CS 301

Lecture 12 – Pushdown automata

A new type of machine

DFAs and NFAs are *finite* and that turns out to be too limiting

Even simple languages like $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$ are too complicated

What we want is some way to remember things about the input that we've seen so far which can be arbitrarily long

A new type of machine

DFAs and NFAs are *finite* and that turns out to be too limiting

Even simple languages like $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$ are too complicated

What we want is some way to remember things about the input that we've seen so far which can be arbitrarily long

So let's add a *stack* to an NFA!

Pushdown automaton (PDA)

Like an NFA, it has

- A finite set of states Q
- An input alphabet Σ
- A transition function δ
- A start state q_0
- A set of accepting states F

New to the PDA is a stack and a corresponding stack alphabet Γ

The transition function is modified to handle the stack

PDA transition function

A PDA can examine both the next symbol in the input and the top of the stack to decide what actions to take with respect to the stack

There are four stack actions. The PDA can

1. ignore the stack entirely;

PDA transition function

A PDA can examine both the next symbol in the input and the top of the stack to decide what actions to take with respect to the stack

There are four stack actions. The PDA can

1. ignore the stack entirely;
2. push a symbol onto the stack without examining the top of the stack;

PDA transition function

A PDA can examine both the next symbol in the input and the top of the stack to decide what actions to take with respect to the stack

There are four stack actions. The PDA can

1. ignore the stack entirely;
2. push a symbol onto the stack without examining the top of the stack;
3. pop a symbol off of the stack; or

PDA transition function

A PDA can examine both the next symbol in the input and the top of the stack to decide what actions to take with respect to the stack

There are four stack actions. The PDA can

1. ignore the stack entirely;
2. push a symbol onto the stack without examining the top of the stack;
3. pop a symbol off of the stack; or
4. replace the symbol on the top of the stack with a new (or the same) symbol

PDA transition function

A PDA can examine both the next symbol in the input and the top of the stack to decide what actions to take with respect to the stack

There are four stack actions. The PDA can

1. ignore the stack entirely;
2. push a symbol onto the stack without examining the top of the stack;
3. pop a symbol off of the stack; or
4. replace the symbol on the top of the stack with a new (or the same) symbol

In cases 1 and 2, the PDA makes its decision *without* looking at the symbol on the top of the stack

In cases 3 and 4, the PDA explicitly examines the symbol at the top of the stack and either removes it or replaces it

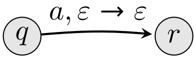
PDAs are nondeterministic

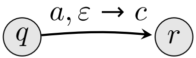
At each step, the PDA has multiple options:

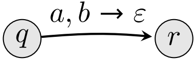
- It can move to one of several possible states
- It can perform one of the four stack actions
- It can transition without examining the next input symbol

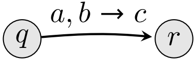
Transitions

There are four possible transitions from state q to state r on input a

1. 

ignore the stack
2. 

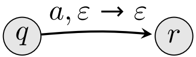
push c onto the stack
3. 

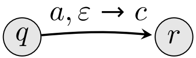
pop b from the top of the stack
4. 

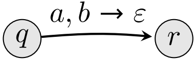
replace b on the top of the stack with c

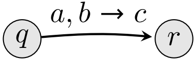
Transitions

There are four possible transitions from state q to state r on input a

1. 

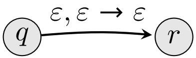
ignore the stack
2. 

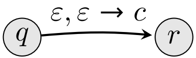
push c onto the stack
3. 

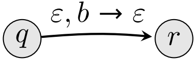
pop b from the top of the stack
4. 

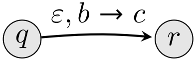
replace b on the top of the stack with c

There are four possible transitions from state q to state r on no input (ε-transition)

1. 

ignore the stack
2. 

push c onto the stack
3. 

pop b from the top of the stack
4. 

replace b on the top of the stack with c

Example

Build a PDA to recognize $A = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$

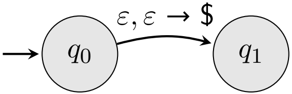
Informal description

1. While the next input symbol is `a`, push it onto the stack
2. Once all of the `a`s have been read, transition to a new state
3. While the next input symbol is `b` and the top of the stack is `a`, pop the `a`
4. At the end of the input, if the stack is empty, accept

How do we know if the stack is empty?

The stack alphabet Γ doesn't need to be the same as the input alphabet Σ

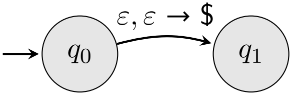
Let's add a end-of-stack marker \$ as the first step



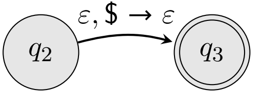
How do we know if the stack is empty?

The stack alphabet Γ doesn't need to be the same as the input alphabet Σ

Let's add a end-of-stack marker \$ as the first step

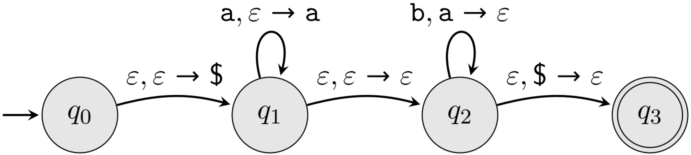


Before we accept, we can ensure the stack is empty by popping the \$



Example

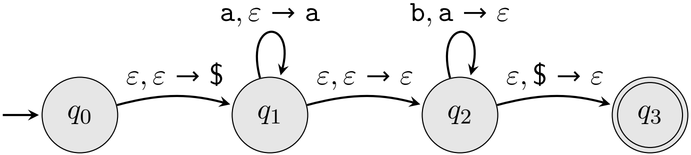
Build a PDA to recognize $A = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$
The input alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$; let's use a stack alphabet $\Gamma = \{\mathtt{a}, \$\}$



When run on some input, the PDA

Example

Build a PDA to recognize $A = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$
The input alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$; let's use a stack alphabet $\Gamma = \{\mathtt{a}, \$\}$

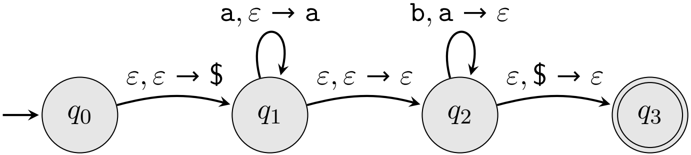


When run on some input, the PDA

1. starts in q_0;

Example

Build a PDA to recognize $A = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$
The input alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$; let's use a stack alphabet $\Gamma = \{\mathtt{a}, \$\}$

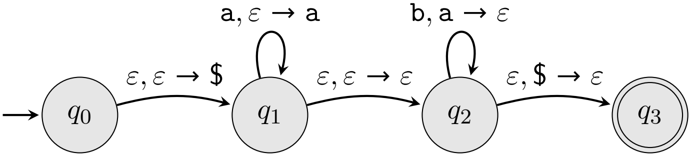


When run on some input, the PDA

1. starts in q_0;
2. pushes \$ onto the stack and moves to q_1;

Example

Build a PDA to recognize $A = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$
The input alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$; let's use a stack alphabet $\Gamma = \{\mathtt{a}, \$\}$

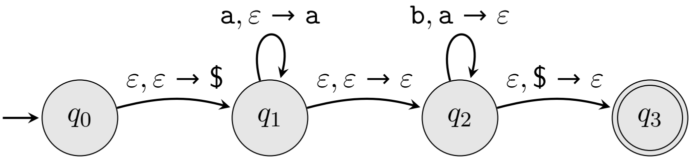


When run on some input, the PDA

1. starts in q_0;
2. pushes \$ onto the stack and moves to q_1;
3. remains in q_1 reading as and pushing them on the stack;

Example

Build a PDA to recognize $A = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$
The input alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$; let's use a stack alphabet $\Gamma = \{\mathtt{a}, \$\}$

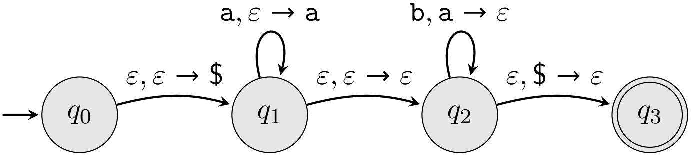


When run on some input, the PDA

1. starts in q_0;
2. pushes \$ onto the stack and moves to q_1;
3. remains in q_1 reading as and pushing them on the stack;
4. moves to q_2

Example

Build a PDA to recognize $A = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$
The input alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$; let's use a stack alphabet $\Gamma = \{\mathtt{a}, \$\}$

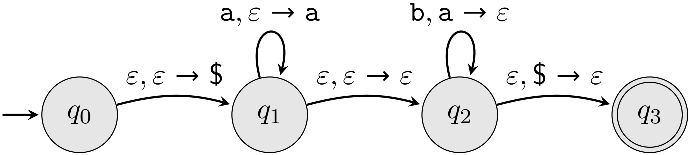


When run on some input, the PDA

1. starts in q_0;
2. pushes \$ onto the stack and moves to q_1;
3. remains in q_1 reading as and pushing them on the stack;
4. moves to q_2
5. remains in q_2 reading bs and popping as off the stack;

Example

Build a PDA to recognize $A = \{\texttt{a}^n\texttt{b}^n \mid n \geq 0\}$
The input alphabet is $\Sigma = \{\texttt{a}, \texttt{b}\}$; let's use a stack alphabet $\Gamma = \{\texttt{a}, \$\}$

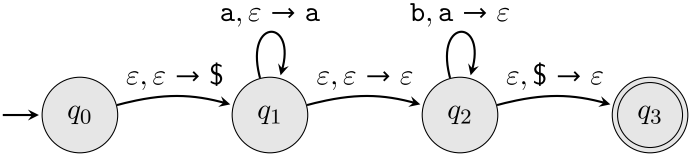


When run on some input, the PDA

1. starts in q_0;
2. pushes \$ onto the stack and moves to q_1;
3. remains in q_1 reading as and pushing them on the stack;
4. moves to q_2
5. remains in q_2 reading bs and popping as off the stack;
6. once \$ is on the top of the stack, it moves to q_3; and

Example

Build a PDA to recognize $A = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$
The input alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$; let's use a stack alphabet $\Gamma = \{\mathtt{a}, \$\}$



When run on some input, the PDA

1. starts in q_0;
2. pushes \$ onto the stack and moves to q_1;
3. remains in q_1 reading as and pushing them on the stack;
4. moves to q_2
5. remains in q_2 reading bs and popping as off the stack;
6. once \$ is on the top of the stack, it moves to q_3; and
7. if there's no more input, it accepts

Formal definition

A PDA is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ with

- Q – finite set of states
- Σ – input alphabet
- Γ – stack alphabet
- δ – transition function
- q_0 – start state
- F – set of accepting states

Formal definition

A PDA is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ with

- Q – finite set of states
- Σ – input alphabet
- Γ – stack alphabet
- δ – transition function
- q_0 – start state
- F – set of accepting states

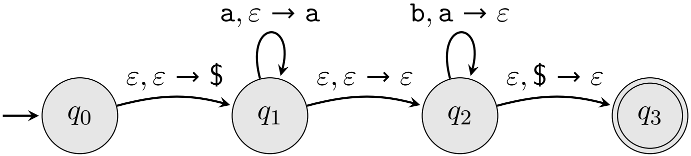
The transition function is complicated

$$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$$

It takes as input a state, an input symbol or ε, a stack symbol or ε

It returns 0 or more pairs of a state and a stack symbol or ε

Example's transition function



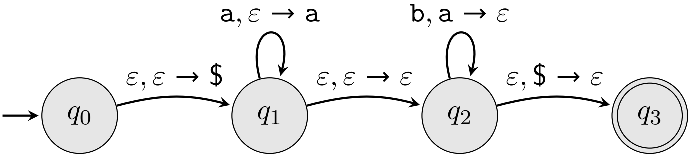
$$\delta(q_0, t, s) = \begin{cases} \{(q_1, \varepsilon)\} & \text{if } t = \varepsilon \text{ and } s = \varepsilon \\ \varnothing & \text{otherwise} \end{cases}$$

$$\delta(q_1, t, s) = \begin{cases} \{(q_1, \mathtt{a})\} & \text{if } t = \mathtt{a} \text{ and } s = \varepsilon \\ \{(q_2, \varepsilon)\} & \text{if } t = \varepsilon \text{ and } s = \varepsilon \end{cases}$$

$$\delta(q_2, t, s) = \begin{cases} \{(q_2, \varepsilon)\} & \text{if } t = \mathtt{b} \text{ and } s = \mathtt{a} \\ \{(q_3, \varepsilon)\} & \text{if } t = \varepsilon \text{ and } s = \$ \end{cases}$$

$$\delta(q_3, t, s) = \varnothing$$

Example's transition function in tabular form



$\delta(q, t, s)$:

	$t = \texttt{a}$			$t = \texttt{b}$			$t = \varepsilon$		
	$s = \texttt{a}$	$s = \$$	$s = \varepsilon$	$s = \texttt{a}$	$s = \$$	$s = \varepsilon$	$s = \texttt{a}$	$s = \$$	$s = \varepsilon$
q_0									$\{(q_1, \$)\}$
q_1			$\{(q_1, \texttt{a})\}$						$\{(q_2, \varepsilon)\}$
q_2				$\{(q_2, \varepsilon)\}$				$\{(q_3, \varepsilon)\}$	
q_3									

All blank entries are $\varnothing$

PDA acceptance

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a string $w = w_1 w_2 \cdots w_n$ for $w_i \in \Sigma_\varepsilon$ if there exist

- states $r_0, r_1, \ldots, r_n \in Q$ and
- strings $s_0, s_1, \ldots, s_n \in \Gamma^*$ (representing the stacks)

such that

PDA acceptance

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a string $w = w_1 w_2 \cdots w_n$ for $w_i \in \Sigma_\varepsilon$ if there exist

- states $r_0, r_1, \ldots, r_n \in Q$ and
- strings $s_0, s_1, \ldots, s_n \in \Gamma^*$ (representing the stacks)

such that

1. $r_0 = q_0$ and $s_0 = \varepsilon$
   (i.e., M starts in the start state with an empty stack);

PDA acceptance

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a string $w = w_1 w_2 \cdots w_n$ for $w_i \in \Sigma_\varepsilon$ if there exist

- states $r_0, r_1, \ldots, r_n \in Q$ and
- strings $s_0, s_1, \ldots, s_n \in \Gamma^*$ (representing the stacks)

such that

1. $r_0 = q_0$ and $s_0 = \varepsilon$
   (i.e., M starts in the start state with an empty stack);
2. $xu = s_{i-1}$ for some $x \in \Gamma_\varepsilon$ and $u \in \Gamma^*$, $(r_i, y) \in \delta(r_{i-1}, w_i, x)$, and $s_i = yu$
   (i.e., M moves from state r_{i-1} with stack s_{i-1} to state r_i with stack s_i according to δ); and

PDA acceptance

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a string $w = w_1 w_2 \cdots w_n$ for $w_i \in \Sigma_\varepsilon$ if there exist

- states $r_0, r_1, \ldots, r_n \in Q$ and
- strings $s_0, s_1, \ldots, s_n \in \Gamma^*$ (representing the stacks)

such that

1. $r_0 = q_0$ and $s_0 = \varepsilon$
   (i.e., M starts in the start state with an empty stack);
2. $xu = s_{i-1}$ for some $x \in \Gamma_\varepsilon$ and $u \in \Gamma^*$, $(r_i, y) \in \delta(r_{i-1}, w_i, x)$, and $s_i = yu$
   (i.e., M moves from state r_{i-1} with stack s_{i-1} to state r_i with stack s_i according to δ); and
3. $r_n \in F$
   (i.e., M ends in an accept state)

More PDAs!

Build a PDA to recognize the languages

- $B = \{w \mid w \in \{\mathtt{a}, \mathtt{b}\}^* \text{ and } w \text{ has the same number of as as bs}\}$
- $C = \{w\#w^{\mathcal{R}} \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$
- $D = \{\mathtt{a}^k\#w \mid k > 0,\ w \in \{\mathtt{a}, \mathtt{b}\}^*, \text{ and } |w| = k\}$
- $E = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i = j \text{ or } j = k\}$
- $F = \{ww^{\mathcal{R}} \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$
- $G = \{w \mid w \in \{\mathtt{a}, \mathtt{b}\}^* \text{ and } w = w^{\mathcal{R}}\}$
- H is given by the CFG

$$S \to SS \mid \mathtt{(}S\mathtt{)} \mid \mathtt{[}S\mathtt{]} \mid \varepsilon$$

- I is given by the CFG

$$E \to E\mathtt{+}E \mid E\mathtt{-}E \mid \mathtt{(}E\mathtt{)} \mid BN$$
$$N \to BN \mid \varepsilon$$
$$B \to \mathtt{0} \mid \mathtt{1}$$