PUSH-DOWN MEMORY AND PUSH-DOWN AUTOMATA

Push-down (stack) memory:

- Top (most recent) item in memory; only \( c_k \) is accessible at this point

\[
\begin{array}{cccccc}
  c_1 & c_2 & c_2 & \cdots & c_{k-1} & c_k \\
\end{array}
\]

\[
\text{stack} = c_1c_2\cdots c_{k-1}c_k.
\]

- \( c_{k-1} \) is accessible only after the removal of \( c_k \)

\[
\begin{array}{cccc}
  c_1 & c_2 & c_2 & \cdots & c_{k-1} \\
\end{array}
\]

\[
\text{stack} = c_1c_2\cdots c_{k-1}
\]

- Addition of \( c_{k+1} \) makes \( c_k \) inaccessible

\[
\begin{array}{cccccc}
  c_1 & c_2 & c_2 & \cdots & c_{k-1} & c_k & c_{k+1} \\
\end{array}
\]

\[
\text{stack} = c_1c_2\cdots c_{k-1}c_kc_{k+1}
\]

Push-down automata = FSA + push-down memory.

- The stack-alphabet may be different from input alphabet.

Each transition does four things:

1. Reads an input-symbol (and moves the read-head right).
2. Looks at the top stack-symbol.
3. Replaces the top stack-symbol by zero or more symbols.
4. Changes state (perhaps back to the same state).
EXAMPLE OF A PDA

Example. A PDA for $L = \{a^n b^n : n \geq 1\}$.

- Save the $a$’s on the stack, then match them off with the $b$’s.

$$\lambda/a/a, a/a/aa \quad a/b/\lambda$$

$P_{a^n b^n}$: \hspace{1cm} \hspace{1cm} Deterministic PDA

- Transition-label: (stack-symbol, input-symbol, replacement-string).
- Initially stack is empty; in this case, stack-symbol = $\lambda$.

**Processing of input** $x = aaabbb$:

<table>
<thead>
<tr>
<th>Stack (rightmost symbol on top)</th>
<th>State</th>
<th>Remaining input string</th>
<th>Transition used</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>$q_0$</td>
<td>$aaabbb$</td>
<td>$\lambda/a/a$</td>
</tr>
<tr>
<td>$a$</td>
<td>$q_0$</td>
<td>$aabbb$</td>
<td>$a/a/aa$</td>
</tr>
<tr>
<td>$aa$</td>
<td>$q_0$</td>
<td>$abbb$</td>
<td>$a/a/aa$</td>
</tr>
<tr>
<td>(end of saving a’s) $aaa$</td>
<td>$q_0$</td>
<td>$bbb$</td>
<td>$a/b/\lambda$</td>
</tr>
<tr>
<td>$aa$</td>
<td>$q_1$</td>
<td>$bb$</td>
<td>$a/b/\lambda$</td>
</tr>
<tr>
<td>$a$</td>
<td>$q_1$</td>
<td>$b$</td>
<td>$a/b/\lambda$</td>
</tr>
<tr>
<td>(end of matching b’s) $\lambda$</td>
<td>$q_1$</td>
<td>$\lambda$</td>
<td></td>
</tr>
</tbody>
</table>

**Language Defined by PDA**: \{ $x$: $x$ is accepted \}.

$x$ is accepted: The PDA reaches a final-state on reading all of $x$ and the stack is empty at that point.

$x$ is rejected: $x$ cannot be read completely or after reading $x$, PDA does not reach a final-state or the stack is not empty.
PDA FOR BALANCED PARENTHESELICAL STRINGS

We write $a$ for ’(’ and $b$ for ’)’ for readability.

$L_{\lambda+\text{bal-par}} = \{ \lambda, ab, abab, aabb, 
\quad ababab, abaabb, aabbab, aababb, aaabbb, \ldots \}$

- Each of these strings, except for $\lambda$, contains an initial string of the form $a^n b^n \ (n > 0)$ or has such a string following an initial part $a^m$.
- The removal of that $a^n b^n$ makes the smaller string again balanced.

**Idea:** Merge $q_0$ and $q_1$ in $P_{a^n b^n}$ into a single state to allow the saving of $a$’s and matching off with $b$’s can occur repeatedly.

$$P_{\lambda+\text{bal-par}}: \begin{array}{c}
\frac{a/b/\lambda,}{a/a/aa,} \\
\lambda/a/a
\end{array}$$

Deterministic PDA

**Processing of** $x = a \cdot ab \cdot ab \ b$:

<table>
<thead>
<tr>
<th>Stack memory</th>
<th>State</th>
<th>Remaining input string</th>
<th>Transition used</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>$q_0$</td>
<td>$aababb$</td>
<td>$\lambda/a/a$</td>
</tr>
<tr>
<td>$a$</td>
<td>$q_0$</td>
<td>$ababb$</td>
<td>$a/a/aa$</td>
</tr>
<tr>
<td>$aa$</td>
<td>$q_0$</td>
<td>$babb$</td>
<td>$a/b/\lambda$</td>
</tr>
<tr>
<td>$a$</td>
<td>$q_0$</td>
<td>$abb$</td>
<td>$a/a/aa$</td>
</tr>
<tr>
<td>$aa$</td>
<td>$q_0$</td>
<td>$bb$</td>
<td>$a/b/\lambda$</td>
</tr>
<tr>
<td>$a$</td>
<td>$q_0$</td>
<td>$b$</td>
<td>$a/b/\lambda$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>$q_0$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
</tr>
</tbody>
</table>
EXERCISE.

1. Construct a PDA to accept $L_{bal-par}$; $\lambda \notin L_{bal-par}$.

2. For each PDA below, describe the language accepted by it. Which of the PDAs are deterministic?

3. Consider the language $L_{m \geq n} = \{a^m b^n : m \geq n \geq 1\}$ and the PDAs below. For each PDA, find a smallest string $x \in L_{m \geq n}$ not accepted or $x \notin L_{m \geq n}$ but accepted. For each PDA, eliminate any unnecessary non-determinism if possible (without changing the language of the PDA).

4. Show a PDA which accepts all non-empty strings over the alphabet \{0, 1, $\times$, $=$\} which represent correct multiplications of the form $2^m 2^n = 2^{m+n}$. For $m = 2$ and $n = 3$, the input string is $100 \times 1000 = 100000$. (Hint: the symbols '$\times$' and '$=$' in the input string should drive the PDA to different states.)

5. Show that the language described in Problem 4 is not regular.

6. Give the state-diagram for a deterministic PDA which accepts the language $L_{m=2n} = \{a^m b^n : m = 2n, n \geq 1\}$. Do the same for the language $L_{m=2n+1}$, defined in a similar way.
**CONTEXT-FREE LANGUAGES AND PDAs**

**CFL:** A language is context-free if it accepted by a PDA.

**Language point of view:**
- CFLs describe more general patterns than those described by regular languages. (Each regular language is also a CFL.)

![Diagram showing CFLs and RLs]

- \{a^n b^n : n \geq 1\}
- symmetric strings in \{a, b\}
- balanced parenthetical strings
- bin. mult. of the form \(2^m 2^n = 2^{m+n}\)
- \{a^m b^n : m \geq n \geq 1\}

- \{a^n b^n c^n : n \geq 1\}
- a non-CFL: symmetric balanced parenthetical strings

**Machine point of view:**
- PDAs characterize computations that can be performed by a limited use of external memory, called the stack memory, where the amount of memory used is no more than a constant times the size of the input.

```
A PDA uses at most a linear
(in the size of the input) amount of stack memory.
```
PDA FOR SYMMETRIC STRINGS

\[ L_{\text{sym}} = \text{all non-empty symmetric strings over \{a, b\}}. \]
\[ = \{a, b, aa, bb, aaa, aba, bab, bbb, \ldots\} \]

Idea:

1. Non-deterministically decide a mid-point of the string (and changes state from \( q_0 \) to \( q_1 \)).
2. Non-deterministically decide whether the string is of even length or odd length. In the second case, ignore the unique middle symbol (not matched with the next symbol).
3. Save the first half of the string on the stack, and then match it off with the second half.

\[ P_{\text{sym}}: \quad \begin{array}{cccc}
\lambda/\lambda, \lambda/b/\lambda, & \lambda/a/\lambda, \lambda/b/\lambda, & a/\lambda/a, \lambda/b/\lambda, & a/\lambda/a, \lambda/b/\lambda, \\
\lambda/a/a, \lambda/b/\lambda, & a/\lambda/a, \lambda/b/\lambda, & a/\lambda/a, \lambda/b/\lambda, & a/\lambda/a, \lambda/b/\lambda, \\
\lambda/a/a, \lambda/b/\lambda, & a/\lambda/a, \lambda/b/\lambda, & a/\lambda/a, \lambda/b/\lambda, & a/\lambda/a, \lambda/b/\lambda,
\end{array} \]

Processing of \( x = aabbaa \) (deterministic accepting behavior).

<table>
<thead>
<tr>
<th>Stack memory</th>
<th>State</th>
<th>Remaining input string</th>
<th>Transition used</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda )</td>
<td>( q_0 )</td>
<td>( aabbaa )</td>
<td>( \lambda/a/a )</td>
</tr>
<tr>
<td>( a )</td>
<td>( q_0 )</td>
<td>( abbaa )</td>
<td>( a/a/aa )</td>
</tr>
<tr>
<td>( aa )</td>
<td>( q_0 )</td>
<td>( bbaa )</td>
<td>( a/b/ab )</td>
</tr>
<tr>
<td>( aab )</td>
<td>( q_0 )</td>
<td>( baa )</td>
<td>( b/b/\lambda )</td>
</tr>
<tr>
<td>( aa )</td>
<td>( q_1 )</td>
<td>( aa )</td>
<td>( a/a/\lambda )</td>
</tr>
<tr>
<td>( a )</td>
<td>( q_1 )</td>
<td>( a )</td>
<td>( a/a/\lambda )</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>( q_1 )</td>
<td>( \lambda )</td>
<td>( \lambda/a/\lambda )</td>
</tr>
</tbody>
</table>

Any other processing lead to rejection of \( aabbaa \).
ALL MOVE-SEQUENCES IN $P_{sym}$ FOR $x = ab \notin L_{sym}$

**Configuration:** (stack, state, remaining-input).
- For input $x$ and start-state $= q_0$, initial configuration is $(\lambda, q_0, x)$.

```
(\lambda, q_0, ab)
  \downarrow
(a, q_0, b)    (\lambda, q_1, b)
  \downarrow
(ab, q_0, \lambda)    (a, q_1, \lambda)
```

non-accepting  non-accepting

$x = ab$ is not accepted by $P_{sym}$.

**EXERCISE.**

1. Give the numbers of symmetric strings and non-symmetric strings for length $m = 2k$ and length $m = 2k + 1$.

2. Give the state-diagram of a deterministic PDA for accepting the language $L_{\#a=\#b} = \{ x \in (a+b)^\dagger: \#(a, x) = \#(b, x) \}$.

3. Give a PDA for $L_{a^{n^3}b^n}$.

4. Give the state diagram of deterministic-behavior PDA for $L_{non-sym} = \{ ab, ba, aab, abb, baa, bba, \ldots \}$ with deterministic-behavior. Show the tree of all move sequences of the PDA for $x = ab$.

5. Show a PDA for the language $L_{m\neq n} = \{ a^m b^n: m \neq n, m, n \geq 1 \}$ with deterministic behavior.

Show the configurations in all possible move sequences for your PDA for the input $x = a^2b^2$ to show that it is not accepted.
COMPLEMENT OF \( \{a^n b^n : n \geq 1\} \)

Five kinds of strings in the complement:

<table>
<thead>
<tr>
<th>(1) ( x = \lambda ).</th>
<th>(4) ( x = a^n b^m ) and ( m &lt; n ) (&gt; 0).</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) ( x ) begins with ( b ), i.e., ( x = by ).</td>
<td>(5) ( x = a^n b^m ay ) and ( m \leq n ) (&gt; 0).</td>
</tr>
<tr>
<td>(3) ( x = a^n b^{n+1} y ) ( n &gt; 0 ).</td>
<td></td>
</tr>
</tbody>
</table>

Non-deterministic PDA, with a deterministic accepting behavior.

EXERCISE.

1. Construct a PDA to accept the complement of \( L_{bal-par} \); the complement includes \( \lambda \).
INTERSECTION OF CFL AND RL IS CFL

AND-construction of a PDA and an FSA:
- The states of the new PDA have the form $q_i q'_j$, where $q_i$ is a state of the PDA and $q'_j$ is a state of the FSA. The parts $q_i$ and $q'_j$ change according to the PDA and the FSA, respectively.
- The stack operations remain as in the old PDA.

PDA for $L_{sym} \cap L_{end-b}$:
- Non-deterministic transitions but deterministic behavior.
- The transition $\lambda/a/\lambda$ from $q_0 A$ to $q_1 A$ never takes part in accepting a string (and can be removed). Are there other such transitions?

Question:
- Show the tree of all configurations in processing $aba$. 
EXERCISE

1. Argue that the union of a CFL and an RL is a CFL.

2. Assume that $L_{a^n b^n c^n} = \{a^n b^n c^n : n \geq 1\}$ is not CFL, i.e., it is not accepted by any PDA. Then, show that $\{x \in (a + b + c)^* : #(x, a) = #(x, b) = #(x, c)\}$ is not CFL.

3. Give a PDA for the language $L_{#a=#b} = \{x \in (a + b)^* : #(x, a) = #(x, b)\}$.

4. If $L$ is a CFL and $L'$ is regular, then argue that $LL'$ and $L'L$ are CFL.

5. Since $L_{a^n b^n c^n} = L_{a^n b^n c^n} \cap L_{a^* b^n c^n}$ and both $L_{a^n b^n c^n}$ and $L_{a^* b^n c^n}$ are CFL, it follows that intersection of CFLs need not be a CFL. Thus, complement of a CFL need not be a CFL either.
FINDING NORTHMOST POSITION IN A LINEAR MAZE

Linear Maze: Each square has \( \leq 2 \) neighbors; it is a strip laid in a straight line or a curved non-crossing fashion.

Some observations:

- Starting at 1, when the robot reaches 20 (or 23) the stack must tell us that the robot is one step further north of position 1.
- At 28 (also at 34), the stack should tell us that the maximum north reached so far was one step higher than the start position, and that currently the robot is 4 steps south from that max-north position.
- At 45, the stack must tell us we are still south of the max-north position and it must allow us to get back to position 23, say.

EXERCISE.

1. Design a PDA-control of a robot so that for any initial placement of it in a linear maze, it will stop at a northmost position. Hints: (1) First make the robot go to a dead-end. (2) Use the stack to keep track of north-south motion as it now moves to the other dead-end. Show the stack for each position of the robot till it stops.
TRAVERSING AN ACYCLIC MAZE ONCE

1. Design a PDA to control the motion of a robot so that if it is put in any acyclic maze as shown below, then it will start traversing the maze using "keep to the right" strategy and come to a stop when it arrives at its initial position for the first time; in the process it may have traversed the whole maze or only a part of the maze. (Hint: Use the stack to keep track of information related to the path taken so that when you return to the initial position, the stack is empty.) Show the stack during the operation of robot starting at 3 till it gets back to it for the first time.

![Maze Diagram](image)

2. Is it possible to make the robot stop at a northmost position in any acyclic maze?