ab \triangleright c = a\ b \triangleright c = cab$</td>
<td>$ab \triangleright a = b$</td>
</tr>
<tr>
<td>$ab \triangleleft a = aab$</td>
<td>$ab \triangleright $ = undef$</td>
</tr>
<tr>
<td>$ab \triangleleft $ = $ab$ $\epsilon \triangleright $ = undef$</td>
<td>$\epsilon \triangleleft $ = $</td>
</tr>
<tr>
<td>$\epsilon \triangleright a = undef$</td>
<td>$\epsilon \triangleleft a = a$</td>
</tr>
</tbody>
</table>
Formalizing acceptance

Rule 0. We can go from state $q$ and stack $s$ into state $q'$ and stack $s'$ with input $y \in \Sigma$ if we can construct $s'$ from a push $o$ and a pop $o'$ on stack $s$.

$$
(q', o') \in \delta(q, y, o) \\
(q, s) \xrightarrow{y,o} (q', s \uparrow o \downarrow o')
$$

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, let the steps through relation, notation $q \leadsto_M w$, be defined as:

Rule 1. State $q$ steps through $\square$ if $q$ is a final state.

Rule 2. If we can go from $q$ to $q'$ with $y$ and $q'$ steps through $w$, then $q$ steps through $y \cdot w$.

Acceptance. We say that $M$ accepts $w$ if, and only if, $q_0, \square \leadsto_M w$. 
Example of acceptance

We can build a chain of states as follows

\[(q_{init}, []) \xrightarrow{\epsilon, \epsilon} (q_a, [$]) \xrightarrow{a, \epsilon} (q_a, [a, \$]) \xrightarrow{a, \epsilon} (q_b, [a, a, \$]) \xrightarrow{b, a} (q_b, [a, a, \$]) \xrightarrow{b, a} (q_b, [a, \$]) \xrightarrow{\epsilon, \$} (q_F, [])\]

Since \(q_F\) is a final state, we have that

\[(q_{init}, []) \sim [a, a, b, b]\]

Recall
Example 2.16
Example 2.16

A sequence of a-s then b-s and finally c-s with as many a-s as there are b-s or as there are c-s.

\[ \{ a^i b^j c^k \mid i = j \lor i = k \} \]
Example 2.16

A sequence of a-s then b-s and finally c-s with as many a-s as there are b-s or as there are c-s.

\( \{a^i b^j c^k \mid i = j \lor i = k\} \)

A solution

Step 1. read and push a total of \( N \) a's.

Step 2. Either:

- \((i = j)\) read \( N \) b's and pop a's; followed by reading an arbitrary number of c's
- \((i = k)\) read an arbitrary number of b's followed by read \( N \) c's and pop a's
State diagram of Example 2.16
Example 2.16 accept $[a, a, b, b, c, c]$?
Example 2.16 accept $[a, a, b, b, c, c]$?
Example 2.16 accept \([a, a, b, c, c]\)?
Example 2.16 accept $[a, a, b, c, c]$?
Example 2.16 accept $[a, a, b, b, c]$?
Example 2.16 accept $[a, a, b, b, c]$?
Example 2.16 rejects \([a, a, b, b, b, c]\)?
Example 2.16 rejects $[a, a, b, b, b, c]$?
Union for PDAs?
Example 2.16

\[ \{a^i b^j c^k \mid i = j \lor i = k\} = \{a^i b^j c^k \mid i = j\} \cup \{a^i b^j c^k \mid i = k\} \]
Example 2.16

\[ \{ a^i b^j c^k \mid i = j \lor i = k \} = \{ a^i b^j c^k \mid i = j \} \cup \{ a^i b^j c^k \mid i = k \} \]