

# TDDD14 / TDDD85 – Lecture 9

## Pushdown Automata

August Ernstsson, 2026 (based on lecture notes by Jonas Wallgren)

# Pushdown Automata: Introduction

- **Known:** Finite automata can express regular languages
- We seek an automaton that can *express context-free languages*
- We start with the model for an NFA, and extend it with a *memory component*
- A pushdown memory; a "**stack**":
  - Only the **top** element can be read,
  - we can **push** a new top element,
  - or **pop** the old top element and discard it.

# Example 1: A Pushdown Automaton

- A PDA for  $L = \{ w c w^R \mid w \in \{0, 1\}^* \}$  ( $w^R$  means  $w$  reversed)
  - i.e., strings in  $L$  are *palindromes* over  $\{0,1\}$  with a "c" in the center.
  - **Example:** "01c10"
- **Each step:**
  - Read 0, 1, or c.
  - Put B, G, or R on the stack.
- **Start:** in state  $q_1$ , with R on the stack.

# Example 1: A Pushdown Automaton

Top of stack	State	Symbol read		
		0	1	c
B	q <sub>1</sub>	PushB q <sub>1</sub>	PushG q <sub>1</sub>	q <sub>2</sub>
	q <sub>2</sub>	Pop q <sub>2</sub>		
G	q <sub>1</sub>	PushB q <sub>1</sub>	PushG q <sub>1</sub>	q <sub>2</sub>
	q <sub>2</sub>		Pop q <sub>2</sub>	
R	q <sub>1</sub>	PushB q <sub>1</sub>	PushG q <sub>1</sub>	q <sub>2</sub>
	q <sub>2</sub>	Without reading: Pop q <sub>2</sub>		

Action  
New state

- A PDA for  $L = \{ wcw^R \mid w \in \{0, 1\}^* \}$
- where  $w^R$  is  $w$  but reversed, i.e., strings in  $L$  *palindrome* over  $\{0,1\}$  with a "c" in the center.
- **Each step:** Read 0, 1, or c.
- Put B, G, or R on the stack.
- **Start:** in state  $q_1$ , with R on the stack.

# Example 2: PDA string acceptance

Top of stack	State	Symbol read		
		0	1	c
B	q <sub>1</sub>	PushB q <sub>1</sub>	PushG q <sub>1</sub>	q <sub>2</sub>
	q <sub>2</sub>	Pop q <sub>2</sub>		
G	q <sub>1</sub>	PushB q <sub>1</sub>	PushG q <sub>1</sub>	q <sub>2</sub>
	q <sub>2</sub>		Pop q <sub>2</sub>	
R	q <sub>1</sub>	PushB q <sub>1</sub>	PushG q <sub>1</sub>	q <sub>2</sub>
	q <sub>2</sub>	Without reading: Pop q <sub>2</sub>		

- Read the string **01c10**:

State	Remaining string	Stack	Comment
q <sub>1</sub>	01c10	R	Start situation
q <sub>1</sub>	1c10	BR	According to column 0, row Rq <sub>1</sub> in the table
q <sub>1</sub>	c10	GBR	According to column 1, row Bq <sub>1</sub> in the table
q <sub>2</sub>	10	GBR	According to column c, row Gq <sub>1</sub> in the table
q <sub>2</sub>	0	BR	According to column 1, row Gq <sub>2</sub> in the table
q <sub>2</sub>			<b>R According to column 0, row Bq<sub>2</sub> in the table</b>
q <sub>2</sub>			String is read, stack is empty: <b>accept!</b>

# Definition 1: Pushdown Automata

- Specifically, Nondeterministic Push-Down Automata (NPDA)
- A NPDA is a septuple  $\langle Q, \Sigma, \Gamma, \delta, s, \perp, F \rangle$ 
  - $Q$  = set of states
  - $\Sigma$  = (input) alphabet
  - $\Gamma$  = stack alphabet
  - $s$  = start state  $\in Q$
  - $F$  = final states  $\subseteq Q$
  - $\perp$  = start stack symbol (= "bottom" symbol)
  - $\delta$  = transition relation  $\subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$  (Next slide)

# The transition relation $\delta$

- $\delta = \text{transition relation} \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$

↓ Read-transition!

- $\langle \langle \mathbf{p}, \mathbf{a}, \mathbf{A} \rangle, \langle \mathbf{q}, \mathbf{B}_1\mathbf{B}_2\dots\mathbf{B}_n \rangle \rangle \in \delta$  means:
  - In state  $\mathbf{p}$  with  $\mathbf{A}$  on top of the stack:
    - Read  $\mathbf{a}$ ,
    - go to state  $\mathbf{q}$ ,
    - change the stack top  $\mathbf{A}$  to  $\mathbf{B}_1\mathbf{B}_2\dots\mathbf{B}_n$  (the leftmost is the stack top).

# The transition relation $\delta$

- $\delta = \text{transition relation} \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$

↓ Epsilon-transition!

- $\langle \langle p, \varepsilon, A \rangle, \langle q, B_1, B_2 \dots B_n \rangle \rangle \in \delta$  means:
  - In state  $p$  with  $A$  on top of the stack, *without reading anything*:
    - Go to state  $q$ ,
    - change the stack top  $A$  to  $B_1 B_2 \dots B_n$

# Example 3

- The PDA in Example 1 can formally be specified as
  - $\langle \{q_1, q_2\}, \{0, 1, c\}, \{B, G, R\}, \delta, q_1, R, \emptyset \rangle$ ,
  - where  $\delta$  is defined according to the table.

Top of stack	State	Symbol read		
		0	1	c
B	q <sub>1</sub>	PushB q <sub>1</sub>	PushG q <sub>1</sub>	q <sub>2</sub>
	q <sub>2</sub>	Pop q <sub>2</sub>		
G	q <sub>1</sub>	PushB q <sub>1</sub>	PushG q <sub>1</sub>	q <sub>2</sub>
	q <sub>2</sub>		Pop q <sub>2</sub>	
R	q <sub>1</sub>	PushB q <sub>1</sub>	PushG q <sub>1</sub>	q <sub>2</sub>
	q <sub>2</sub>	Without reading: Pop q <sub>2</sub>		

# Configurations

- What we handled in Example 2 are called *configurations*.
- **Definition 2.** A configuration is a triple  $\langle q, x, \gamma \rangle$ , where:
  - $q = \text{state} \in Q$
  - $x = \text{string} \in \Sigma^*$
  - $\gamma = \text{stack} \in \Gamma^*$
- **Definition 3.** The next-configuration relation  $\rightarrow$ 
  - If  $\langle \langle p, a, A \rangle, \langle q, \gamma \rangle \rangle \in \delta$  then  $\langle p, ay, A\beta \rangle \rightarrow \langle q, y, \gamma\beta \rangle$
  - If  $\langle \langle p, \varepsilon, A \rangle, \langle q, \gamma \rangle \rangle \in \delta$  then  $\langle p, y, A\beta \rangle \rightarrow \langle q, y, \gamma\beta \rangle$

State	Remaining string	Stack
q <sub>1</sub>	01c10	R
q <sub>1</sub>	1c10	BR
q <sub>1</sub>	c10	GBR
q <sub>2</sub>	10	GBR
q <sub>2</sub>	0	BR
q <sub>2</sub>		R
q <sub>2</sub>		

From Example 2:

$\langle q_1, 01c01, R \rangle \rightarrow \langle q_1, 1c01, BR \rangle$

# Acceptance

- Two different modes of acceptance are used in the literature.
- A PDA can accept a string if
  - the *stack is empty*, or
  - if it reaches a *final state*.
- **Definition 4**
  - A PDA accepts the string  $x$  if  $\langle s, x, \perp \rangle \rightarrow^* \langle q, \varepsilon, \gamma \rangle$  when  $q \in F$ .
- **Definition 5**
  - A PDA accepts the string  $x$  if  $\langle s, x, \perp \rangle \rightarrow^* \langle q, \varepsilon, \varepsilon \rangle$  when  $q \in Q$

# Acceptance, cont.

- Two different modes of acceptance are used in the literature.
- A PDA can accept a string if
  - the *stack is empty*, or
  - if it reaches a *final state*.
- **Definition 6.** The language of a PDA.
  - $M =$  some PDA
  - $L(M) =$  the set of all strings that are accepted by  $M$ .

# Equivalence between ways of acceptance

- **We will prove that:**
  - *if* there is a PDA accepting the string  $x$  with empty stack,
  - *then* there is a PDA accepting it in final state, and vice versa.
- As the two ways of accepting a string are *equally powerful*,
  - they define the *same* class of languages.
- We will perform the two proofs (the two directions of the implications making up the equivalence) somewhat in parallel.
- Our starting point is the PDA  $M = \langle Q, \Sigma, \Gamma, \delta, s, \perp, F \rangle$ .
  - It accepts strings in either way.

# Equivalence between ways of acceptance

- Two new sets are defined.
- If M accepts with **empty stack**:
  - $G = Q$
  - $\Delta = \{\perp\}$
- If M accepts in **final state**:
  - $G = F$
  - $\Delta = \Gamma \cup \{\perp\}$

# Equivalence between ways of acceptance

- We now add some extra handling in the start and in the end.
- We define a new PDA  $M' = \langle Q \cup \{u, t\}, \Sigma, \Gamma \cup \{\perp\}, \delta', u, \perp, \{t\} \rangle$ , where
  - $u$  = new start state
  - $t$  = new final state
  - $\perp$  = new stack bottom symbol
  - $\delta'$  = new transition relation, defined as:
    - $\delta' = \delta \cup \{ \langle \langle u, \varepsilon, \perp \rangle, \langle s, \perp \perp \rangle \rangle,$
    - $\langle \langle q, \varepsilon, A \rangle, \langle t, A \rangle \rangle \quad \text{for } q \in Q, A \in \Delta,$
    - $\langle \langle t, \varepsilon, A \rangle, \langle t, \varepsilon \rangle \rangle \quad \text{for } A \in \Gamma \cup \{\perp\} \quad \}$

# Equivalence between ways of acceptance

- $\delta' = \delta \cup \{ \langle \langle u, \varepsilon, \perp \rangle, \langle s, \perp, \perp \rangle \rangle, \langle \langle q, \varepsilon, A \rangle, \langle t, A \rangle \rangle \text{ for } q \in G, A \in \Delta, \langle \langle t, \varepsilon, A \rangle, \langle t, \varepsilon \rangle \rangle \text{ for } A \in \Gamma \cup \{ \perp \} \}$
- The first new element of  $\delta'$  says that in  $M'$ , you first just go from its start state to the start state of  $M$  and you put the stack bottom of  $M$  on the stack.
  - To prepare for the simulation of  $M$ .
- The next new part of  $\delta'$  says that when you are in an accepting situation in  $M$  you go to the accepting state of  $M'$ .
- The third part says that if there is anything left on the stack you could remove it.
- So,  $M'$  accepts with empty stack and in final state.

# Equivalence between ways of acceptance

- **Lemma 1.** If  $M$  accepts  $x$  with empty stack then  $M'$  accepts it.
- **Proof.**  $M$  accepts with empty stack:  $\langle s, x, \perp \rangle \xrightarrow{n} M \langle q, \varepsilon, \varepsilon \rangle$ .
  - Then  $\langle u, x, \perp \rangle \xrightarrow{1} M' \langle s, x, \perp \perp \rangle \xrightarrow{n} M' \langle q, \varepsilon, \perp \rangle \xrightarrow{1} M' \langle t, \varepsilon, \perp \rangle \xrightarrow{1} M' \langle t, \varepsilon, \varepsilon \rangle$ .
    1. The first step as the new elements of  $\delta'$  states,
    2. the second step since  $\delta'$  contains  $\delta$ ,
    3. the last two steps as the new elements of  $\delta'$  states.
  - So,  $\langle u, x, \perp \rangle \xrightarrow{*} M' \langle t, \varepsilon, \varepsilon \rangle$ , i.e.  $M'$  accepts  $x$  if  $M$  accepts it with empty stack.

# Equivalence between ways of acceptance

- **Lemma 2.** If  $M$  accepts  $x$  in final state then  $M'$  accepts it.
- **Proof.**  $M$  accepts in final state:  $\langle s, x, \perp \rangle \xrightarrow{n} M \langle q, \varepsilon, \gamma \rangle$ , where  $q \in F$ .
  - Then  $\langle u, x, \perp \rangle \xrightarrow{1} M' \langle s, x, \perp \perp \rangle \xrightarrow{n} M' \langle q, \varepsilon, \gamma \perp \rangle \xrightarrow{1} M' \langle t, \varepsilon, \gamma \perp \rangle \xrightarrow{*} M' \langle t, \varepsilon, \varepsilon \rangle$ .
    1. The first step as the new elements of  $\delta'$  states,
    2. the second step since  $\delta'$  contains  $\delta$ ,
    3. the last two steps as the new elements of  $\delta'$  states.

# Equivalence between ways of acceptance

- **Lemma 3.** If  $M$  accepts  $x$  then  $M'$  accepts it.
  - **Proof.** Follows from Lemma 1 and Lemma 2

# Equivalence between ways of acceptance

- **Lemma 4.** If  $M'$  accepts  $x$  then  $M$  accepts it.
- **Proof:**
- $\langle u, x, \perp \rangle \xrightarrow{1} M' \langle s, x, \perp \perp \rangle \xrightarrow{n} M' \langle q, y, \gamma \perp \rangle \xrightarrow{1} M' \langle t, y, \gamma \perp \rangle \xrightarrow{*} M' \langle t, \varepsilon, \varepsilon \rangle$ 
  1. The first step is the same initial one again.
  2. For  $M'$  to accept  $x$  it must in the process have reached a state  $q \in G$  since that is the only way  $\delta'$  can get the automaton into the state  $t$ . That is shown in step 2, with a  $y$  that maybe not is  $\varepsilon$ .
  3. The third step is just a move from  $q$  to  $t$ .
  4. Since we know that  $M'$  accepts  $x$  there must be some way to perform the last step. But in state  $t$   $M$  cannot read anything, so  $y$  must be  $\varepsilon$ . So if the second step is read from the  $M$  perspective we get  $\langle s, x, \perp \rangle \xrightarrow{n} M \langle q, \varepsilon, \gamma \rangle$ , i.e.  $M$  accepts  $x$ .

# Equivalence between ways of acceptance

- **Theorem 1.**
  - Accepting with empty stack and accepting in final state are equivalent.
  - *Proof.* It follows from Lemma 3 and Lemma 4

# Coming up

- This week
  - **Wednesday:** Pushdown automata (PDA) **Done!**
  - **Friday:** Equivalence between CFG and PDA
- Later
  - Properties of CFGs and parsing methods for CFGs

Thanks for today!