Pushdown Automata

The machine model for the context-free language.

Pushdown Automata

A pushdown automaton is an NFA with a last-in, first-out storage device called **stack**.

Pushdown Automata

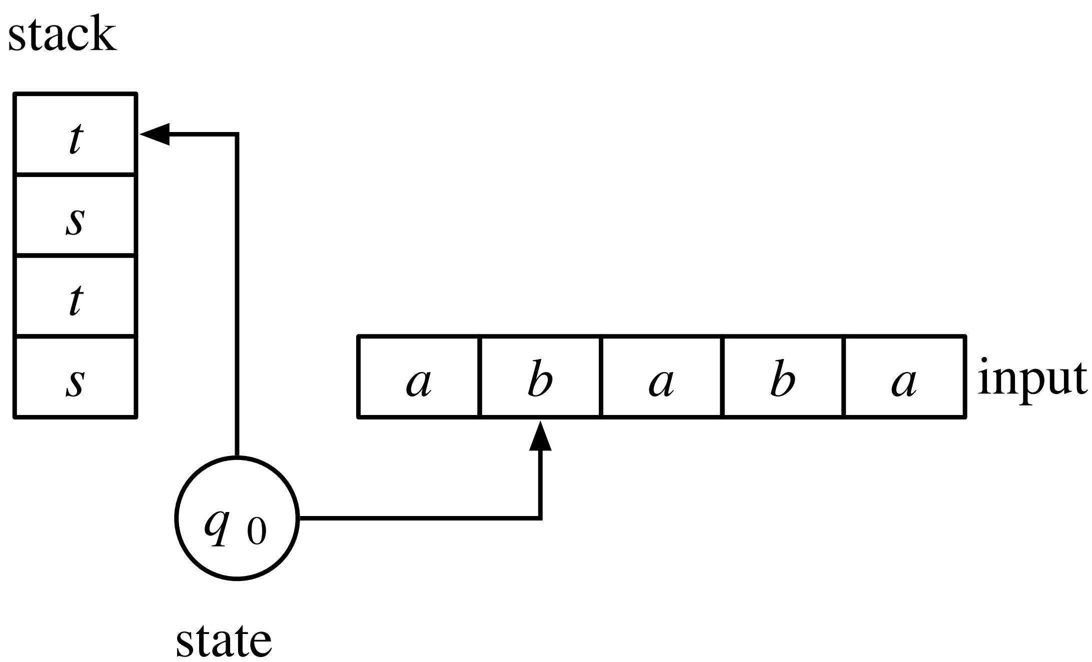
A pushdown automaton is an NFA with a last-in, first-out storage device called **stack**.

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

1. Q is a finite **set of states**,
2. Σ is a finite **input alphabet**,
3. Γ is a finite **stack alphabet**,
4. $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow \mathcal{P}(Q \times \Gamma_\epsilon)$ is the **transition**,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the **set of accept states**,

where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ and $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$.

A Schematic Representation of a PDA



Represent the contents of the stack by taking the letters from top to bottom and putting them from left to right. Here the stack has the word $tsts$.

Computation by Pushdown Automata

(A) If not the entire input has been read, it may choose to **read the next letter**.

- If either the entire input has already been read or it decides not to read the next letter, the input letter is considered to be ϵ.

Computation by Pushdown Automata

(B) It may choose to **read the top letter** of the stack.

- If the stack is already empty, then the computation stops there without accepting.
- If it decides not to read the stack, the stack letter is considered to be ϵ.

Computation by Pushdown Automata

(C) Depending on the current state, the input letter, and the stack letter, it **nondeterministically decides the next state** and **a letter to be placed on the stack**, with a possible option of not placing a letter, in which case the letter is considered to be ϵ.

Transition Function of a PDA

- The input: $Q \times \Sigma_\epsilon \times \Gamma_\epsilon$.
- The output: $2^{Q \times \Gamma_\epsilon}$.

Acceptance of Pushdown Automata

A pushdown automaton M *accepts* an input w if M arrives at an accept state sometimes after reading all the input letters.

A computation path of M on input w **halts without accepting** if there is no applicable next move.

If the current state is q, the next input letter is a, and the top stack letter is b, there are four possible courses of action:

1. a move in $\delta(q, \epsilon, \epsilon)$
2. a move in $\delta(q, \epsilon, b)$
3. a move in $\delta(q, a, \epsilon)$
4. a move in $\delta(q, a, b)$

Acceptance of Pushdown Automata

A pushdown automaton M *accepts* an input w if M arrives at an accept state sometimes after reading all the input letters.

A computation path of M on input w **halts without accepting** if there is no applicable next move.

If the current state is q, the next input letter is a, and the top stack letter is b, there are four possible courses of action:

1. a move in $\delta(q, \epsilon, \epsilon)$
2. a move in $\delta(q, \epsilon, b)$ – **impossible if stack is empty**
3. a move in $\delta(q, a, \epsilon)$ – **impossible if no input letter is left**
4. a move in $\delta(q, a, b)$ – **impossible if no input letter is left or stack is empty**

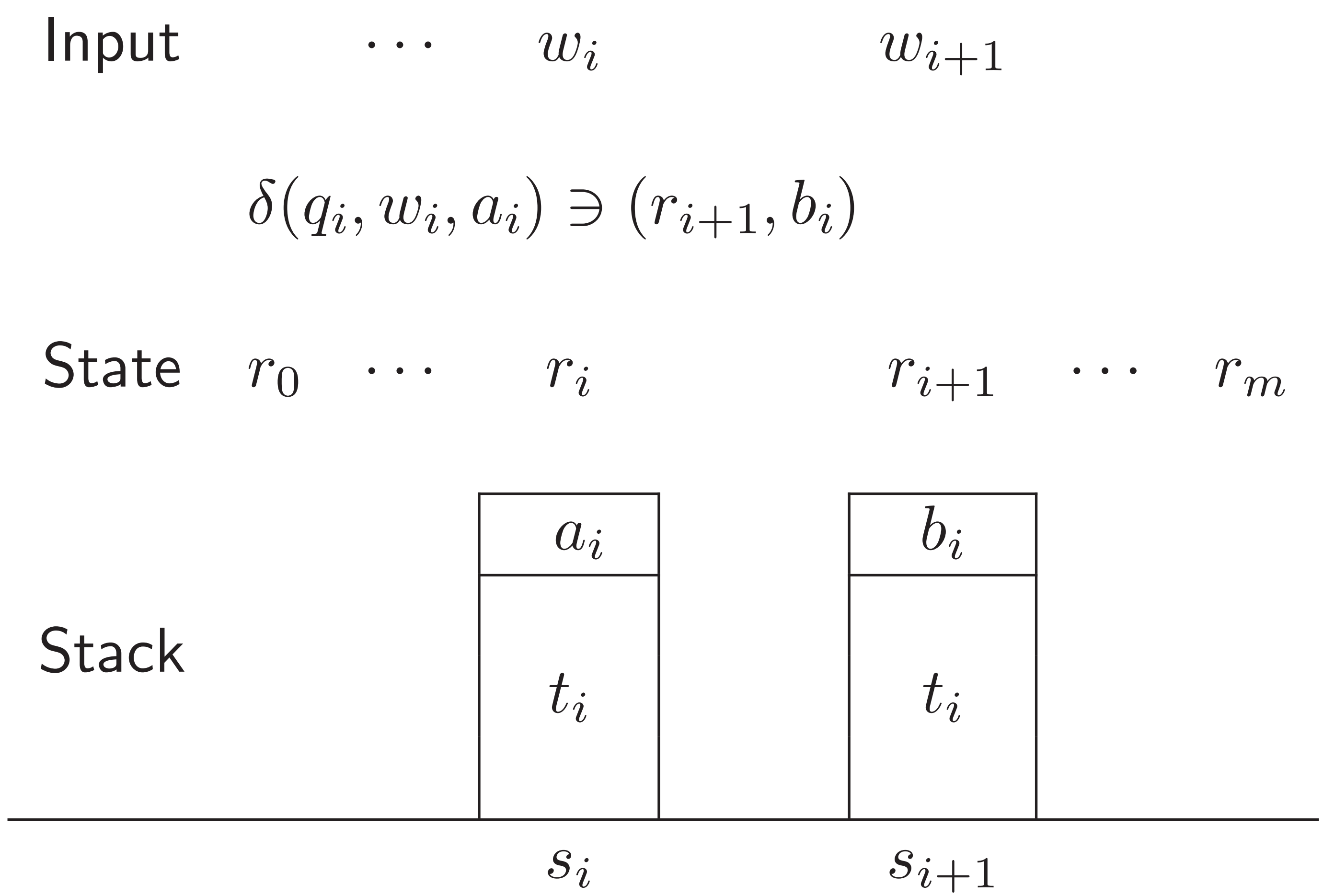
Acceptance by a Pushdown Automaton

Formally, M **accepts** $w \in \Sigma^*$ if there exist

- $r_0, \ldots, r_m \in Q$ (states), $w_1, \ldots, w_m \in \Sigma_\epsilon$ (input letters),
- $a_1, \ldots, a_m \in \Gamma_\epsilon$ (stack letters read),
- $b_1, \ldots, b_m \in \Gamma_\epsilon$ (stack letters written),
- $s_1, \ldots, s_m \in \Gamma^*$ (stack word before pop),
- $t_1, \ldots, t_m \in \Gamma^*$ (stack word after pop), such that:

1. $r_0 = q_0$, $r_m \in F$ (start with q_0 and accept),
2. $w = w_1 \cdots w_m$ (the input decomposition),
3. $a_1 = s_1 = t_1 = \epsilon$ (start with empty stack),
4. for all $i, 1 \le i \le m-1$, $s_i = a_i t_i$ and $s_{i+1} = b_i t_i$ (preservation of stack content other than top),
5. for all $i, 1 \le i \le m$, $(r_i, b_i) \in \delta(r_{i-1}, a_{i-1}, b_{i-1})$ (following transition).

Diagram



Input $\cdots$ w_i w_{i+1}
$\delta(q_i, w_i, a_i) \ni (r_{i+1}, b_i)$
State r_0 $\cdots$ r_i r_{i+1} $\cdots$ r_m
Stack
a_i
t_i
b_i
t_i
s_i
s_{i+1}

Example 1

A PDA for $L = \{0^n 1^n \mid n \geq 0\}$.

For example, 000111 is a member, 0011 is a member, but 00110011 is not.

Example 1

Idea

Using stack keep a tally of the leading 0s. Compare the number against the number of trailing 1s.

Example 1

Algorithm

1. May accept immediately.
2. Place a special symbol $\$$ on the stack to **mark the bottom.**
3. Read input symbols without popping from stack. If the symbol is a 0, put a 0 onto stack; otherwise, stop.
4. Guess the start of 1s and begin simultaneously reading input and popping from stack.

   If either the input symbol is not a 1 or the stack symbol popped is not a 0, stop.

   Any time during this, stop reading the input and then
   - verify that the top of stack is $\$$ and accept.

How This Method Works

- 0^n1^n for some $n \geq 1$: Will accept.
- ϵ: Will accept by choosing to verify $\$$ immediately after place it on the top.

How This Method Works

- $0^n 1^n$ for some $n \geq 1$: Will accept.
- ϵ: Will accept by choosing to verify $\$$ immediately after place it on the top.
- $1y$ for some $y \in \{0, 1\}^*$: Will either pop the $\$$ on reading the first 1 or enter accept without reading any input letter.
- $0^n 1^{n+k}$ for some $n, k \geq 1$: Will either pop the $\$$ on reading the first 1 or enter accept without finishing to read the input.
- $0^n 1^{n-k}$ for some $n, k \geq 1$: Verification for $\$$ will fail.

Transition Function

$\Gamma = \{0, 1, \$\}$, $Q = \{q_1, q_2, q_3, q_4\}$, $F = \{q_1, q_4\}$, q_1 is the initial state

Input:	0		1		ϵ		
Stack:	$0/\$$	ϵ	0	$\$/\epsilon$	0	$\$$	ϵ
q_1							$\{(q_2, \$)\}$
q_2		$\{(q_2, 0)\}$	$\{(q_3, \epsilon)\}$				
q_3			$\{(q_3, \epsilon)\}$			$\{(q_4, \epsilon)\}$	

Example 1

Let (q, u, v), $q \in Q$, $u \in \Sigma^*$, $v \in \Gamma^*$, denote the configuration where **the state is q, the remaining input is u, and the stack word is v.**

000111 is accepted by the following path:

$(q_1, 000111, \epsilon) \Rightarrow (q_2, 000111, \$) \Rightarrow (q_2, 00111, 0\$)$
$\Rightarrow (q_2, 0111, 00\$) \Rightarrow (q_2, 111, 000\$) \Rightarrow (q_3, 11, 00\$)$
$\Rightarrow (q_3, 1, 0\$) \Rightarrow (q_3, \epsilon, \$) \Rightarrow (q_4, \epsilon, \epsilon)$.

Example 2

$L = \{u \in \{0,1\}^* \mid u$ has the same number of 0s as 1s $\}$.

For example, 011100 is a member, 10010101 is a member, and 00010 is a nonmember.

Example 2

$L = \{u \in \{0,1\}^* \mid u$ has the same number of 0s as 1s $\}$.

Idea

Using stack maintain a *tally of the difference between the number of 0s and the number of 1s that have been read so far*. Use a tally of 0s if there have been more 0s than 1s and a tally of 1s if there have been more 1s than 0s.

Compare the first letter of the tally and an input letter. If one is a 0 and the other is a 1, they cancel out; otherwise, increase the tally.

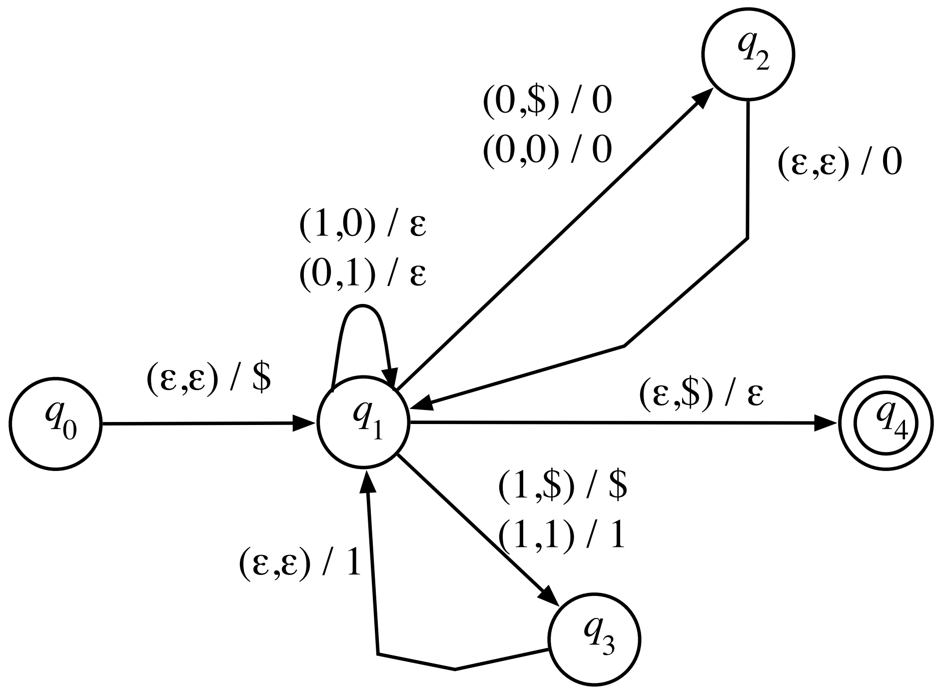
Example 2

$L = \{u \in \{0,1\}^* \mid u$ has the same number of 0s as 1s $\}$.
$Q = \{q_0, q_1, q_2, q_3, q_4\}$, q_4:final, q_0:initial
No permissible actions in q_4
$\Gamma = \{0, 1, \$\}$

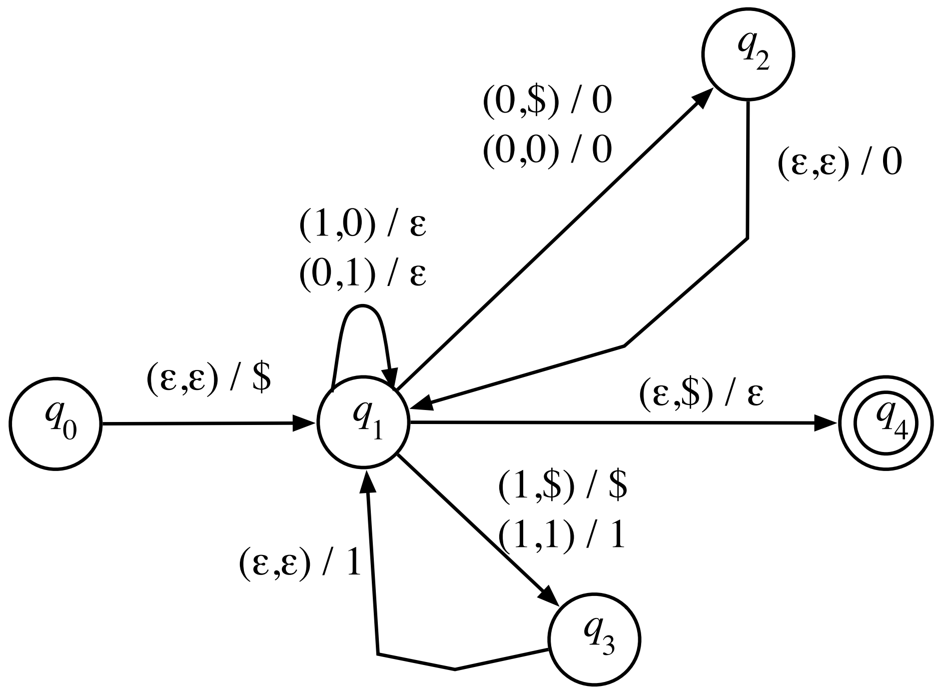
State	Input	Stack			
		0	1	$\$$	ϵ
q_0	ϵ				$(q_1, \$)$
q_1	0	$(q_2, 0)$	(q_1, ϵ)	$(q_2, \$)$	
	1	(q_1, ϵ)	$(q_3, 1)$	$(q_3, \$)$	
	ϵ			(q_4, ϵ)	
q_2	ϵ				$(q_1, 0)$
q_3	ϵ				$(q_1, 1)$

Here { and } are omitted.

Example 2 Diagram



Example 2 Diagram



Example: $(q_0, 011100, \epsilon) \Rightarrow (q_1, 011100, \$) \Rightarrow (q_2, 11100, \$) \Rightarrow (q_1, 11100, 0\$) \Rightarrow (q_1, 1100, \$) \Rightarrow (q_3, 100, \$) \Rightarrow (q_1, 100, 1\$) \Rightarrow (q_3, 00, 1\$) \Rightarrow (q_1, 00, 11\$) \Rightarrow (q_1, 0, 1\$) \Rightarrow (q_1, \epsilon, \$) \Rightarrow (q_4, \epsilon, \epsilon)$