

## Pushdown Automata

12345678910



## Pushdown Automata

### Formal Definition

A **Pushdown Automaton** (PDA) is a 6-tuple  $M = (K, \Sigma, \Gamma, \Delta, s, F)$  consisting of:

1. A finite collection of **states**  $K$ .
2. An input alphabet  $\Sigma$ .
3. A stack alphabet  $\Gamma$ .
4. A **transition relation**  $\Delta$  on  $(K \times \Sigma \cup \{\Lambda\}) \times \Gamma^* \times (K \times \Gamma^*)$ .
5. An **initial state**  $s \in K$ .
6. A collection of **final** or **accepting states**  $F \subseteq K$ .

12345678910

## Pushdown Automata

### Introduction

We've already seen that finite automata are not powerful enough to accept some types of languages.

What is lacking in a finite automaton (at least for the types of languages that can be generated by context-free grammars) is some sort of memory other than the state the automaton is in.

When trying to accept strings in the language  $\{a^n b^n \mid n \geq 0\}$  we need some way for the machine to remember how many  $a$ 's have been seen as we start consuming the  $b$ 's.

Suggestions?

12345678910

## Pushdown Automata

### Transition Relation

The transition relation  $\Delta$  is actually similar to that found in nondeterministic finite automata; the main difference is that now a **stack** is consulted and perhaps modified.

If  $((p, a, x), (q, y)) \in \Delta$  then  $p$  is the current state the PDA is in and  $q$  is the state it can transition to. The input symbol is  $a$  and the element on the top of the stack is  $x$ . Thus, this transition can only occur if the automaton is in state  $p$ , the symbol under the tape head is  $a$ , and the symbol on the top of the stack is  $x$ .

If this transition occurs then the stack element  $x$  is replaced with the element  $y$ . We say that  $x$  is **popped** and  $y$  is **pushed**.

The empty string  $\Lambda$  appearing in the  $x$  stack-symbol position indicates that nothing is popped; if it appears in the  $y$  position then nothing is pushed.

12345678910

### Pushdown Automata

[prev](#) | [slides](#) | [next](#)

#### Example

The following PDA accepts the language  $L = \{a^n b^n \mid n \geq 0\}$ .

$K = \{q_0, q_1, q_2\}$	with $\Delta$ given by
$\Sigma = \{a, b\}$	$(q_0, a, \Lambda) \rightarrow (q_1, x)$
$\Gamma = \{x\}$	$(q_1, a, \Lambda) \rightarrow (q_1, x)$
$s = q_0$	$(q_1, b, x) \rightarrow (q_2, \Lambda)$
$F = \{q_0, q_2\}$	$(q_2, b, x) \rightarrow (q_2, \Lambda)$

12345678910

### Pushdown Automata

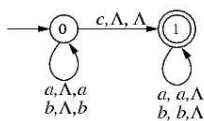
[prev](#) | [slides](#) | [next](#)

#### Another Example

Consider the language  $L = \{wcw^R \mid w \text{ in } \{a, b\}^*\}$ . The following PDA accepts this language.

$K = \{q_0, q_1\}$	with $\Delta$ given by
$\Sigma = \{a, b, c\}$	$(q_0, a, \Lambda) \rightarrow (q_0, a)$
$\Gamma = \{a, b\}$	$(q_0, b, \Lambda) \rightarrow (q_0, b)$
$s = q_0$	$(q_0, c, \Lambda) \rightarrow (q_1, \Lambda)$
$F = \{q_1\}$	$(q_1, a, a) \rightarrow (q_1, \Lambda)$
	$(q_1, b, b) \rightarrow (q_1, \Lambda)$

Transition Diagram:



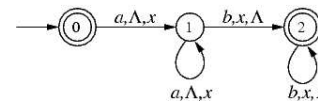
12345678910

### Pushdown Automata

[prev](#) | [slides](#) | [next](#)

#### Example

We can draw a transition diagram that may be easier to digest:



Here the transitions are labeled with 3-tuples: the first element is the input symbol, the second element is the symbol to pop off the stack and the third element is the symbol to push on the stack. As before,  $\Lambda$  indicates that nothing is popped or that nothing is pushed.

**Important Note:** A PDA accepts a string if and only if it halts in an accepting state with no more input to consume **and the stack is empty**, i.e. the top symbol on the stack is  $\Lambda$ .

12345678910

### Pushdown Automata

[prev](#) | [slides](#) | [next](#)

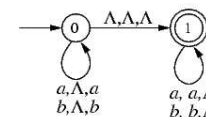
#### Yet Another Example

Compare the last language with the language  $L = \{ww^R \mid w \text{ in } \{a, b\}^*\}$ . Because PDA's can be nondeterministic it is not difficult to construct a PDA that accepts this language. The only change between this PDA and the last one is in the third rule of  $\Delta$ .

$K = \{q_0, q_1\}$	with $\Delta$ given by
$\Sigma = \{a, b, c\}$	$(q_0, a, \Lambda) \rightarrow (q_0, a)$
$\Gamma = \{a, b\}$	$(q_0, b, \Lambda) \rightarrow (q_0, b)$
$s = q_0$	$(q_0, \Lambda, \Lambda) \rightarrow (q_1, \Lambda)$
$F = \{q_1\}$	$(q_1, a, a) \rightarrow (q_1, \Lambda)$
	$(q_1, b, b) \rightarrow (q_1, \Lambda)$

$K = \{q_0, q_1\}$	with $\Delta$ given by
$\Sigma = \{a, b, c\}$	$(q_0, a, \Lambda) \rightarrow (q_0, a)$
$\Gamma = \{a, b\}$	$(q_0, b, \Lambda) \rightarrow (q_0, b)$
$s = q_0$	$(q_0, \Lambda, \Lambda) \rightarrow (q_1, \Lambda)$
$F = \{q_1\}$	$(q_1, a, a) \rightarrow (q_1, \Lambda)$
	$(q_1, b, b) \rightarrow (q_1, \Lambda)$

Transition Diagram:



12345678910

## Pushdown Automata

[prev](#) | [slides](#) | [next](#)

### Configuration of a PDA

Recall that the state a finite automaton is in, along with an input symbol completely determines what a deterministic finite automaton will do. Even in the case of NFAs the current state and the input symbol specify a list of possible transitions. The future behavior of the automaton is independent of how it got to the current state.

PDAs are similar in that the current state, the input symbol and the symbol on the top of the stack completely determine possible transitions. How the PDA behaves from that point on is completely independent of how it got there.

The **configuration of a PDA** is an element of  $K \times \Sigma^* \times \Gamma^*$ .

The initial configuration of a PDA with start state  $s$  and input string  $w$  is  $(s, w, \Lambda)$ . To accept the string  $w$  the PDA must finish processing  $w$  in the configuration  $(f, \Lambda, \Lambda)$  where  $f$  is a state in  $F$ .

[12345678910](#)

## Pushdown Automata

[prev](#) | [slides](#) | [next](#)

### Configuration of a PDA

For example, we can see how the string  $ababbaba$  is accepted by the PDA that accepts the language  $\{ww^R \mid w \in \{a, b\}^*\}$ .

$$\begin{aligned} (0, ababbaba, \Lambda) &\vdash (0, babbaba, a) \\ &\vdash (0, abbaba, ba) \\ &\vdash (0, bbaba, aba) \\ &\vdash (0, baba, baba) \\ &\vdash (1, baba, baba) \\ &\vdash (1, aba, aba) \\ &\vdash (1, ba, ba) \\ &\vdash (1, a, a) \\ &\vdash (1, \Lambda, \Lambda) \end{aligned}$$

[12345678910](#)