Computational Complexity and Review

Formal Languages and Abstract Machines

Week 12

Baris E. Suzek, PhD

Outline

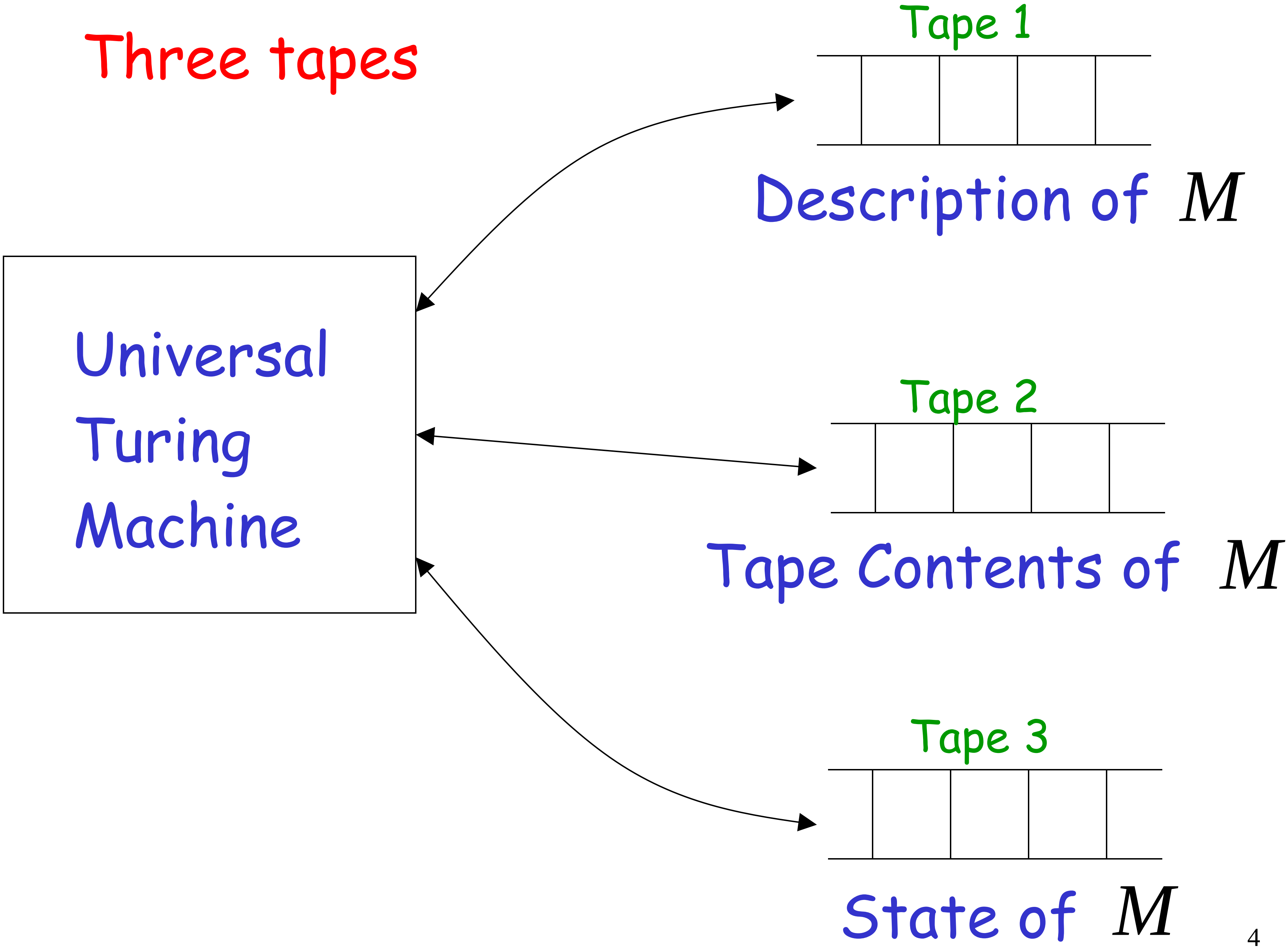
- Review of last week
- Computational Complexity
- Review

A limitation of Turing Machines:

$$\text{Turing Machines are “}\underbrace{\text{hardwired}}_{\substack{\text{they execute} \\ \text{only one program}}}\text{”}$$

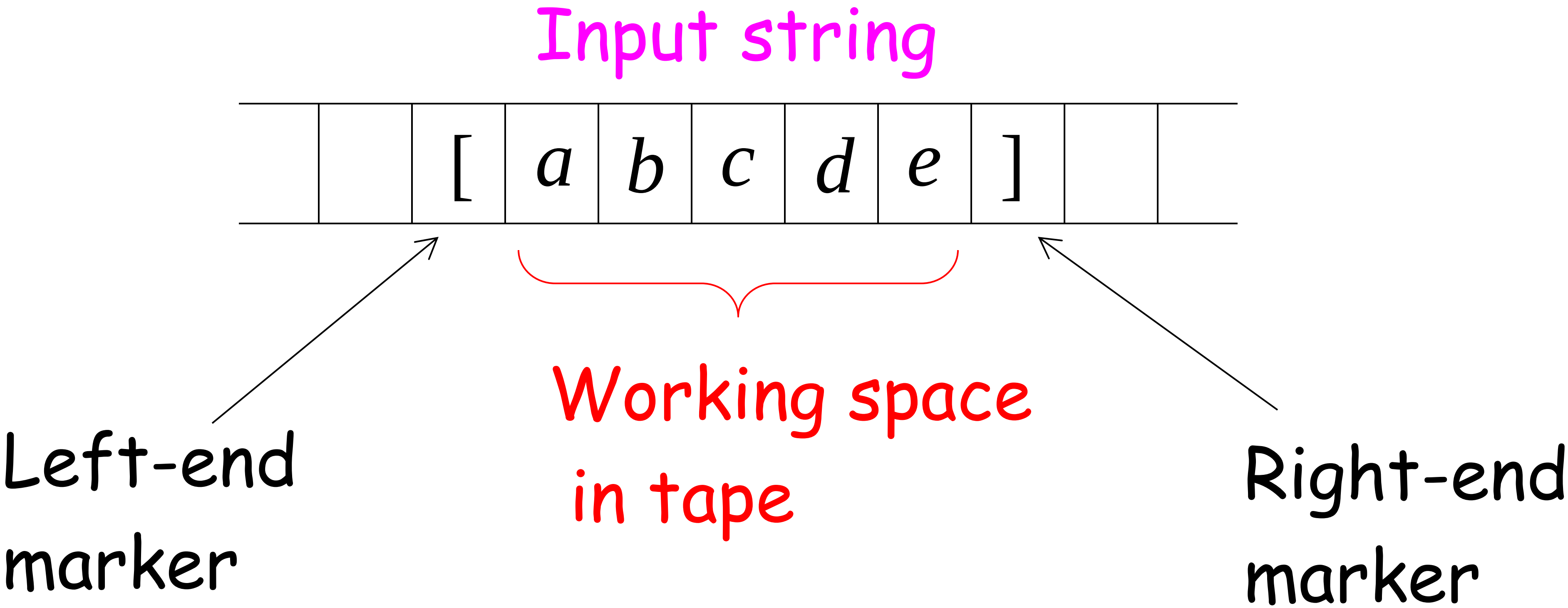
Real Computers are re-programmable

Three tapes



Tape 1
Description of M
Universal
Turing
Machine
Tape 2
Tape Contents of M
Tape 3
State of M

Linear Bounded Automaton (LBA)



All computation is done between end markers

Context-Sensitive Grammars:

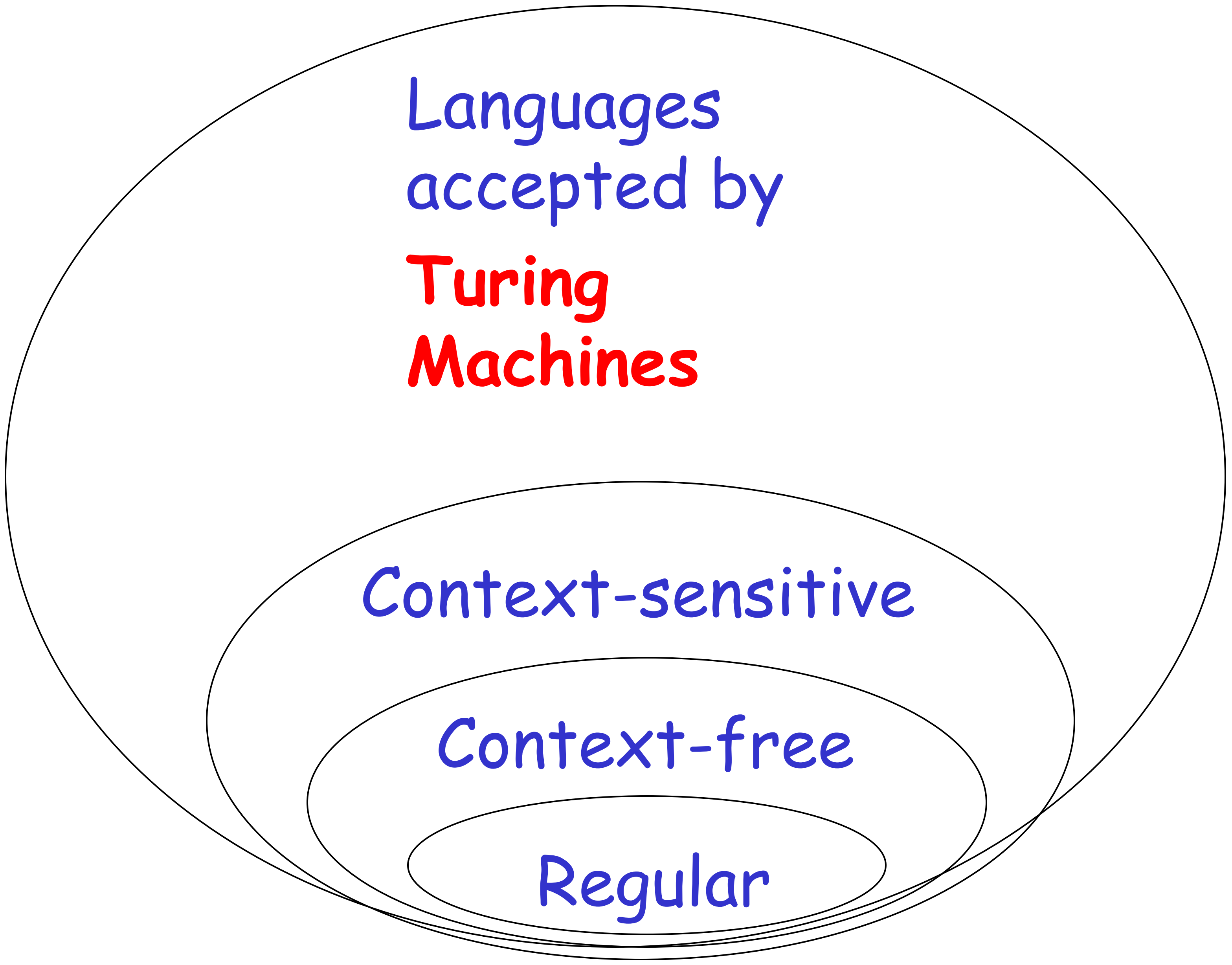
Productions

$$u \rightarrow v$$

String of variables and terminals

String of variables and terminals

and: $|u| \leq |v|$


Languages
accepted by
Turing
Machines
Context-sensitive
Context-free
Regular

Unrestricted Grammars:

Productions

$$u \rightarrow v$$

String of variables and terminals

String of variables and terminals

Definition:

A language is **recursively enumerable** if some Turing machine accepts it

	a^1	a^2	a^3	a^4	$\cdots$
$L(M_1)$	0	1	0	1	$\cdots$
$L(M_2)$	1	0	0	1	$\cdots$
$L(M_3)$	0	1	1	1	$\cdots$
$L(M_4)$	0	0	0	1	$\cdots$

$$\overline{L} = \{a^1, a^2, \ldots\}$$

Theorem:

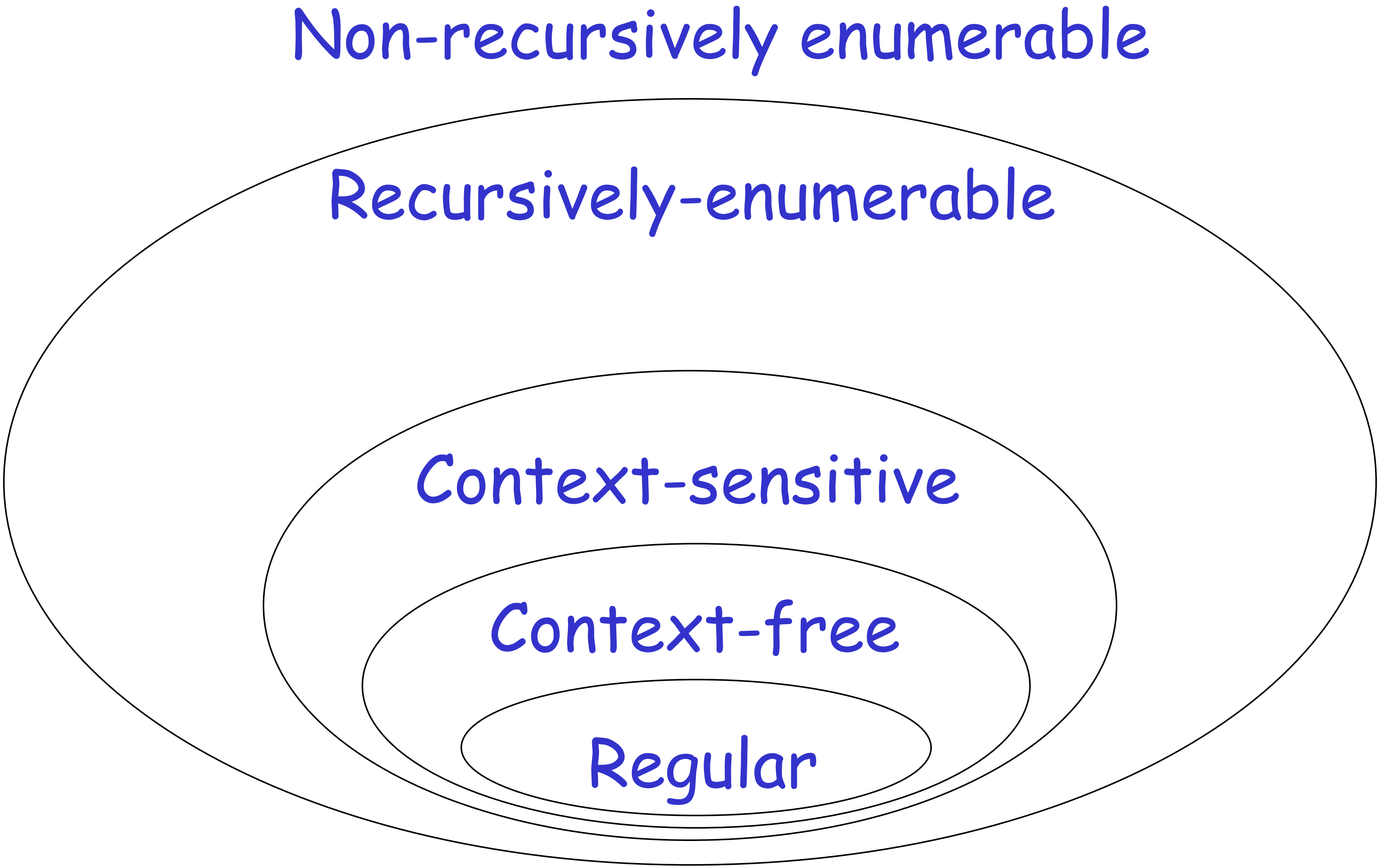
Language $\overline{L}$ is not recursively enumerable

Non Recursively Enumerable

$\overline{L}$

Recursively Enumerable

The Chomsky Hierarchy



Outline

- Review of last week
- Computational Complexity
- Review

Efficiency of a Computation

Time Complexity: The number of steps during a computation

Space Complexity: Space used during a computation

Studying Computational Complexity

- Model: Turing Machine
- Size of the problem: n
  - e.g. sorting n numbers.
- Analysis focus:
  - How algorithm behaves when the problem size changes?

Time Complexity

- We use a multitape Turing machine

- We count the number of steps until a string is accepted

- We use the $O(k)$ notation

Example: $L = \{a^n b^n : n \geq 0\}$

Algorithm to accept a string w :

- Use a two-tape Turing machine

- Copy the a on the second tape and remove from first

- Compare the a and b on both tapes

$$L = \{a^n b^n : n \geq 0\}$$

Time (moves) needed:

- Copy the a on the second tape $\qquad O(|w|)$

- Compare the a and b $\qquad O(|w|)$

Total time: $\qquad O(|w|)$

$$L = \{a^n b^n : n \geq 0\}$$

For string of length n

time needed for acceptance: $O(n)$

What will be the O(n) if single tape used?

Language class: $DTIME(n)$

$DTIME(n)$

L_1 L_2 L_3

A Deterministic Turing Machine accepts each string of length n in time $O(n)$

$$DTIME(n)$$

$$\{a^n b^n : n \geq 0\}$$

$$\{ww\}$$

In a similar way we define the class

$$DTIME(T(n))$$

for any time function: $T(n)$

Examples: $DTIME(n^2), DTIME(n^3), \ldots$

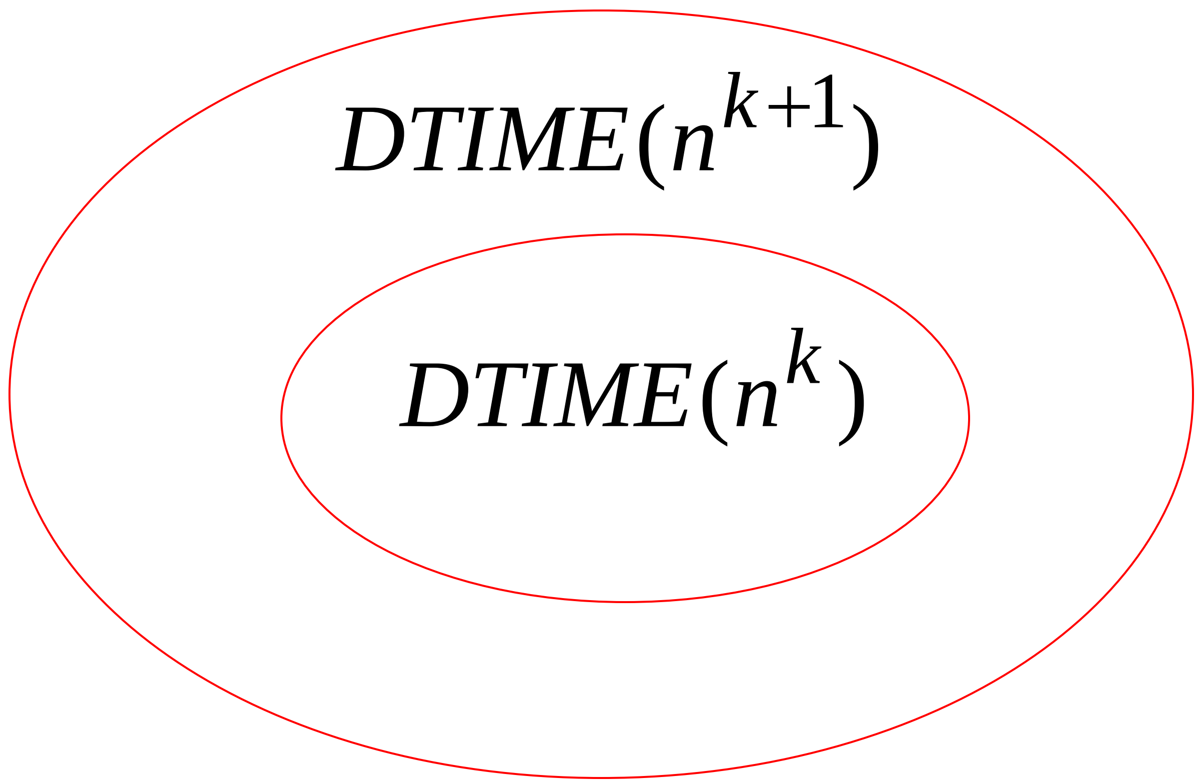
Example: The membership problem for context free languages

$$L = \{w : w \text{ is generated by grammar } G\}$$

$$L \in DTIME(n^3)$$ (CYK - algorithm)

Polynomial time

Theorem: $DTIME(n^{k+1}) \subset DTIME(n^k)$



Polynomial time algorithms: $DTIME(n^k)$

Represent tractable (solvable) algorithms:

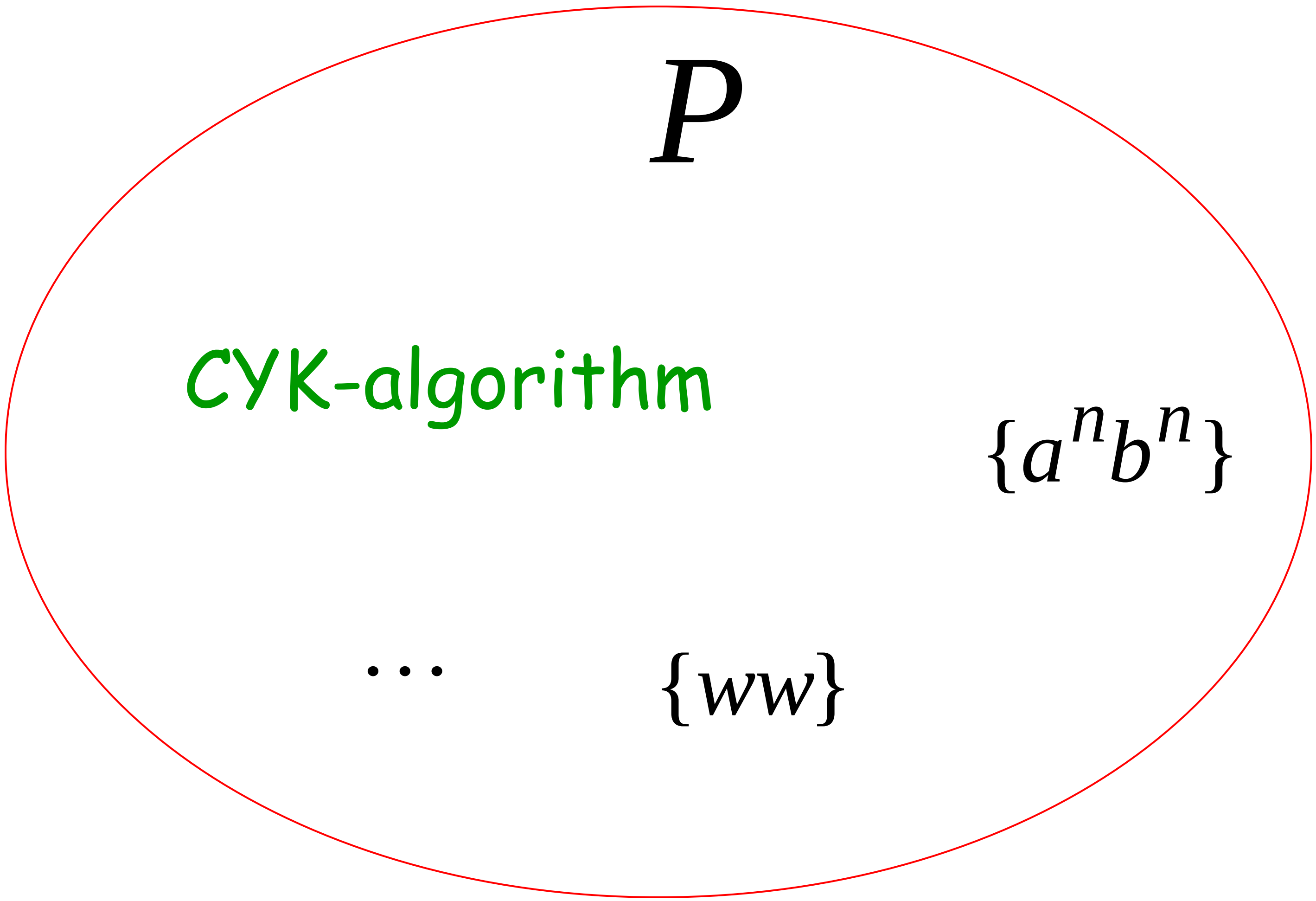
For small k we can compute the result fast

The class P

$$P = \cup DTIME(n^k) \qquad \text{for all} \quad k$$

- Polynomial time

- All tractable problems(can be solved effectively)

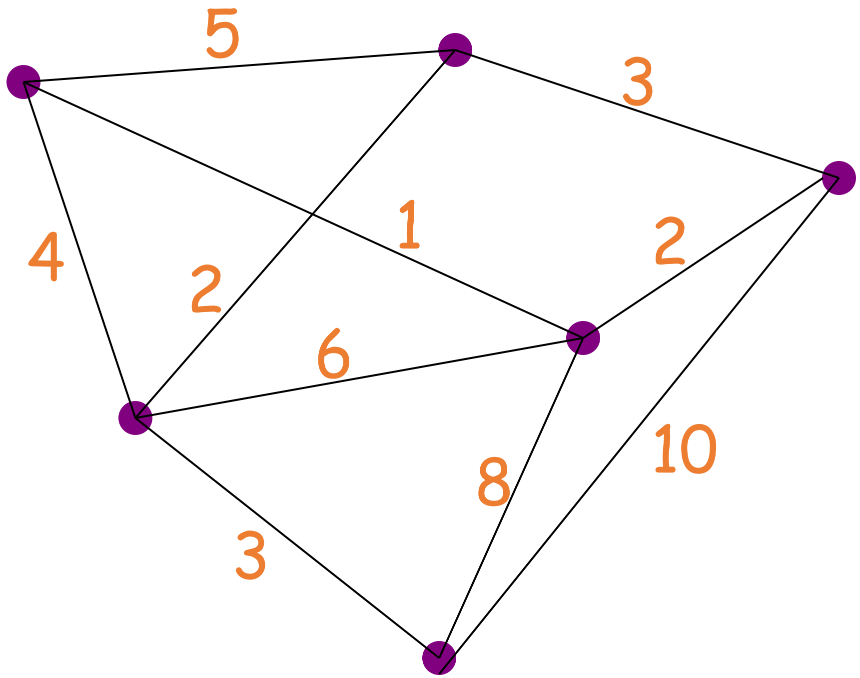

P
CYK-algorithm
$\{a^n b^n\}$
$\cdots$
$\{ww\}$

Exponential time algorithms: $DTIME(2^n)$

Represent intractable algorithms:

Some problem instances may take centuries to solve

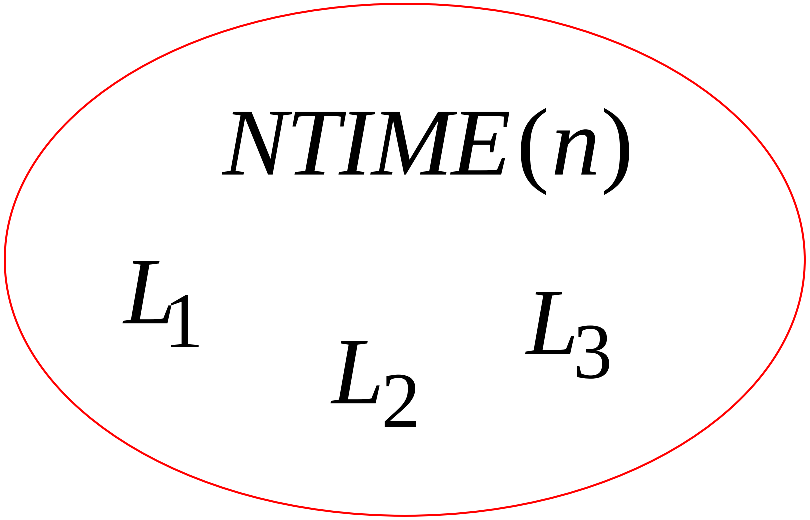
Example: the Traveling Salesperson Problem



Question: what is the shortest route that connects all cities?

Non-Determinism

Language class: $NTIME(n)$



A Non-Deterministic Machine accepts each string of length n in time $O(n)$ (testing solution is $O(n)$)

Example: $L = \{ww\}$

Non-Deterministic Algorithm
to accept a string ww :

- Use a two-tape Turing machine

- Guess the middle of the string
  and copy w on the second tape

- Compare the two tapes

$$L = \{ww\}$$

Time needed:

- Use a two-tape Turing machine

- Guess the middle of the string and copy w on the second tape $\quad O(|w|)$

- Compare the two tapes $\quad O(|w|)$

Total time: $\quad O(|w|)$

$$NTIME(n)$$

$$L=\{ww\}$$

In a similar way we define the class

$$NTIME(T(n))$$

for any time function: $T(n)$

Examples: $NTIME(n^2), NTIME(n^3), \ldots$

Non-Deterministic Polynomial time algorithms:

$$L \in NTIME(n^k)$$

The class NP

$$NP = \cup NTIME(n^k) \qquad \text{for all} \quad k$$

Non-Deterministic Polynomial time

Example: The Satisfiability Problem

Boolean expressions in
Conjunctive Normal Form:

$$t_1 \wedge t_2 \wedge t_3 \wedge \cdots \wedge t_k$$

$$t_i = x_1 \vee \bar{x}_2 \vee x_3 \vee \cdots \vee \bar{x}_p$$

Variables

Question: is expression satisfiable?
(if it can be made TRUE by assigning
appropriate logical values)

Example: $$(\overline{x}_1 \vee x_2) \wedge (x_1 \vee x_3)$$

Satisfiable: $$x_1 = 0, \;\; x_2 = 1, \;\; x_3 = 1$$

$$(\overline{x}_1 \vee x_2) \wedge (x_1 \vee x_3) = 1$$

Example: $(x_1 \vee x_2) \wedge \bar{x}_1 \wedge \bar{x}_2$

Not satisfiable

$$L = \{w : \text{expression } w \text{ is satisfiable}\}$$

For n variables: $L \in DTIME(2^n)$

exponential

Algorithm:

search exhaustively all the possible binary values of the variables

$$L = \{w : \text{expression } w \text{ is satisfiable}\}$$

$$L \in NP$$

Non-Deterministic Polynomial

The satisfiability problem is an NP - Problem

1) a non-deterministic machine guess about the solution,
2) a deterministic algorithm verifies or rejects the guess as a valid solution to the problem

$$L = \{w : \text{expression } w \text{ is satisfiable}\}$$

Time for n variables:

- Guess an assignment of the variables $O(n)$

- Check if this is a satisfying assignment $O(n)$

Total time: $O(n)$

- P So if problems are
    - They can be solved in polynomial time.

- NP
    - You can quickly (in polynomial time) test whether a solution is correct
        - without worrying about how hard it might be to find the solution).
    - They are still relatively easy: if only we could guess the right solution, we could then quickly test it.
        - e.g. RSA(key,text) and the "known plaintext attack"

Observation:

$$P \subseteq NP$$

Deterministic Polynomial

Non-Deterministic Polynomial

Open Problem: $P = NP\ ?$

Example: Does the Satisfiability problem have a polynomial time deterministic algorithm?

WE DO NOT KNOW THE ANSWER

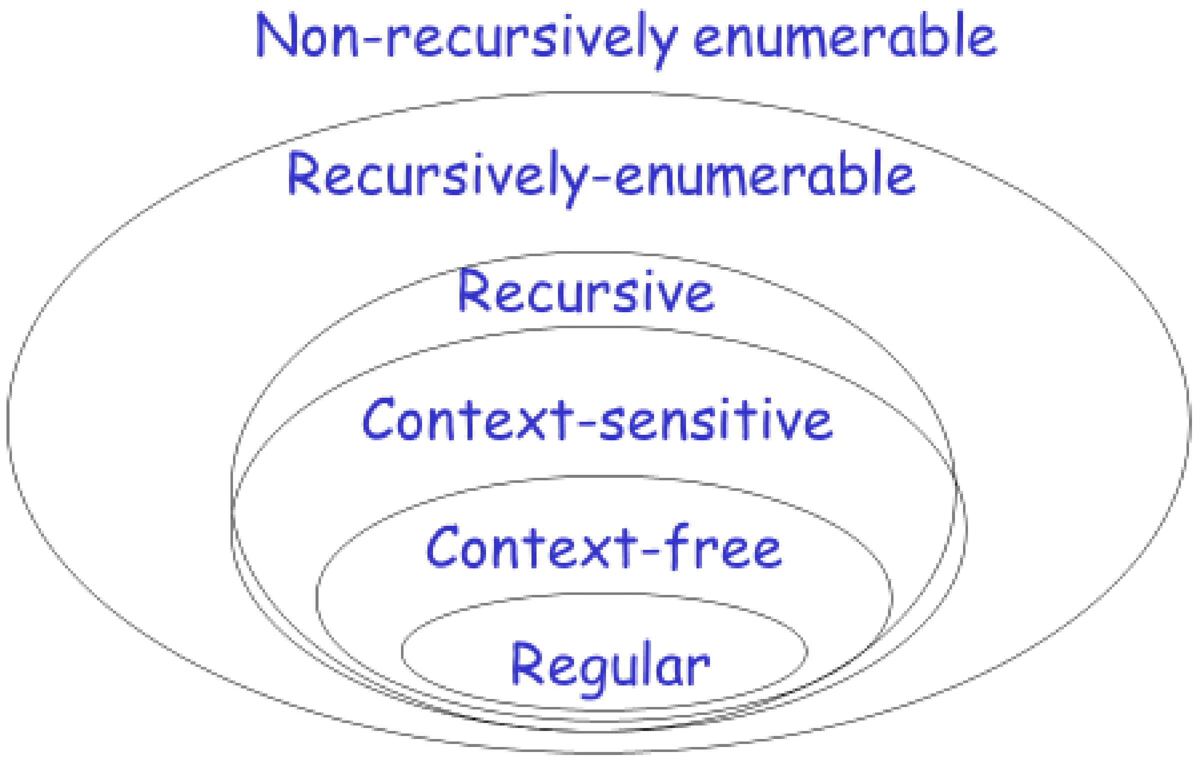
Outline

- Review of last week
- Computational Complexity
- Review

Exam Info

- Cheatsheet allowed
  - A4-size paper (both sides)
  - Your own work

The Chomsky Hierarchy



You will still be asked:

- N/DFA, RE, RG

Give the Chomsky Normal Form of G:

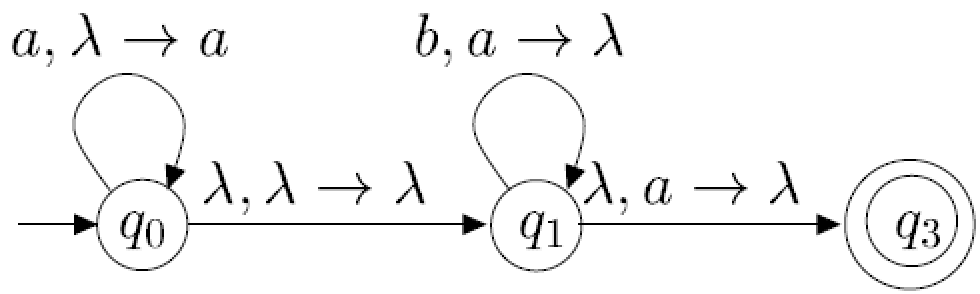
$$S \to AbBa$$
$$A \to ABa \mid a$$
$$B \to BaA \mid b$$

Use CYK algorithm to show

- “ababba” $\in$ L(G)

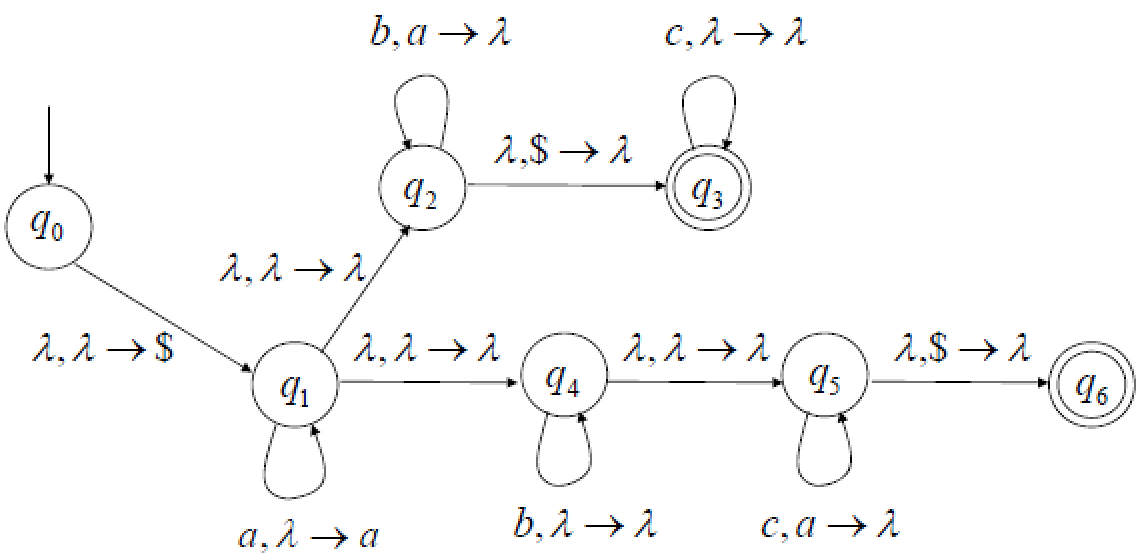
Give a NPDA for:

$$\{a^n b^k \quad : \; n > k \geq 0\}$$


$a, \lambda \to a$
$b, a \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, a \to \lambda$
q_0
q_1
q_3

What language is accepted by PDA?

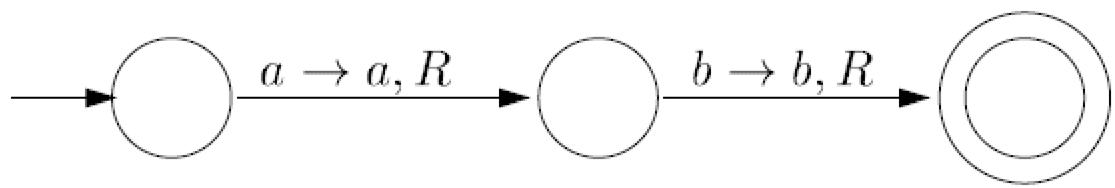
Test the strings aabbc and aabcc.



: $L = \{a^i\, b^j\, c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$

Give a Turing Machine for:

$$L = \{ab(a+b)^*\}$$


$a \to a, R$
$b \to b, R$

Provide the pseudocode for a Turing Machine computing:

$$f(x) = x^2$$

Something like:

- Copy x once: x(1)-x(2)-BLANK (- is a separator)
- For each 1 in x(1)
  - Replace it with 0
  - Append x(2) to BLANK
- Replace x(1),x(2) and separator between them with blanks

B	B	B	B	B	B	B	B	B	1	1	1	1	1	1	1	1	1																						

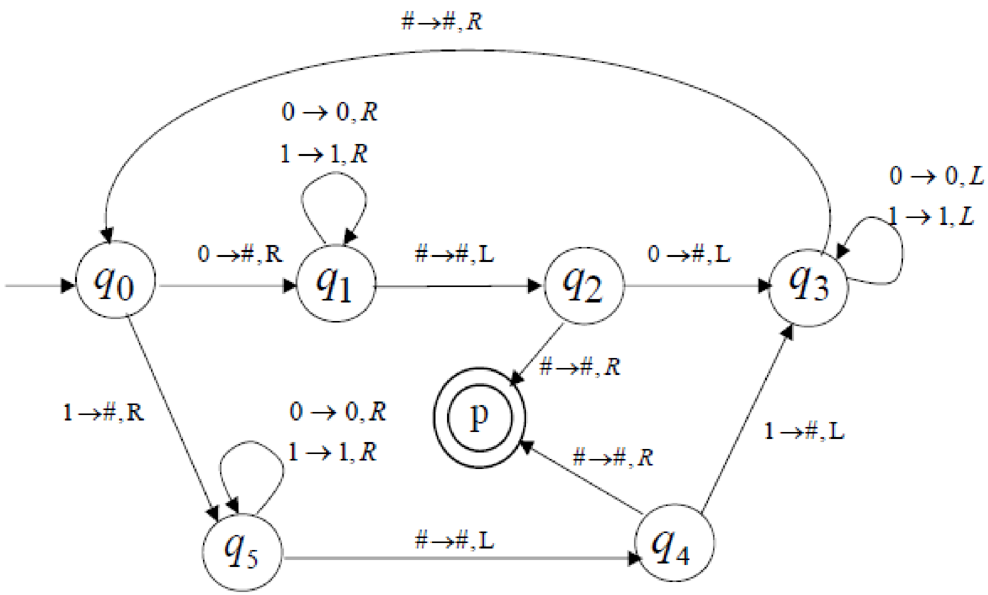
5) (5 points (BONUS)) Write the pseudocode for a three-tape Turing machine computing function:

$$f(x) = x \,\&\, y$$

where inputs x,y are binary numbers and "&" is "bitwise AND" operator.

- Copy x to TAPE1
- Copy y to TAPE2
- Traverse x and y simultaneously, for each position
  - If x or y has “0” then print “0” to TAPE3
  - If x and y has “1” then print “1” to TAPE3

What language is accepted by Turing Machine?



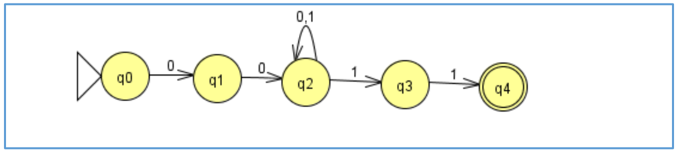
#	0	1	1	1	0	#																

- Odd-length palindromes generated using alphabet {0,1} (e.g. 010, 10101..etc)

1) (5+5+10=20) For the regular language $L = \{w \in \{0,1\}^* | w \text{ begins with } 00 \text{ and ends with } 11\}$

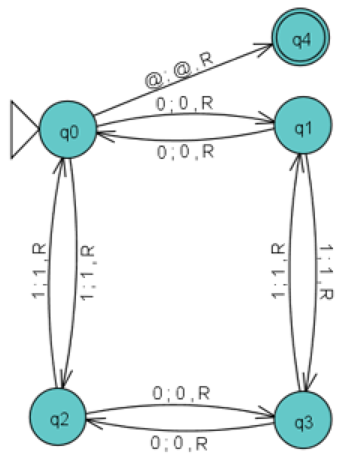
a) Generate a regular expression.
b) Convert the regular expression to a finite automata.
c) Convert finite automata to a regular grammar.

00(1+0)*11



S	$\longrightarrow$	0A
B	$\longrightarrow$	0B
B	$\longrightarrow$	1B
D	$\longrightarrow$	λ
C	$\longrightarrow$	1D
A	$\longrightarrow$	0B
B	$\longrightarrow$	1C

5) (10+10=20) Given following Turing Machine (@ is used to denote blank tape locations):



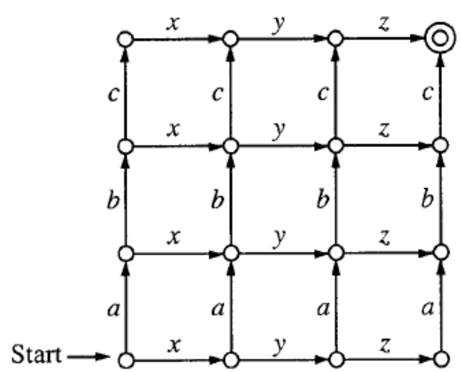
a) Show the steps (both head and tape) to derive “011011”.
b) Describe the language accepted by this machine.

@**q0**011011@
@0**q1**11011@
@01**q3**1011@
@011**q1**011@
@0110**q0**11@
@01101**q1**1@
@011011**q0**@

@011011@**q4**

The machine accepts strings with even number of 0’s and 1’s.

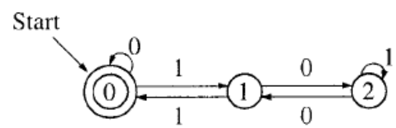
25.



The finite automaton above recognizes a set of strings of length 6. What is the total number of strings in the set?

(A) 18 (B) 20 (C) 30 (D) 32
(E) None of the above

28.



State 0 is both the starting state and the accepting state.

Each of the following is a regular expression that denotes a subset of the language recognized by the automaton above EXCEPT

(A) 0*(11)*0*

(B) 0*1(10*1)*1

(C) 0*1(10*1)*10*

(D) 0*1(10*1)0(100)*

(E) (0*1(10*1)*10* + 0*)*

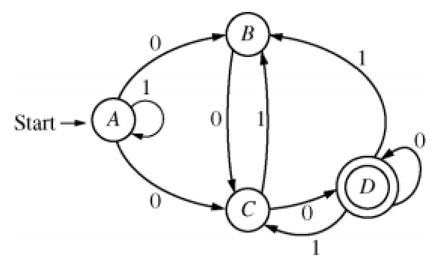
70. If DFA denotes "deterministic finite automata" and NDFA denotes "nondeterministic finite automata," which of the following is FALSE?

(A) For any language L, if L can be recognized by a DFA, then $\overline{L}$ can be recognized by a DFA.
(B) For any language L, if L can be recognized by an NDFA, then $\overline{L}$ can be recognized by an NDFA.
(C) For any language L, if L is context-free, then $\overline{L}$ is context-free.
(D) For any language L, if L can be recognized in polynomial time, then $\overline{L}$ can be recognized in polynomial time.
(E) For any language L, if L is decidable, then $\overline{L}$ is decidable.

5. Which of the following regular expressions will not generate a string with two consecutive 1s? (Note that ε denotes the empty string.)

I. $(1 + \varepsilon)(01 + 0)^*$
II. $(01 + 10)^*$
III. $(0 + 1)^*(0 + \varepsilon)$

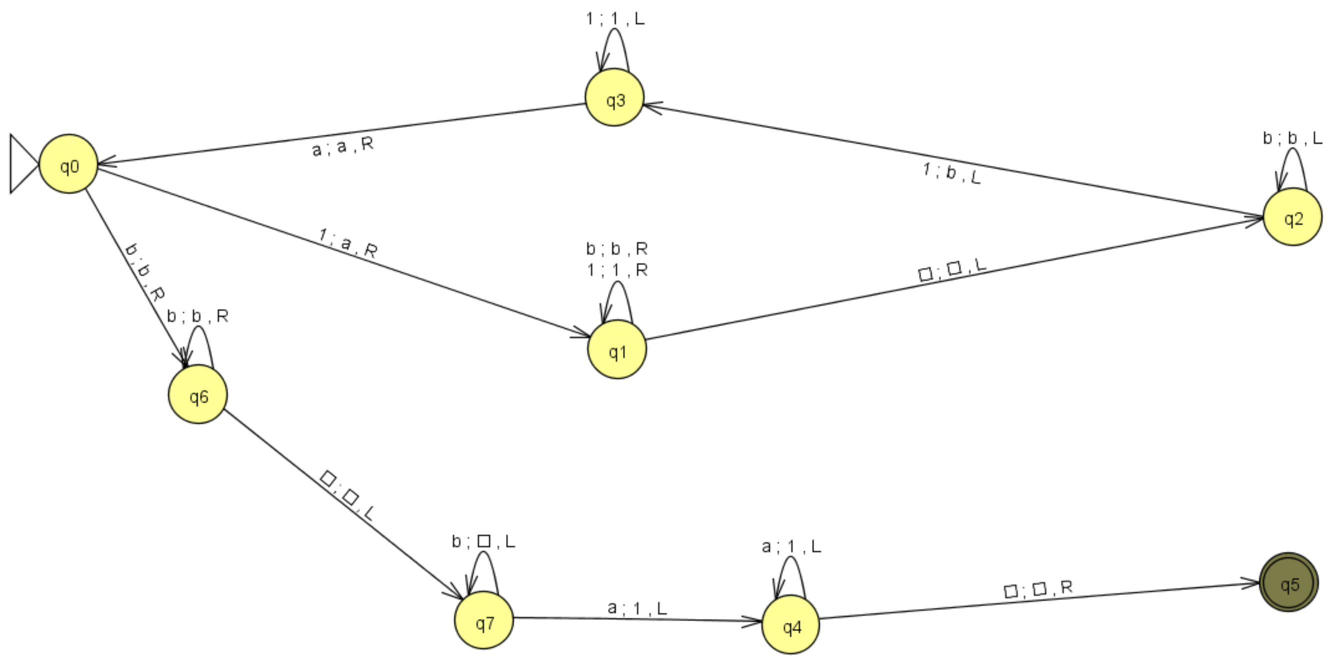
(A) I only
(B) II only
(C) III only
(D) I and II only
(E) II and III only

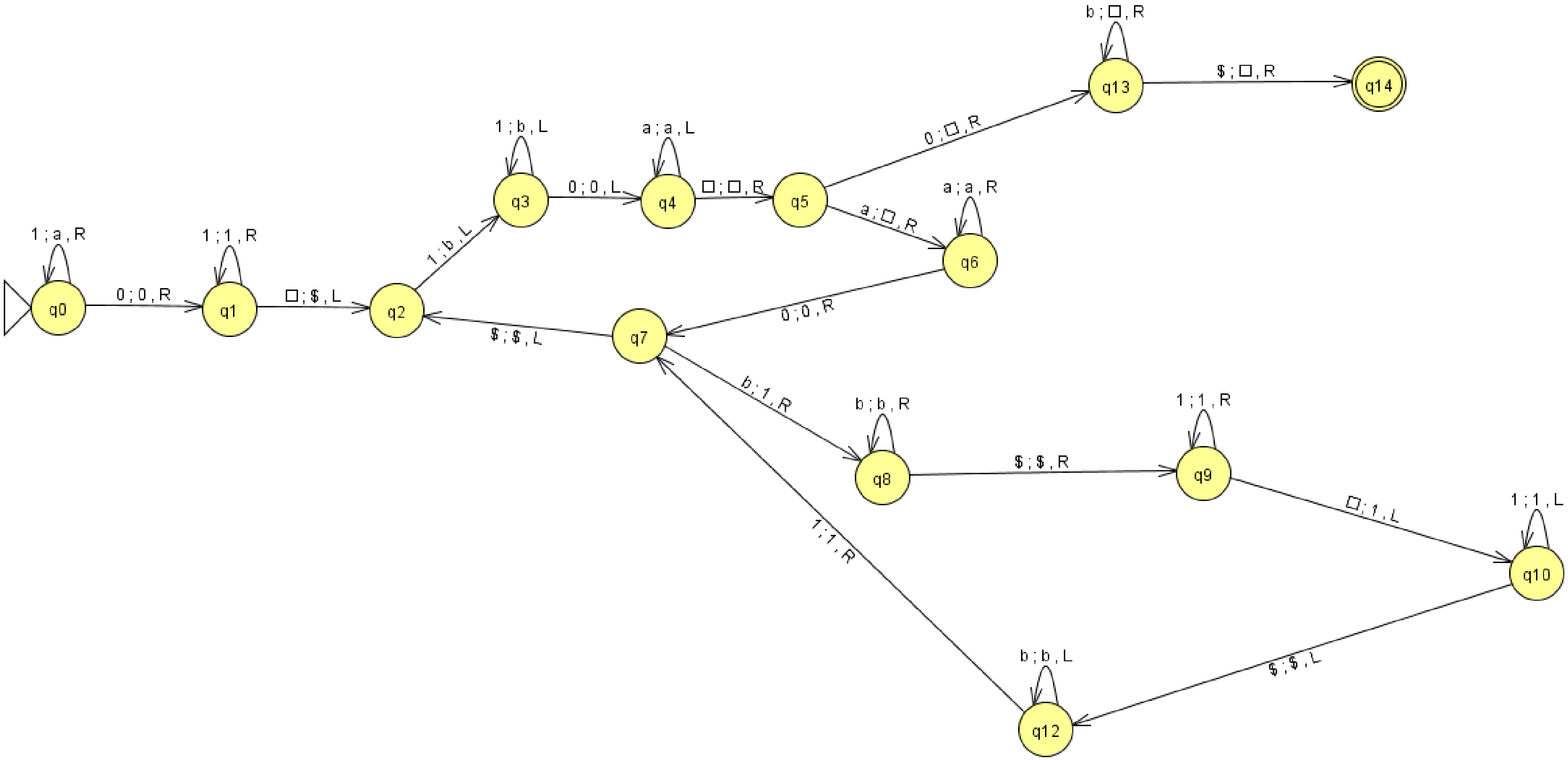


14. The figure above represents a nondeterministic finite automaton with accepting state D. Which of the following strings does the automaton accept?

(A) 001
(B) 1101
(C) 01100
(D) 000110
(E) 100100

- Turing machine to compute
  - f(x)=x/2 where x is unary and |x| mod 2 = 0
  - f(x)=x*y where x and y are unary e.g. 110111 is 2*3


1 ; 1 , L
q3
a ; a , R
1 ; b , L
b ; b , L
q0
q2
1 ; a , R
b ; b , R
1 ; 1 , R
□ ; □ , L
b ; b , R
b ; b , R
q1
q6
□ ; □ , L
b ; □ , L
a ; 1 , L
q7
a ; 1 , L
q4
□ ; □ , R
q5



1 ; a , R
q0
0 ; 0 , R
1 ; 1 , R
q1
□ ; $, L
q2
1 ; b , L
1 ; b , L
q3
0 ; 0 , L
a ; a , L
q4
□ ; □ , R
q5
0 ; □ , R
b ; □ , R
q13
$; □ , R
q14
a ; □ , R
a ; a , R
q6
0 ; 0 , R
q7
$; $, L
b ; 1 , R
b ; b , R
q8
$; $, R
1 ; 1 , R
q9
□ ; 1 , L
1 ; 1 , L
q10
$; $, L
b ; b , L
q12
1 ; 1 , R