

## Languages and Machines (Module 7 TCS+IAM)

L&M 7: Beyond Context-Free Languages:  
Pumping Lemma for CFG and Turing Machines

Ch 7:4-5; Ch 8:1-7

Alexander Skopalik

Discrete Mathematics and Mathematical Programming, Applied Mathematics, University  
of Twente

Lecture 7

## Contents

- 1 Pumping lemma for Context-free Languages
- 2 Class of Context-free Languages
- 3 Beyond Context-free: Turing Machines
- 4 Variations on Turing Machines
  - Multi-tape Turing Machines
  - Nondeterministic Turing Machines

## Context-Free Grammars and Pushdown Automata

## Parsing Algorithm (parse tree)

- Naive parsing: inefficient
- CYK parsing: dynamic programming, cubic in input length

## Chomsky normal form

- Rules  $A \rightarrow BC$ ,  $A \rightarrow a$  or  $S \rightarrow \lambda$
- Systematically constructed through transformation

## Greibach normal form

- Rules  $A \rightarrow aB_1B_2 \dots B_n$  or  $S \rightarrow \lambda$
- Systematically constructed through transformation

## Push-down automata

- Finite automata + unbounded stack (push/pop)
- Acceptance by “final state” and “empty stack”
- Recognize precisely the context-free languages

## Contents

- 1 Pumping lemma for Context-free Languages
- 2 Class of Context-free Languages
- 3 Beyond Context-free: Turing Machines
- 4 Variations on Turing Machines
  - Multi-tape Turing Machines
  - Nondeterministic Turing Machines

## Non-context-free languages

- Not all languages (sets of strings over  $\Sigma$ ) are context-free
- How can we prove that a given language is not context-free?
- Answer: *pumping lemma*, similar to those for regular languages

### Reminder: pumping lemma regular languages

If a language  $X$  is **regular**, then

- there exists a  $k \geq 1$ , such that
- for every  $z \in X$  with  $|z| > k$ ,
- there exist  $u, v, w \in \Sigma^*$ , such that  $z = uvw$ ,  $|uv| \leq k$ ,  $|v| > 0$
- and for every  $i \geq 0$ , we have  $uv^i w \in X$

### Pumping lemma context-free languages

If a language  $X$  is **context-free**,

- there exists a  $k \geq 1$ , such that
- for every  $z \in X$  with  $|z| > k$ , there
- exist  $u, v, w, x, y \in \Sigma^*$ , with  $z = uvwxy$ ,  $|vwx| \leq k$ ,  $|vx| > 0$
- and for every  $i \geq 0$ , we have  $uv^i wx^i y \in X$

## Example of the context-free pumping game

Consider language  $X = \{a^i b^i c^i \mid i \geq 0\}$ .  $X$  is *not* context-free:

- Let  $k$  be arbitrarily given,
- Choose  $z = a^k b^k c^k$
- Let  $uvwxy = z$  such that  $|vx| > 0$  and  $|vwx| \leq k$
- We distinguish the following possible cases:
  - 1  $v$  and  $x$  both contain one (repeated) symbol from  $\{a, b, c\}$ 
    - Then  $v \in \{a^m, b^m, c^m\}$  and  $x \in \{a^n, b^n, c^n\}$  with  $m + n = |vx| > 0$
    - Let  $\{\alpha, \beta, \gamma\} \subseteq \{a, b, c\}$  such that  $v = \alpha^m$  and  $x = \beta^n$  and  $\gamma \notin \{\alpha, \beta\}$ .
    - Then  $uv^2 wx^2 y$  contains the same number of  $\gamma$ 's as  $z$  does, but more  $\alpha$ 's and/or  $\beta$ 's
    - Choose  $i = 2$ ; then  $uv^i wx^i y \notin X$
  - 2  $v$  or  $x$  contains (at least) 2 different symbols from  $\{a, b, c\}$ 
    - Then  $v^2$  or  $x^2$  contains a  $b$  or  $c$  before an  $a$  or a  $c$  before a  $b$ ;
    - Choose  $i = 2$ ; then  $uv^2 wx^2 y \notin X$

So by the pumping lemma,  $X$  is not context-free.

## Using the context-free pumping lemma

### Contraposition of the context-free pumping lemma

A set of strings  $X \subseteq \Sigma^*$  is *not* context-free if

- for all  $k \geq 1$
- at least one  $z \in X$  exists such that  $|z| > k$  and
- for all  $u, v, w, x, y \in \Sigma^*$  with  $z = uvwxy$ ,  $|vwx| \leq k$ , and  $|vx| > 0$
- at least one  $i \geq 0$  exists such that  $uv^i wx^i y \notin X$

The Pumping Lemma can be used, to prove that languages are not context-free, e.g.:

- $\{a^i b^i c^i \mid i \geq 0\}$
- $\{a^i b^j a^j b^i \mid i, j \geq 0\}$

## Contents

- 1 Pumping lemma for Context-free Languages
- 2 Class of Context-free Languages
- 3 Beyond Context-free: Turing Machines
- 4 Variations on Turing Machines
  - Multi-tape Turing Machines
  - Nondeterministic Turing Machines

## Closure properties context-free languages

### Reminder: closure properties regular languages

If  $X$  and  $Y$  are regular languages over alphabet  $\Sigma$ , then also

- Union:  $X \cup Y$
- Concatenation:  $XY = \{vw \mid v \in X, w \in Y\}$
- Kleene star:  $X^* = \{v_1 \cdots v_n \mid \forall i \in [1, n] : v_i \in X\}$
- Intersection:  $X \cap Y$
- Complement:  $\Sigma^* - X$

In other words, regular languages are *closed under* these operations.

Proof: by construction of automata and/or regular expressions

### Theorem: Closure properties context-free languages

CF languages are closed under union, concatenation & Kleene star

### Theorem: Non-closure properties context-free languages

CF languages are *not* closed under intersection & complement

## Proof of (non-)closure properties

Given grammars  $G_i = \langle V_i, \Sigma_i, P_i, S_i \rangle$  with  $L(G_i) = X_i$  ( $i = 1, 2$ )

### • Closed under union, concatenation and Kleene star

- $G_1 \cup G_2 = \langle V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \rightarrow S_1 | S_2\}, S \rangle$
- $G_1 G_2 = \langle V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S \rangle$
- $G_1^* = \langle V_1 \cup \{S\}, \Sigma_1, P_1 \cup \{S \rightarrow S_1 S | \lambda\}, S \rangle$

Then  $L(G_1 \cup G_2) = X_1 \cup X_2$ ,  $L(G_1 G_2) = X_1 X_2$  and  $L(G_1^*) = X_1^*$ .

### • Not closed under intersection

- $X_1 = \{a^i b^i c^i \mid i \geq 0\}$  and  $X_2 = \{a^i b^j c^i \mid i \geq 0\}$  are context-free
- $X_1 \cap X_2 = \{a^i b^i c^i \mid i \geq 0\}$  is not

### • Not closed under complement

- Notation:  $\bar{X} = \Sigma^* \setminus X$
- De Morgan:  $X_1 \cap X_2 = \overline{\bar{X}_1 \cap \bar{X}_2} = \overline{\bar{X}_1 \cup \bar{X}_2}$
- Closed under complement implies closed under intersection
- Contradiction, so CF languages are not closed under complement

## Another closure property

### Theorem

Intersection of a regular and a context-free language is context-free

### Proof (another application of parallel composition!)

Construct *product* of a PDA  $M$  and a DFA  $N$ :

$$M \times N = \langle Q_M \times Q_N, \Sigma_M \cup \Sigma_N, \Gamma_M, \delta, (q_{M,0}, q_{N,0}), F_M \times F_N \rangle$$

- $q_1 \xrightarrow{a \ B/C} q'_1$  with  $q_2 \xrightarrow{a} q'_2$  generates  $(q_1, q_2) \xrightarrow{a \ B/C} (q'_1, q'_2)$
- $q_1 \xrightarrow{\lambda \ B/C} q'_1$  generates  $(q_1, q_2) \xrightarrow{\lambda \ B/C} (q'_1, q_2)$

Now  $L(M \times N) = L(M) \cap L(N)$

- Application:  $L = \{ww \mid w \in \{a, b\}^*\}$  is not context-free
- Proof (by contradiction): Suppose that  $L$  is context-free.
  - Then  $L \cap a^* b^* a^* b^* = \{a^i b^j a^i b^j \mid i \geq 0\}$  is as well (theorem).
  - But this is not the case (pumping lemma): Contradiction
  - So  $L$  is not context-free either.

## Side note: Deterministic Context-Free Languages

$L$  is **deterministic context-free** if it is accepted by a deterministic PDA.

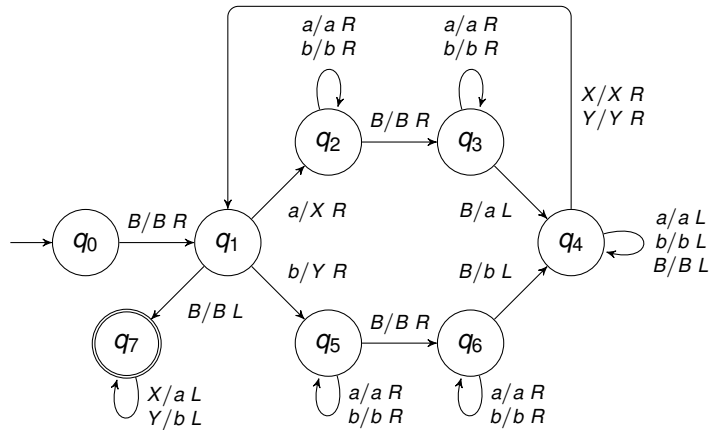
The situation for the class of DPDAs is very different:

- DPDAs are **weaker** than PDAs
- DPDAs are **NOT closed** under union
- DPDAs are **closed** under complement (this fact is very difficult to prove)
- **2002**: Geraud Senizergues receives Gödel Award for algorithm that decides if  $\mathcal{L}(M_1) = \mathcal{L}(M_2)$  for **deterministic** PDAs  $M_1, M_2$ .
- **Provable fact**: No algorithm exists that decides whether  $\mathcal{L}(M_1) = \mathcal{L}(M_2)$  for **arbitrary** PDAs  $M_1, M_2$ .



## Turing machine: Example 2

What does the following Turing machine achieve?



- Copies words over  $\{a, b\}^*$
- $q_0BaabaB \vdash \dots Bq_1XXYXBaabaB \vdash \dots Bq_7aabaBaabaB$

## Languages of Turing machines

Further terminology:

- $M$  recognizes a language  $L$  if  $\forall w: M \text{ accepts } w \Leftrightarrow w \in L$
- $M$  decides a language  $L$  if moreover all computations halt

**Notation:**  $L(M)$  is the language that is recognized by  $M$

**Definition:** let  $L \subseteq \Sigma^*$  be an arbitrary language

- $L$  is *recursively enumerable* if  $L$  is recognized by a Turing machine
- $L$  is *recursive* if  $L$  is decided by a Turing machine

## Acceptance by Turing machines

Further terminology:

- The word that is initially on the tape is called the *input* (from the first "real" position until the infinite series of  $B$ 's)
- The TM *halts* when no further transitions are possible (different from automata: just reaching a final state is not enough!)

### Definition

A Turing machine  $M$  *accepts* a word  $w$  if  $M$  halts in a final state when starting with input  $w$ .

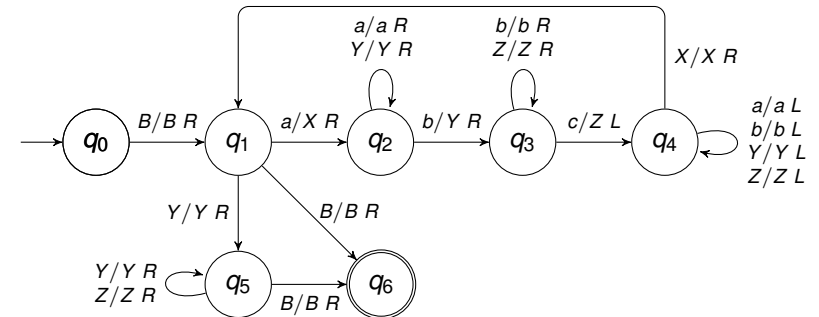
We do *not* pay attention to the contents of the tape after computation!  
We do *not* care if  $w$  is actually read at all.

There are three possible outcomes (for a given input):

- 1 The machine halts in a final state
- 2 The machine halts in a state that is not a final state
- 3 The machine never halts

## Recursive language: Example

Which language is *decided* by the following Turing machine:



- Computations:  $q_0BaabbccB \vdash^* BXXYYZZBq_6$   
 $q_0BaabccB \vdash^* BXXYq_2ZcB$
- Language  $\{a^k b^k c^k \mid k \geq 0\}$ : not context-free!
- Every PDA is convertible into a Turing machine (tape = stack)
- Recursive languages are strictly more powerful than context-free languages

# Contents

- 1 Pumping lemma for Context-free Languages
- 2 Class of Context-free Languages
- 3 Beyond Context-free: Turing Machines
- 4 Variations on Turing Machines
  - Multi-tape Turing Machines
  - Nondeterministic Turing Machines

# Variations on Turing Machines

## Standard Turing Machines (as defined so far)

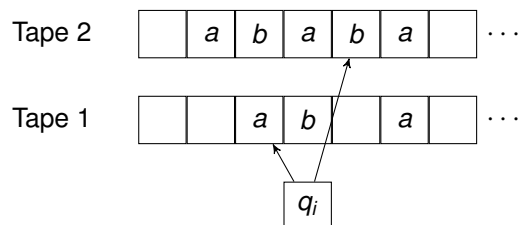
- Single tape
- Deterministic
- Acceptance by “halting” in a “final state”

## The following variations are all equivalent!

- Acceptance by “halting”
- Multi-track tape
- Two-way tape
- Multiple tapes
- Non-deterministic

# Playing with the tape: Multiple tapes

Instead of one tape we can also have multiple:



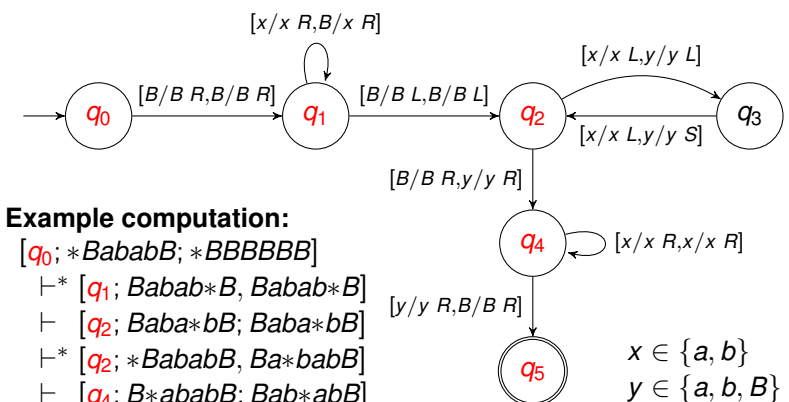
- Difference with multiple tracks: tape head for each tape
- Transitions  $q \xrightarrow{[x_1/y_1, d_1, x_2/y_2, d_2]} q'$
- $d_1, d_2 \in \{R, L, S\}$  may be different:  
 R (right), L (left), S (stationary)

Does this matter for expressiveness?

## Theorem

A language is accepted by a machine with multiple tapes if and only if it is recursively enumerable.

# Example of a machine with multiple tapes

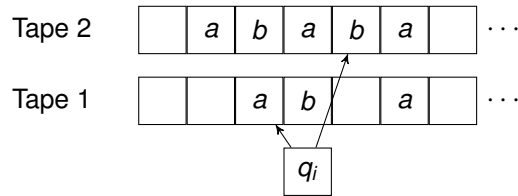


## Example computation:

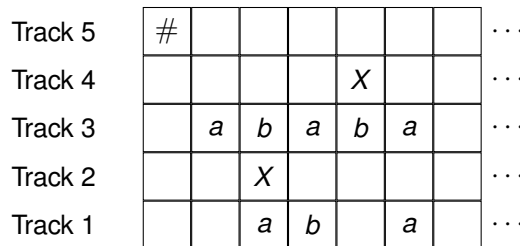
- $[q_0; *BababB; *BBBBBB]$
  - $\vdash^* [q_1; Babab*B, Babab*B]$
  - $\vdash [q_2; Baba*bB; Baba*bB]$
  - $\vdash^* [q_2; *BababB, Ba*babB]$
  - $\vdash [q_4; B*ababB; Bab*abB]$
  - $\vdash^* [q_4; Bab*abB; Babab*B]$
  - $\vdash [q_5; Baba*bB; BababB*]$
- $x \in \{a, b\}$   
 $y \in \{a, b, B\}$

Recognized language:  $\{ww \mid w \in \{a, b\}^*\}$

## Proof idea of equivalence Multi-tape machine



is coded on one tape with five tracks:



The functioning of the two-tape machine is now *simulated*.

## Expressiveness nondeterministic Turing machines

### Proof idea: Backtracking

Suppose that  $L$  is accepted by nondeterministic TM  $M$ .  
Now simulate  $M$  with a deterministic TM  $M'$  with 3 tapes:

- Tape 1: Remember the original input
- Tape 2: Record the current choices for backtracking  $(m_1, \dots, m_n)$
- Tape 3: Simulate  $n$  steps of  $M$ , according to choices on Tape 2

After the simulation of  $n$  steps: Replace Tape 2 by the following choice and restart Tape 3 with the contents of Tape 1.

Application: Recognize  $(a + b)^i b (a + b)^i$ : words with a  $b$  in the middle

## Nondeterminism

**Determinism:**  $\delta$  can be understood as a partial function

$$\delta : Q \times \Gamma \rightarrow \Gamma \times \{L, R\} \times Q$$

(Given  $q$  and  $x$  there exists at most one transition  $q \xrightarrow{x/y d} q'$ .)

**Nondeterminism:**  $\delta : Q \times \Gamma \rightarrow \mathcal{P}(\Gamma \times \{L, R\} \times Q)$

Does (non)determinism matter for expressiveness?

### Theorem

A language is accepted by a nondeterministic machine if and only if it is recursively enumerable.

## Summary so far

- **Turing machines**
  - Memory is an infinite tape instead of just a stack (as in PDA)
  - Recursively enumerable languages (those accepted by a TM)
    - TM halts and accepts if  $w \in L$  (yes/no/don't know)
  - Recursive languages (those decided by a TM)
    - TM decides the language, so it always halts (yes/no)
- **Variations**
  - Multi-tape Turing Machines
  - Nondeterministic Turing Machines

*All variations recognise recursively enumerable languages!*