

# Theory of Computer Science

## B7. Context-free Languages: Normal Form and PDA

Gabriele Röger

University of Basel

March 29, 2021

# Context-free Grammars

# Repetition: Context-free Grammars

## Definition (Context-free Grammar)

A **context-free grammar** is a 4-tuple  $\langle V, \Sigma, R, S \rangle$  with

- 1  $V$  finite set of variables,
- 2  $\Sigma$  finite alphabet of terminal symbols (with  $V \cap \Sigma = \emptyset$ ),
- 3  $R \subseteq V \times (V \cup \Sigma)^*$  finite set of rules,
- 4  $S \in V$  start variable.

## Short-hand Notation for Rule Sets

We abbreviate several rules with the same left-hand side variable in a single line, using “|” for separating the right-hand sides.

For example, we write

$$X \rightarrow 0Y1 \mid XY$$

for:

$$X \rightarrow 0Y1 \text{ and}$$

$$X \rightarrow XY$$

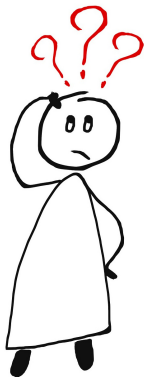
## Context-free Grammars: Exercise

We have used the pumping lemma for regular languages to show that  $L = \{a^n b^n \mid n \in \mathbb{N}_0\}$  is not regular.

Show that it is context-free by specifying a suitable grammar  $G$  with  $\mathcal{L}(G) = L$ .



# Questions



Questions?

# Chomsky Normal Form

# Chomsky Normal Form: Motivation

As in other kinds of structured objects, **normal forms** for grammars are useful:

- they show which aspects are critical for defining grammars and which ones are just syntactic sugar
- they allow proofs and algorithms to be restricted to a limited set of grammars (inputs): those in normal form

Hence we now consider a **normal form** for context-free grammars.

# Chomsky Normal Form: Definition

## Definition (Chomsky Normal Form)

A context-free grammar  $G$  is in **Chomsky normal form (CNF)** if all rules have one of the following three forms:

- $A \rightarrow BC$  with variables  $A, B, C$  and  $B$  and  $C$  are not the start variable, or
- $A \rightarrow a$  with variable  $A$  and terminal symbol  $a$ , or
- $S \rightarrow \varepsilon$  with start variable  $S$ .

**German:** Chomsky-Normalform

formally: rule set  $R \subseteq (V \times ((V \setminus \{S\})(V \setminus \{S\}) \cup \Sigma)) \cup \{(S, \varepsilon)\}$

# Chomsky Normal Form: Theorem

## Theorem

*For every context-free grammar  $G$  there is a context-free grammar  $G'$  in Chomsky normal form with  $\mathcal{L}(G) = \mathcal{L}(G')$ .*

# Chomsky Normal Form: Theorem

## Theorem

*For every context-free grammar  $G$  there is a context-free grammar  $G'$  in Chomsky normal form with  $\mathcal{L}(G) = \mathcal{L}(G')$ .*

## Proof.

The following algorithm converts the rule set of  $G = \langle V, \Sigma, R, S \rangle$  into CNF:

**Step 1:** Add new start variable  $S'$ .

Add a new variable  $S'$  which will be the start variable, and add a rule  $S' \rightarrow S$ , where  $S$  is the original start variable.

Afterwards, the (new) start variable does not occur on the right-hand side of a rule.

We will write  $V'$  for the new variable set ( $V' = V \cup \{S'\}$ ) and  $R'$  for the new rule set.

...

# Chomsky Normal Form: Theorem

Proof (continued).

Step 2: Eliminate  $\varepsilon$ -rules of the form  $A \rightarrow \varepsilon$  ( $A \neq S'$ ).

# Chomsky Normal Form: Theorem

## Proof (continued).

Step 2: Eliminate  $\varepsilon$ -rules of the form  $A \rightarrow \varepsilon$  ( $A \neq S'$ ).

- Let  $V_\varepsilon$  be the set of variable from which one can derive the empty word. We find this set  $V_\varepsilon$  by first collecting all variables  $A \in V'$  with rule  $A \rightarrow \varepsilon \in R'$  and then successively adding additional variables  $B$  if there is a rule  $B \rightarrow A_1 A_2 \dots A_k \in R'$  and the variables  $A_i$  are already in the set for all  $1 \leq i \leq k$ .

# Chomsky Normal Form: Theorem

## Proof (continued).

Step 2: Eliminate  $\varepsilon$ -rules of the form  $A \rightarrow \varepsilon$  ( $A \neq S'$ ).

- Let  $V_\varepsilon$  be the set of variable from which one can derive the empty word. We find this set  $V_\varepsilon$  by first collecting all variables  $A \in V'$  with rule  $A \rightarrow \varepsilon \in R'$  and then successively adding additional variables  $B$  if there is a rule  $B \rightarrow A_1 A_2 \dots A_k \in R'$  and the variables  $A_i$  are already in the set for all  $1 \leq i \leq k$ .
- Add rules that obviate the need for  $A \rightarrow \varepsilon$  rules:  
for every existing rule  $B \rightarrow w \in R'$  with  $B \in V'$ ,  
 $w \in (V' \cup \Sigma)^+$ , let  $I_\varepsilon$  be the set of positions where  $w$  contains a variable  $A \in V_\varepsilon$ . For every non-empty set  $I' \subseteq I_\varepsilon$ , add a new rule  $B \rightarrow w'$ , where  $w'$  is constructed from  $w$  by removing the variables at all positions in  $I'$ .

# Chomsky Normal Form: Theorem

## Proof (continued).

Step 2: Eliminate  $\varepsilon$ -rules of the form  $A \rightarrow \varepsilon$  ( $A \neq S'$ ).

- Let  $V_\varepsilon$  be the set of variable from which one can derive the empty word. We find this set  $V_\varepsilon$  by first collecting all variables  $A \in V'$  with rule  $A \rightarrow \varepsilon \in R'$  and then successively adding additional variables  $B$  if there is a rule  $B \rightarrow A_1 A_2 \dots A_k \in R'$  and the variables  $A_i$  are already in the set for all  $1 \leq i \leq k$ .
- Add rules that obviate the need for  $A \rightarrow \varepsilon$  rules:  
for every existing rule  $B \rightarrow w \in R'$  with  $B \in V'$ ,  
 $w \in (V' \cup \Sigma)^+$ , let  $I_\varepsilon$  be the set of positions where  $w$  contains a variable  $A \in V_\varepsilon$ . For every non-empty set  $I' \subseteq I_\varepsilon$ , add a new rule  $B \rightarrow w'$ , where  $w'$  is constructed from  $w$  by removing the variables at all positions in  $I'$ .
- Remove all rules of the form  $A \rightarrow \varepsilon$  ( $A \neq S'$ ).

## Step 2: Example

Consider  $G = \langle \{X, Y, Z, S\}, \{a, b\}, R, S \rangle$  with rules:

$$S \rightarrow \varepsilon \mid XY$$

$$X \rightarrow aXYbX \mid YZ \mid ab$$

$$Y \rightarrow \varepsilon \mid b$$

$$Z \rightarrow \varepsilon \mid a$$

# Chomsky Normal Form: Theorem

Proof (continued).

**Step 3: Eliminate rules of the form  $A \rightarrow B$**  with variables  $A, B$ .

If there are sets of variables  $\{B_1, \dots, B_k\}$  with rules

$B_1 \rightarrow B_2, B_2 \rightarrow B_3, \dots, B_{k-1} \rightarrow B_k, B_k \rightarrow B_1,$

then replace these variables by a new variable  $B$ .

We use  $V''$  to denote the resulting set of variables.

Define a strict total order  $<$  on the variables such that a rule  $A \rightarrow B$  implies that  $A < B$ . Iterate from the largest to the smallest variable  $A$  and eliminate all rules of the form  $A \rightarrow B$  while adding rules  $A \rightarrow w$  for every rule  $B \rightarrow w$  with  $w \in (V'' \cup \Sigma)^+$ . ...

## Step 3: Example

Consider  $G = \langle \{X, Y, Z, S\}, \{a, b\}, R, S \rangle$  with rules:

$$S \rightarrow \varepsilon \mid X$$

$$X \rightarrow aZbY \mid Y \mid ab$$

$$Y \rightarrow Z \mid b$$

$$Z \rightarrow Y \mid bXa$$

# Chomsky Normal Form: Theorem

Proof (continued).

Step 4: Eliminate rules with terminal symbols on the right-hand side that do not have the form  $A \rightarrow a$ .

For every terminal symbol  $a \in \Sigma$  add a new variable  $A_a$  and the rule  $A_a \rightarrow a$ .

Replace all terminal symbols in all rules that do not have the form  $A \rightarrow a$  with the corresponding newly added variables. ...

# Chomsky Normal Form: Theorem

Proof (continued).

Step 5: Eliminate rules of the form  $A \rightarrow B_1 B_2 \dots B_k$  with  $k > 2$

For every rule of the form  $A \rightarrow B_1 B_2 \dots B_k$  with  $k > 2$ , add new variables  $C_2, \dots, C_{k-1}$  and replace the rule with

$$\begin{aligned} A &\rightarrow B_1 C_2 \\ C_2 &\rightarrow B_2 C_3 \\ &\vdots \\ C_{k-1} &\rightarrow B_{k-1} B_k \end{aligned}$$



# Chomsky Normal Form: Exercise

(Example taken from textbook by Sipser)

Consider  $G = \langle \{A, B, S\}, \{a, b\}, R, S \rangle$  with rules:

$$S \rightarrow ASA \mid aB$$

$$A \rightarrow B \mid S$$

$$B \rightarrow \varepsilon \mid b$$



Specify a grammar  $G'$  in CNF with  $\mathcal{L}(G') = \mathcal{L}(G)$ .

# Chomsky Normal Form: Length of Derivations

## Observation

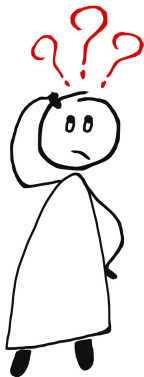
Let  $G$  be a grammar in Chomsky normal form,  
and let  $w \in \mathcal{L}(G)$  be a non-empty word generated by  $G$ .  
Then all derivations of  $w$  have exactly  $2|w| - 1$  derivation steps.

## Proof.

↔ Exercises



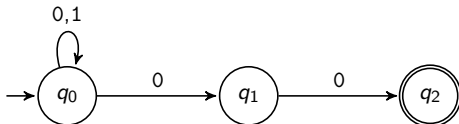
# Questions



Questions?

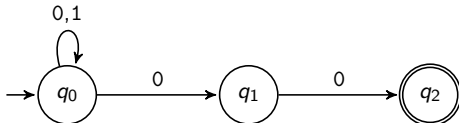
# Push-Down Automata

# Limitations of Finite Automata



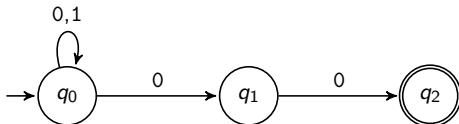
- Language  $L$  is regular.  
     $\iff$  There is a finite automaton that accepts  $L$ .

# Limitations of Finite Automata



- Language  $L$  is regular.
  - $\iff$  There is a finite automaton that accepts  $L$ .
- What information can a finite automaton “store” about the already read part of the word?

# Limitations of Finite Automata

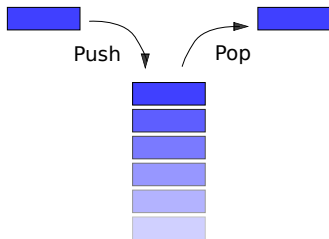


- Language  $L$  is regular.
  - $\iff$  There is a finite automaton that accepts  $L$ .
- What information can a finite automaton “store” about the already read part of the word?
- Infinite memory would be required for  $L = \{x_1x_2 \dots x_nx_n \dots x_2x_1 \mid n > 0, x_i \in \{a, b\}\}$ .
- therefore: extension of the automata model with memory

# Stack

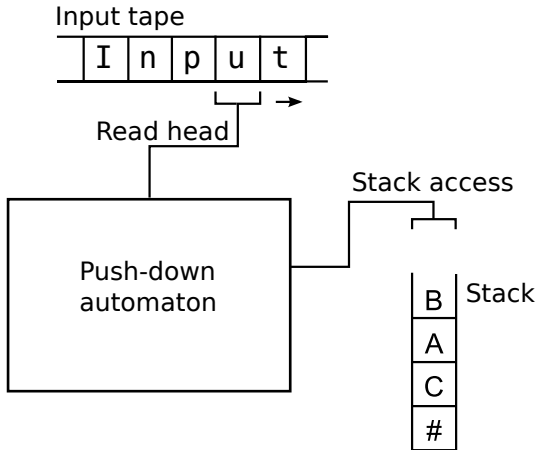
A **stack** is a data structure following the **last-in-first-out (LIFO)** principle supporting the following operations:

- **push**: puts an object on top of the stack
- **pop**: removes the object at the top of the stack



German: Keller, Stapel

# Push-down Automata: Visually



German: Kellerautomat, Eingabeband, Lesekopf, Kellerzugriff

## Push-down Automaton for $\{a^n b^n \mid n \in \mathbb{N}_0\}$ : Idea

- As long as you read symbols  $a$ , push an  $A$  on the stack.
- As soon as you read a symbol  $b$ , pop an  $A$  off the stack as long as you read  $b$ .
- If reading the input is finished exactly when the stack becomes empty, accept the input.
- If there is no  $A$  to pop when reading a  $b$ , or there is still an  $A$  on the stack after reading all input symbols, or if you read an  $a$  following a  $b$  then reject the input.

# Push-down Automata: Non-determinism

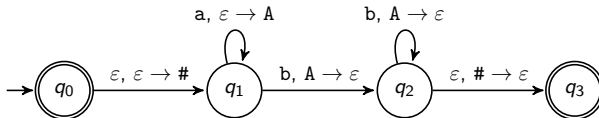
- PDAs are **non-deterministic** and can allow several next transitions from a configuration.
- Like NFAs, PDAs can have transitions that do not read a symbol from the input.
- Similarly, there can be transitions that do not pop and/or push a symbol off/to the stack.

# Push-down Automata: Non-determinism

- PDAs are **non-deterministic** and can allow several next transitions from a configuration.
- Like NFAs, PDAs can have transitions that do not read a symbol from the input.
- Similarly, there can be transitions that do not pop and/or push a symbol off/to the stack.

Deterministic variants of PDAs are strictly less expressive, i. e. there are languages that can be recognized by a (non-deterministic) PDA but not the deterministic variant.

# Push-down Automaton for $\{a^n b^n \mid n \in \mathbb{N}_0\}$ : Diagram



# Push-down Automata: Definition

## Definition (Push-down Automaton)

A **push-down automaton (PDA)** is a 6-tuple  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$  with

- $Q$  finite set of states
- $\Sigma$  the input alphabet
- $\Gamma$  the stack alphabet
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$  the transition function
- $q_0 \in Q$  the start state
- $F \subseteq Q$  is the set of **accept states**

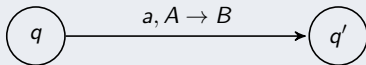
**German:** Kellerautomat, Eingabealphabet, Kelleralphabet, Überföhrungsfunktion

# Push-down Automata: Transition Function

Let  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$  be a push-down automaton.

What is the Intuitive Meaning of the Transition Function  $\delta$ ?

- $\langle q', B \rangle \in \delta(q, a, A)$ : If  $M$  is in state  $q$ , reads symbol  $a$  and has  $A$  as the topmost stack symbol, then  $M$  **can** transition to  $q'$  in the next step popping  $A$  off the stack and pushing  $B$  on the stack.

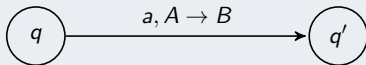


# Push-down Automata: Transition Function

Let  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$  be a push-down automaton.

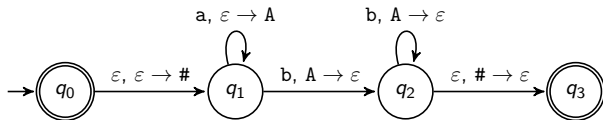
What is the Intuitive Meaning of the Transition Function  $\delta$ ?

- $\langle q', B \rangle \in \delta(q, a, A)$ : If  $M$  is in state  $q$ , reads symbol  $a$  and has  $A$  as the topmost stack symbol, then  $M$  **can** transition to  $q'$  in the next step popping  $A$  off the stack and pushing  $B$  on the stack.



- special case  $a = \varepsilon$  is allowed (spontaneous transition)
- special case  $A = \varepsilon$  is allowed (no pop)
- special case  $B = \varepsilon$  is allowed (no push)

# Push-down Automaton for $\{a^n b^n \mid n \in \mathbb{N}_0\}$ : Formally



$M = \langle \{q_0, q_1, q_2, q_3\}, \{a, b\}, \{A, \#\}, \delta, q_0, \{q_0, q_3\} \rangle$  with

$$\delta(q_0, a, A) = \emptyset$$

$$\delta(q_0, b, A) = \emptyset$$

$$\delta(q_0, \varepsilon, A) = \emptyset$$

$$\delta(q_0, a, \#) = \emptyset$$

$$\delta(q_0, b, \#) = \emptyset$$

$$\delta(q_0, \varepsilon, \#) = \emptyset$$

$$\delta(q_0, a, \varepsilon) = \emptyset$$

$$\delta(q_0, b, \varepsilon) = \emptyset$$

$$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \#)\}$$

$$\delta(q_1, a, A) = \emptyset$$

$$\delta(q_1, b, A) = \{(q_2, \varepsilon)\}$$

$$\delta(q_1, \varepsilon, A) = \emptyset$$

$$\delta(q_1, a, \#) = \emptyset$$

$$\delta(q_1, b, \#) = \emptyset$$

$$\delta(q_1, \varepsilon, \#) = \emptyset$$

$$\delta(q_1, a, \varepsilon) = \{(q_1, A)\}$$

$$\delta(q_1, b, \varepsilon) = \emptyset$$

$$\delta(q_1, \varepsilon, \varepsilon) = \emptyset$$

$$\delta(q_2, a, A) = \emptyset$$

$$\delta(q_2, b, A) = \{(q_2, \varepsilon)\}$$

$$\delta(q_2, \varepsilon, A) = \emptyset$$

$$\delta(q_2, a, \#) = \emptyset$$

$$\delta(q_2, b, \#) = \emptyset$$

$$\delta(q_2, \varepsilon, \#) = \{(q_3, \varepsilon)\}$$

$$\delta(q_2, a, \varepsilon) = \emptyset$$

$$\delta(q_2, b, \varepsilon) = \emptyset$$

$$\delta(q_2, \varepsilon, \varepsilon) = \emptyset$$

and  $\delta(q_3, x, y) = \emptyset$  for all  $x \in \{a, b, \varepsilon\}$ ,  $y \in \{A, \#, \varepsilon\}$

# Push-down Automata: Accepted Words

## Definition

A PDA  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$  **accepts input  $w$**  if it can be written as  $w = w_1 w_2 \dots w_m$  where each  $w_i \in \Sigma \cup \{\varepsilon\}$

## Push-down Automata: Accepted Words

### Definition

A PDA  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$  **accepts input  $w$**  if it can be written as  $w = w_1 w_2 \dots w_m$  where each  $w_i \in \Sigma \cup \{\varepsilon\}$  and sequences of states  $r_0, r_1, \dots, r_m \in Q$  and strings  $s_0, s_1, \dots, s_m \in \Gamma^*$  exist that satisfy the following three conditions:

The strings  $s_i$  represent the sequence of stack contents.

# Push-down Automata: Accepted Words

## Definition

A PDA  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$  **accepts input  $w$**  if it can be written as  $w = w_1 w_2 \dots w_m$  where each  $w_i \in \Sigma \cup \{\varepsilon\}$  and sequences of states  $r_0, r_1, \dots, r_m \in Q$  and strings  $s_0, s_1, \dots, s_m \in \Gamma^*$  exist that satisfy the following three conditions:

- 1  $r_0 = q_0$  and  $s_0 = \varepsilon$

The strings  $s_i$  represent the sequence of stack contents.

# Push-down Automata: Accepted Words

## Definition

A PDA  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$  **accepts input  $w$**  if it can be written as  $w = w_1 w_2 \dots w_m$  where each  $w_i \in \Sigma \cup \{\varepsilon\}$  and sequences of states  $r_0, r_1, \dots, r_m \in Q$  and strings  $s_0, s_1, \dots, s_m \in \Gamma^*$  exist that satisfy the following three conditions:

- 1  $r_0 = q_0$  and  $s_0 = \varepsilon$
- 2 For  $i = 0, \dots, m - 1$ , we have  $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ , where  $s_i = at$  and  $s_{i+1} = bt$  for some  $a, b \in \Gamma \cup \{\varepsilon\}$  and  $t \in \Gamma^*$ .

The strings  $s_i$  represent the sequence of stack contents.

# Push-down Automata: Accepted Words

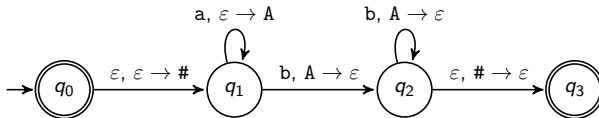
## Definition

A PDA  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$  **accepts input  $w$**  if it can be written as  $w = w_1 w_2 \dots w_m$  where each  $w_i \in \Sigma \cup \{\varepsilon\}$  and sequences of states  $r_0, r_1, \dots, r_m \in Q$  and strings  $s_0, s_1, \dots, s_m \in \Gamma^*$  exist that satisfy the following three conditions:

- 1  $r_0 = q_0$  and  $s_0 = \varepsilon$
- 2 For  $i = 0, \dots, m - 1$ , we have  $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ , where  $s_i = at$  and  $s_{i+1} = bt$  for some  $a, b \in \Gamma \cup \{\varepsilon\}$  and  $t \in \Gamma^*$ .
- 3  $r_m \in F$

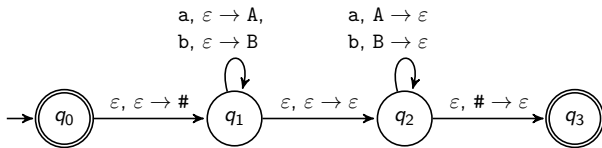
The strings  $s_i$  represent the sequence of stack contents.

# Push-down Automaton for $\{a^n b^n \mid n \in \mathbb{N}_0\}$



The PDA accepts input aabb.

# Acceptance: Exercise



Show that this PDA accepts input  $abba$ .

# PDA: Recognized Language

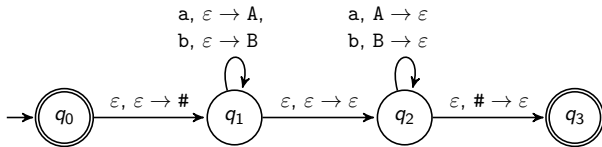
## Definition (Language Recognized by an NFA)

Let  $M$  be a PDA with input alphabet  $\Sigma$ .

The **language recognized by  $M$**  is defined as

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ is accepted by } M\}.$$

# Recognized Language: Exercise



What language does this PDA recognize?

# PDAs Recognize Exactly the Context-free Languages

## Theorem

*A language  $L$  is context-free if and only if  $L$  is recognized by a push-down automaton.*

## PDAs: Exercise (if time)

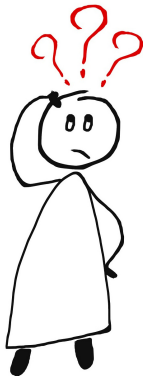
Assume you want to have a possible transition from state  $q$  to state  $q'$  in your PDA that

- processes symbol  $c$  from the input word,
- can only be taken if the top stack symbol is  $A$ ,
- does **not** pop  $A$  off the stack, and
- pushes  $B$ .



What problem do you encounter? How can you work around it?

# Questions



Questions?

# Summary

# Summary

- Every context-free language has a grammar in **Chomsky normal form**. All rules have form
  - $A \rightarrow BC$  with variables  $A, B, C$  ( $B, C$  not start variable), or
  - $A \rightarrow a$  with variable  $A$ , terminal symbol  $a$ , or
  - $S \rightarrow \varepsilon$  with start variable  $S$ .
- **Push-down automata** (PDAs) extend NFAs with memory.
- The **languages recognized by PDAs** are exactly the **context-free languages**.