CS420

Introduction to the Theory of Computation

Lecture 17: 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 $\iff$ CFG
NFA recap

Each transition performs one input operation: 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**

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

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

### 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>

### 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. \( q_{\text{init}} \) \( \xrightarrow{\text{SKIP,SKIP} \to \text{PUSH EMPTY?}} \) \( q_a \)
2. \( q_a \) \( \xrightarrow{\text{READ a,SKIP} \to \text{PUSH a}} \) \( q_a \)
3. \( q_a \) \( \xrightarrow{\text{SKIP,SKIP} \to \text{SKIP}} \) \( q_b \)
4. \( q_b \) \( \xrightarrow{\text{READ b,POP a} \to \text{SKIP}} \) \( q_b \)
5. \( q_b \) \( \xrightarrow{\text{SKIP,EMPTY?} \to \text{SKIP}} \) \( q_F \)

start \( q_{\text{init}} \) \( \xrightarrow{\text{SKIP, SKIP} \to \text{PUSH EMPTY?}} \) \( q_a \) \( \xrightarrow{\text{SKIP, SKIP} \to \text{SKIP}} \) \( q_b \) \( \xrightarrow{\text{SKIP, POP EMPTY?} \to \text{SKIP}} \) \( q_F \)
Exercising transitions
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?):
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?):

   READ 0, EMPTY? $\rightarrow$ SKIP

2. Test if stack is empty:
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</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?):
   - READ 0, EMPTY? $\rightarrow$ SKIP

2. Test if stack is empty:
   - SKIP, EMPTY? $\rightarrow$ 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?):
   - $\text{READ 0, EMPTY? } \rightarrow \text{ SKIP}$

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:
   - $\text{READ b, SKIP } \rightarrow \text{ SKIP}$
Simplifying the notation
Simplifying the notation

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

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

We can replace **SKIP** by $\epsilon$

We can omit **READ**
Simplifying the notation

We can replace SKIP by $\epsilon$

We can omit 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 PUSH/POP
Simplifying the notation

We can omit PUSH/POP

We can replace sentinel EMPTY? by a character $\notin \Gamma$

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

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

We can omit **PUSH/POP**

We can replace sentinel **EMPTY?** by a character $\notin \Gamma$

Since empty? always appear in the same position.
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, \$ \rightarrow \$

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 ($\epsilon$)</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:
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
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 $[\epsilon aabb]$
Acceptance example

Accepting [aabb]
Acceptance example

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

Accepting \([aabb]\)
Acceptance example

Accepting $[aaεbb]$
Acceptance example

Accepting $[aabb]$
Acceptance example

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

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

Accepting \([aabb\epsilon]\)
Acceptance example

Accepting: bb
Acceptance example

Accepting: bb
Acceptance example

Accepting: \(\epsilon\)
Acceptance example

Accepting: $\epsilon$

---

$L_{Push-down}$ automata

Start

$q_{init}$

$a, \epsilon \rightarrow a$

$q_a$

$b, a \rightarrow \epsilon$

$q_b$

$q_f$

$\epsilon, \epsilon \rightarrow $

$q_{init}$

$\epsilon$

$q_a$

$\epsilon$

$q_b$

$\epsilon$

$q_f$
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. $\delta : Q \times \Sigma \epsilon \times \Gamma \epsilon \rightarrow \mathcal{P}(Q \times \Gamma \epsilon)$ is the transition function
5. $q_0 \in Q$ is the start state
6. $F \subseteq Q$ is the set of accepted states
Example

Let \((Q, \Sigma, \Gamma, \delta, q_{init}, \{q_F\})\) 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:

- \(\delta(q_{init}, \epsilon, \epsilon) = \{(q_a, \$)\}\)
- \(\delta(q_a, a, \epsilon) = \{(q_a, a)\}\)
- \(\delta(q_a, \epsilon, \epsilon) = \{(q_b, \epsilon)\}\)
- \(\delta(q_b, b, a) = \{(q_b, \epsilon)\}\)
- \(\delta(q_b, \epsilon, \$) = \{(q_F, \epsilon)\}\)
- \(\delta(q, c, s) = \{}\) otherwise
Exercise
Give a PDA for the following grammar

Balanced parenthesis

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

Balanced parentheses

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

Acceptance: OC
Acceptance

Acceptance: OC
Acceptance

Acceptance: $\epsilon$
Acceptance

Acceptance: $\epsilon$

start $\rightarrow q_1 \xrightarrow{\epsilon, \epsilon \rightarrow \$} q_2 \xrightarrow{\epsilon, \$ \rightarrow \epsilon} q_3$

$q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{\$} q_3$
Acceptance

Acceptance: OOCOCC
Acceptance

Acceptance: OOCOCC
Formalization
Formalizing stack operation

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

**Pop operation**

\[
\begin{align*}
    s \triangleright \varepsilon &= s \\
    n :: s \triangleright n &= s
\end{align*}
\]

**Push operation**

\[
\begin{align*}
    s \triangleleft \varepsilon &= s \\
    s \triangleleft n &= n :: s
\end{align*}
\]

**Examples**

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

**Examples**

- $[0, 1] \triangleleft \varepsilon = [0, 1]$
- $[0, 1] \triangleleft \$ = [0, 1, \$]$
- $[] \triangleleft \$ = [\$]$
- $[0, 1] \triangleleft 0 = [0, 0, 1]$
- $[0, 1] \triangleleft 1 = [1, 0, 1]$
## Stack operation exercises

### Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ab \triangleright c =$</td>
<td></td>
</tr>
<tr>
<td>$ab \triangleleft c =$</td>
<td></td>
</tr>
<tr>
<td>$ab \triangleright a =$</td>
<td></td>
</tr>
<tr>
<td>$ab \triangleleft a =$</td>
<td></td>
</tr>
<tr>
<td>$ab \triangleright $ =$</td>
<td></td>
</tr>
<tr>
<td>$ab \triangleleft $ =$</td>
<td></td>
</tr>
<tr>
<td>$\epsilon \triangleright $ =$</td>
<td></td>
</tr>
<tr>
<td>$\epsilon \triangleleft $ =$</td>
<td></td>
</tr>
<tr>
<td>$\epsilon \triangleright a =$</td>
<td></td>
</tr>
<tr>
<td>$\epsilon \triangleleft 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>$ab \triangleright c =$</td>
<td>undef</td>
</tr>
<tr>
<td>$ab \triangleleft c =$</td>
<td>$cab$</td>
</tr>
<tr>
<td>$ab \triangleright a =$</td>
<td>$b$</td>
</tr>
<tr>
<td>$ab \triangleleft a =$</td>
<td>$aab$</td>
</tr>
<tr>
<td>$ab \triangleright $ =</td>
<td>undef</td>
</tr>
<tr>
<td>$ab \triangleleft $ =</td>
<td>$ab$</td>
</tr>
<tr>
<td>$\epsilon \triangleright $ =</td>
<td>undef</td>
</tr>
<tr>
<td>$\epsilon \triangleleft $ =</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$\epsilon \triangleright a =$</td>
<td>undef</td>
</tr>
<tr>
<td>$\epsilon \triangleleft a =$</td>
<td>$a$</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 $y \in \Sigma_\epsilon$ if we can construct $s'$ from a push $o$ and a pop $o'$ on stack $s$.

$$ (q', o') \in \delta(q, y, o) \quad \frac{(q, s) \xrightarrow{y,o} (q', s \triangleright o \triangleleft o')}{(q, s) \sim_M (q', s') \xrightarrow{y} (q', s') \sim_M w} $$

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

**Rule 1.** State $q$ steps through $\Box$ 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, $q_0, \Box \sim_M w$. 
Example of acceptance

We can build a chain of states as follows

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

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

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