CS116-Automata Theory and Formal Languages

Lecture 1
Introduction

Computer Science Department

1st Semester 2025-2026

Three Central Areas of the Theory of Computation

What are the fundamental capabilities and limitations of computers?

- Complexity Theory: what makes some problems computationally hard and others easy? how much time/space is needed (P, NP, NP-complete, etc.)
- Computability Theory: what can/can't be solved by any algorithm (Decidable vs. Undecidable). Halting problem, problem of determining whether a mathematical statement is true or false
- Automata Theory: machine models (FA, PDA, TM) and the languages they recognize.

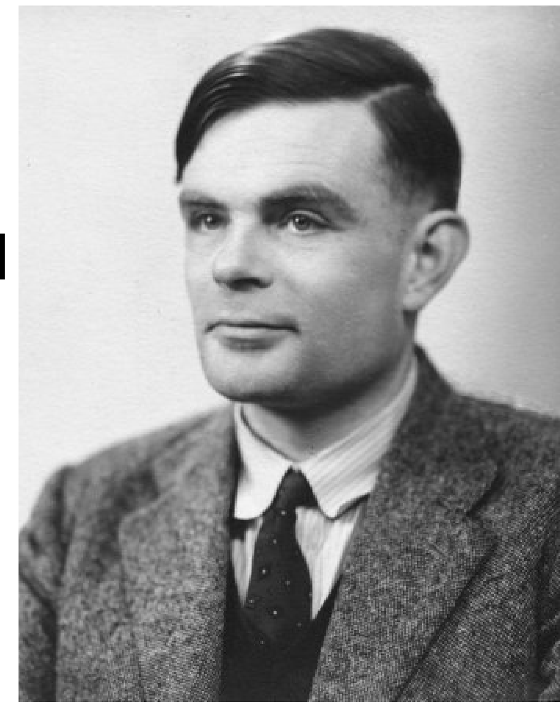
Theory of Computation: A Historical Perspective

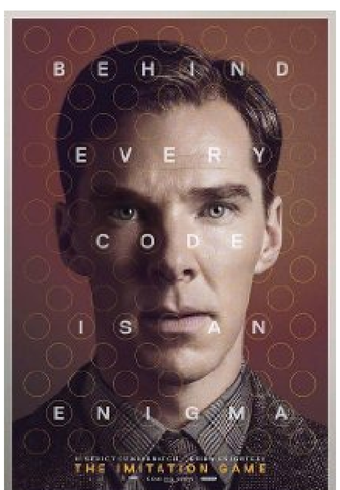
1930s	• Alan Turing studies Turing machines • Decidability • Halting problem
1940-1950s	• "Finite automata" machines studied • Noam Chomsky proposes the "Chomsky Hierarchy" for formal languages
1969	Cook introduces "intractable" problems or "NP-Hard" problems
1970-	Modern computer science: compilers, computational & complexity theory evolve

(A pioneer of automata theory)

Alan Turing (1912-1954)

- Father of Modern Computer Science
- English mathematician
- Studied abstract machines called ***Turing machines*** even before computers existed
- Heard of the Turing test?



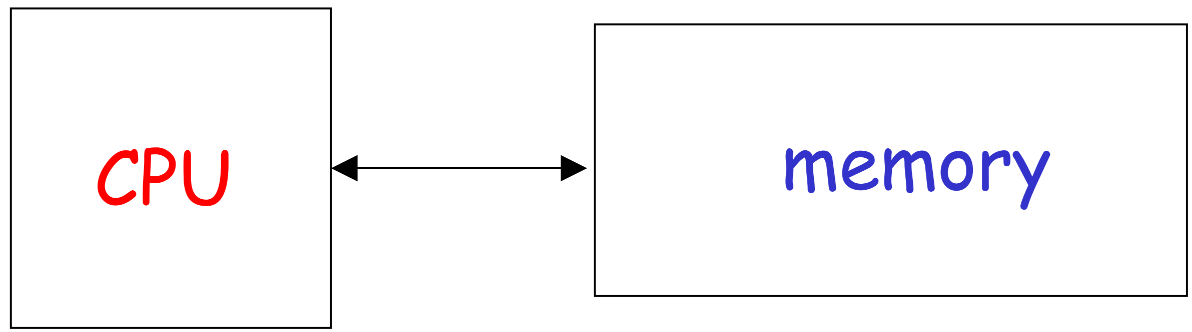


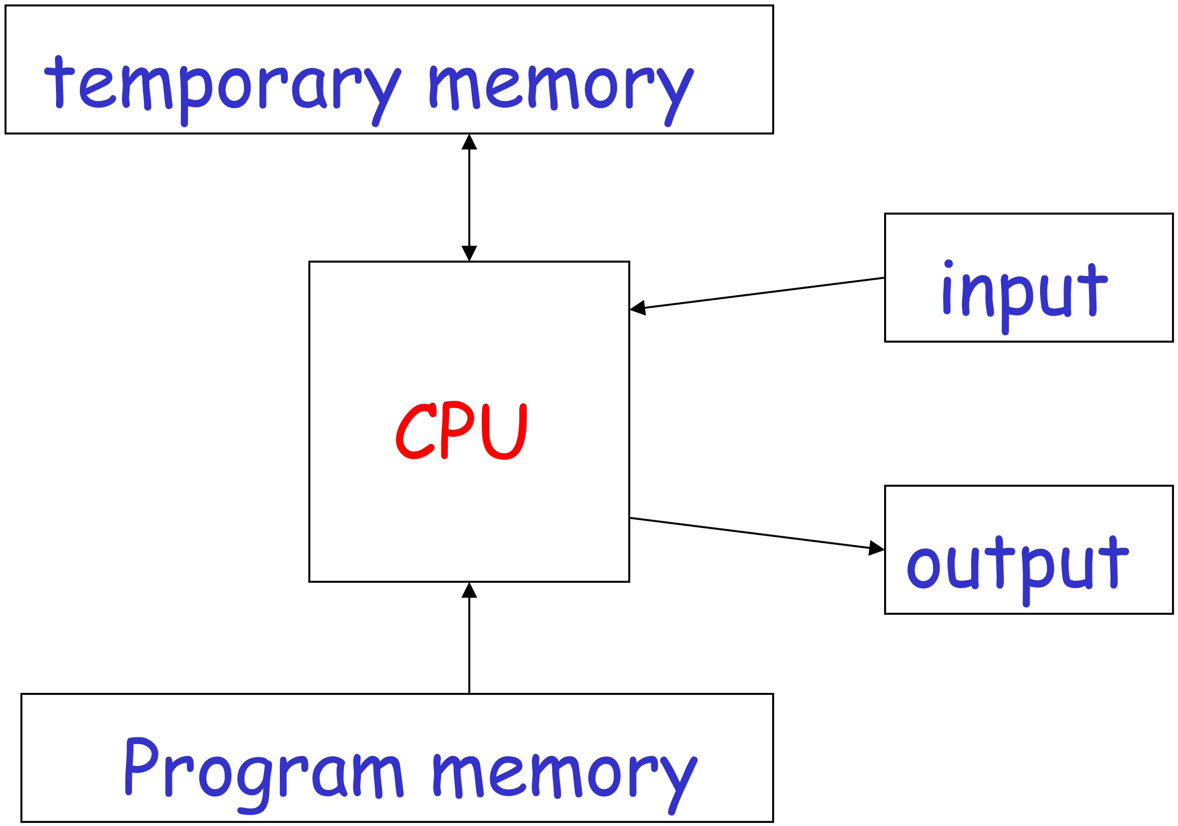
What is Automata Theory?

- *Study of abstract computing devices, or "machines"*
- Automaton = an abstract computing device
  - Note: A "device" need not even be a physical hardware!
- A fundamental question in computer science:
  - Find out what different models of machines can do and cannot do
  - The *theory of computation*
  - *Computation = language recognition*

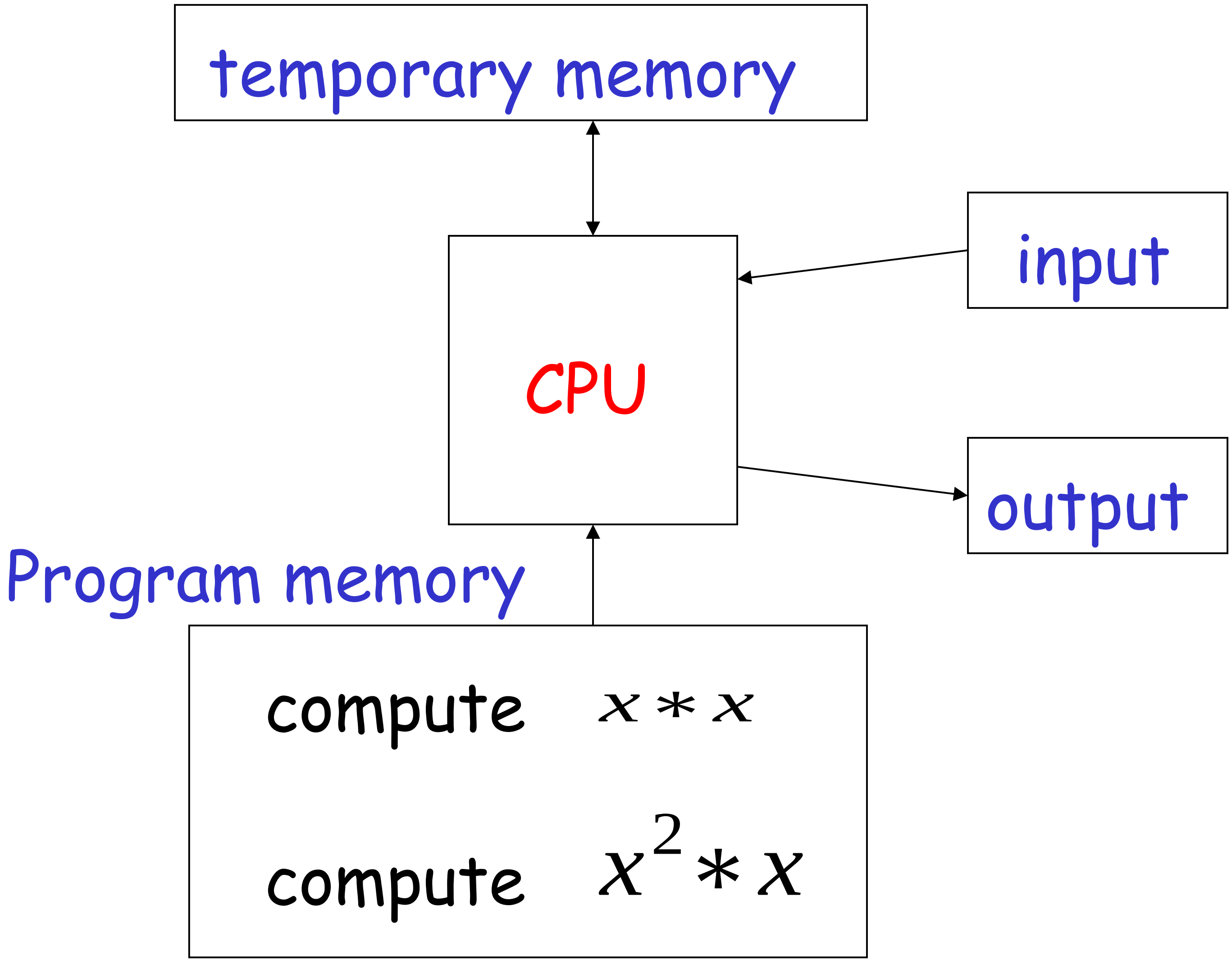
Outline of the course contents

Computation

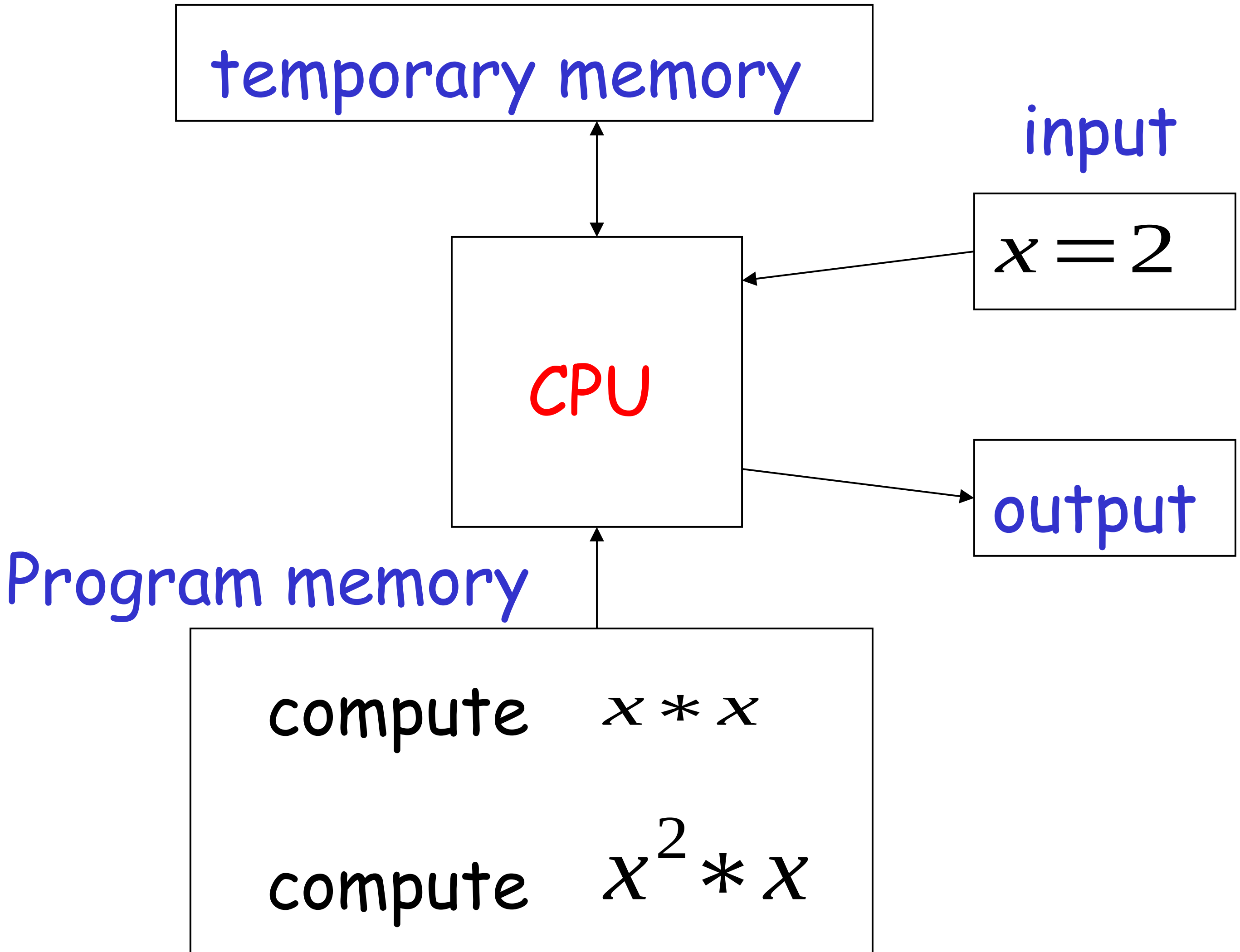



temporary memory
input
CPU
output
Program memory

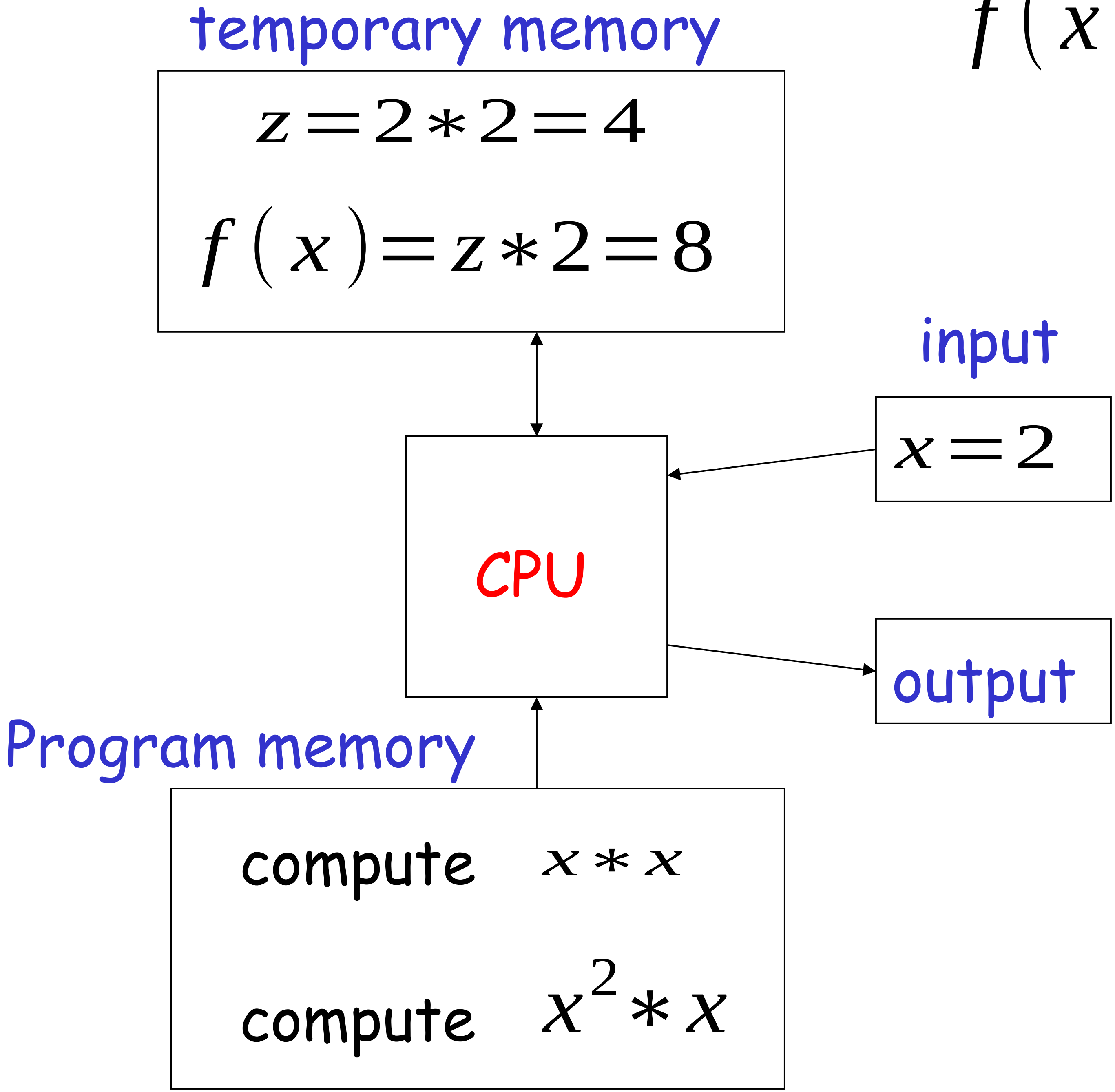
Example: $f(x) = x^3$



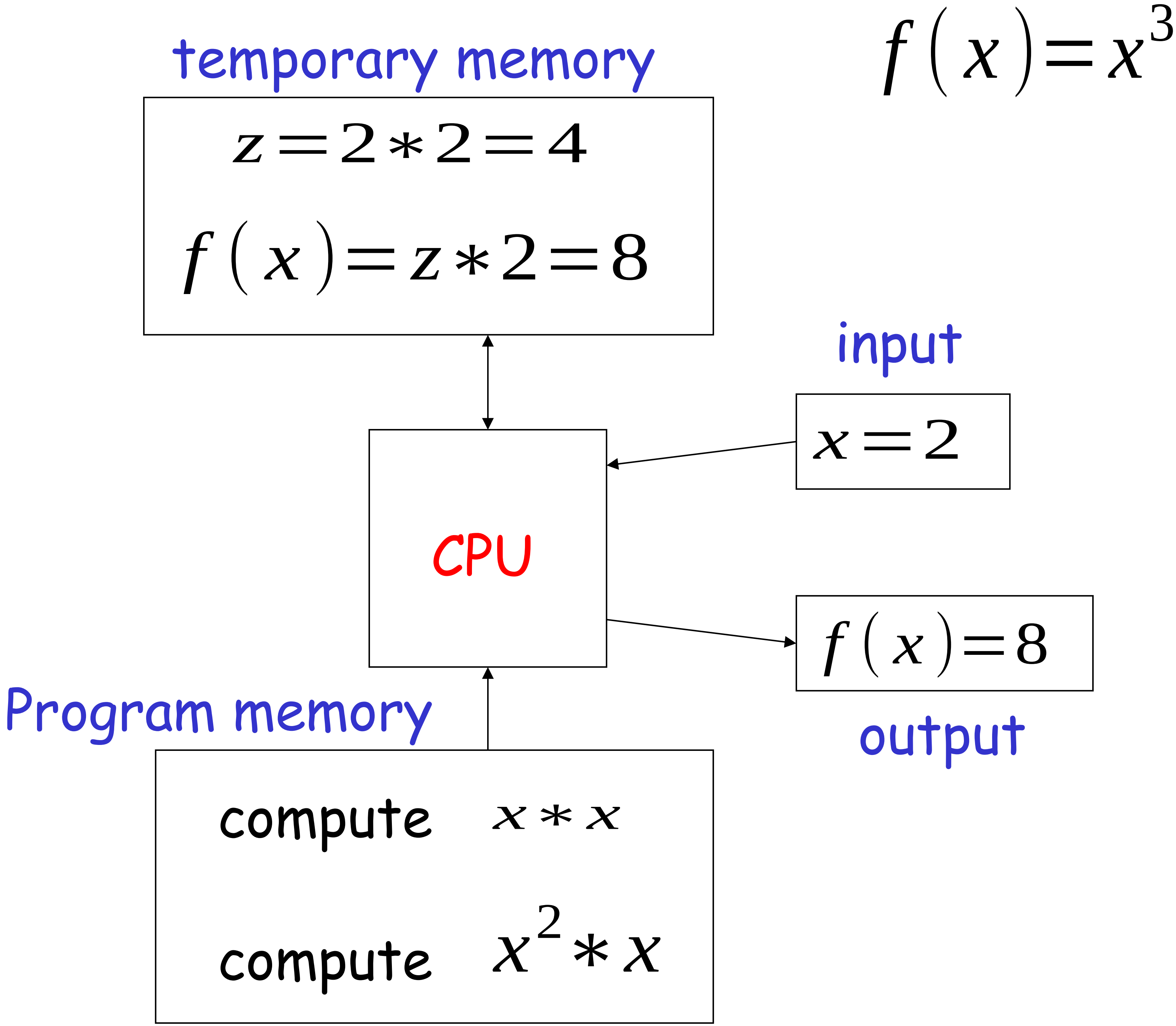
$$f(x) = x^3$$


temporary memory
input
$x = 2$
CPU
output
Program memory
compute $x * x$
compute $x^2 * x$

$$f(x)=x^3$$

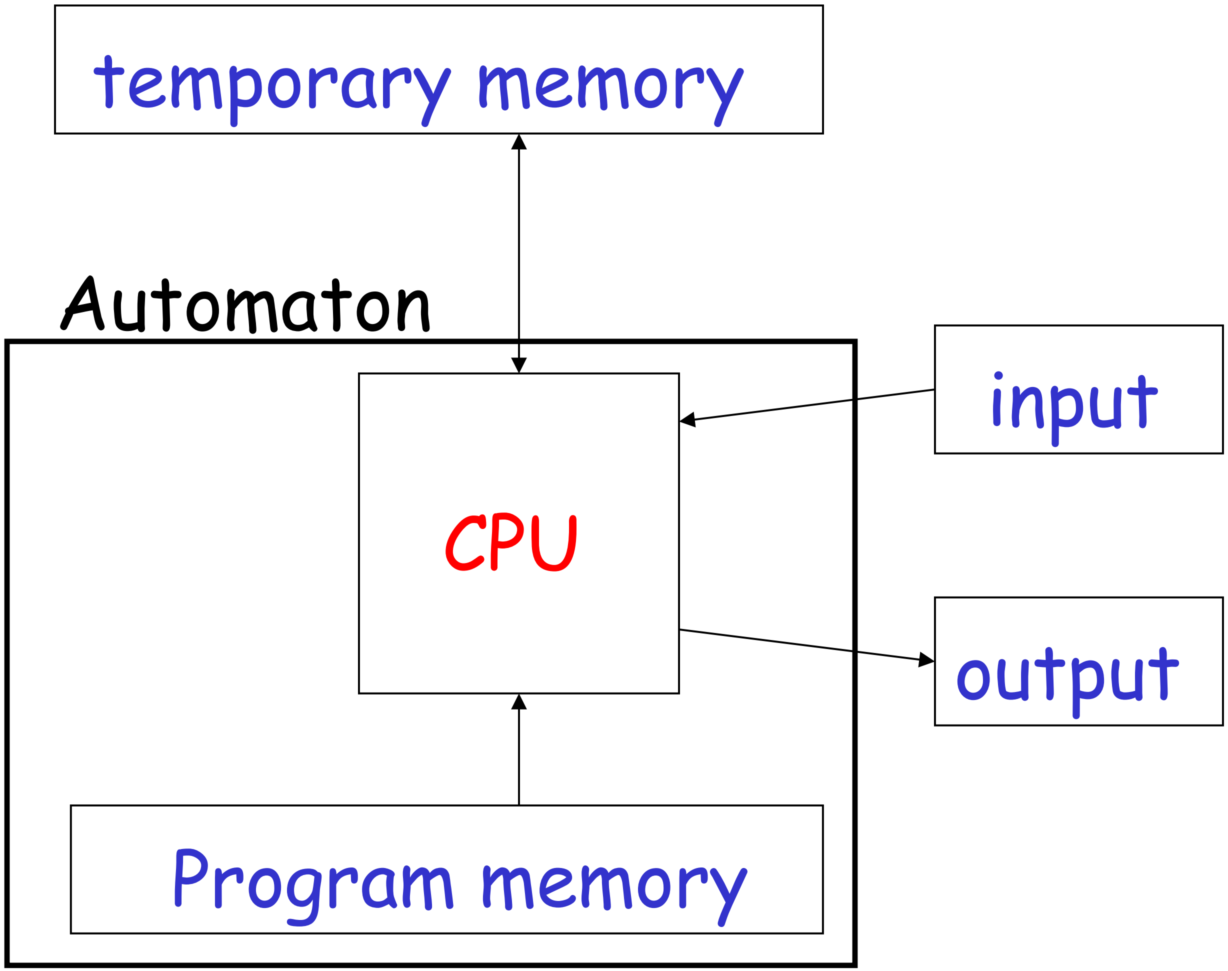


temporary memory
$z=2*2=4$
$f(x)=z*2=8$
input
$x=2$
CPU
output
Program memory
compute $x*x$
compute x^2*x

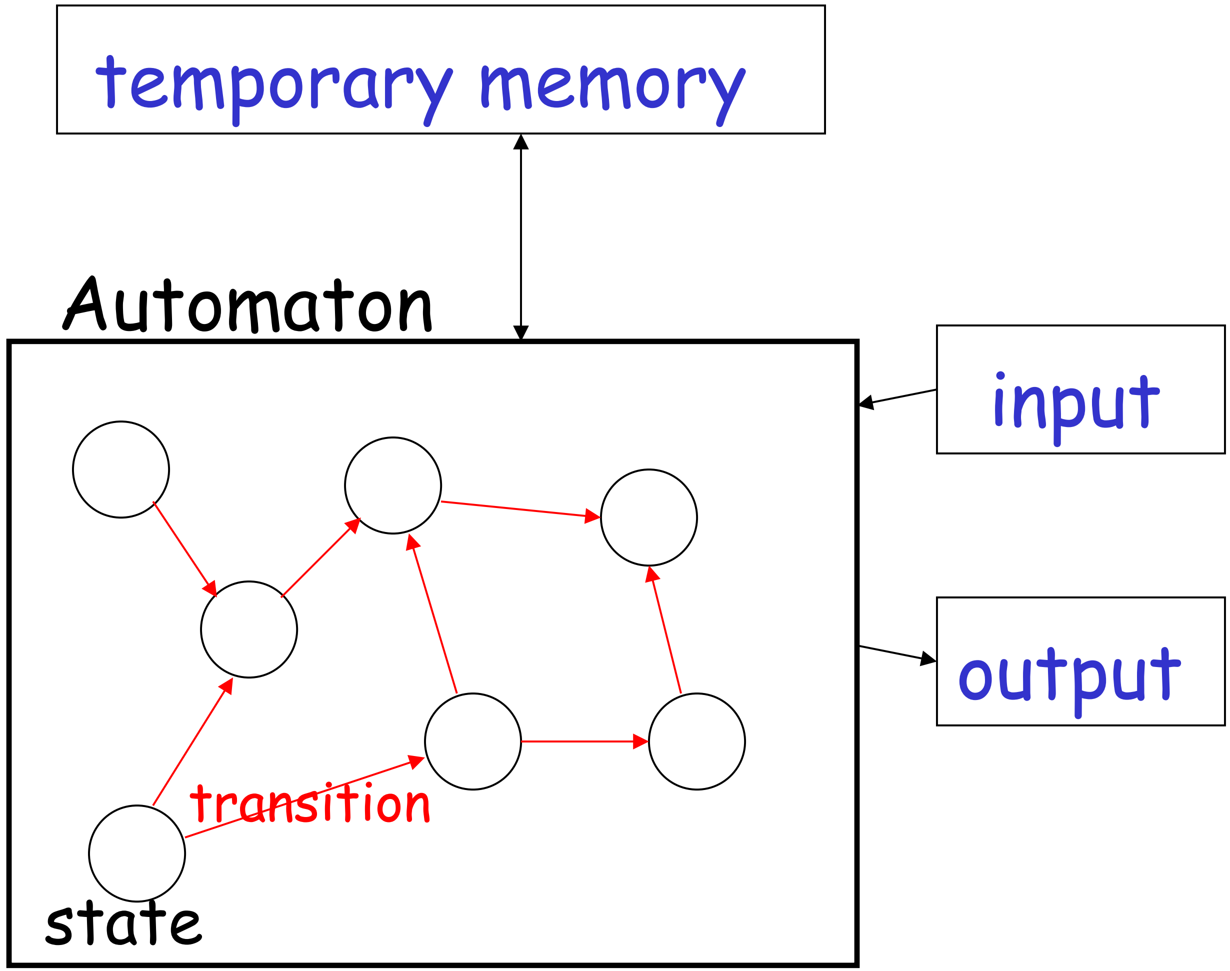


$f(x)=x^3$
temporary memory
$z=2*2=4$
$f(x)=z*2=8$
input
$x=2$
CPU
$f(x)=8$
output
Program memory
compute $x*x$
compute x^2*x

Automaton


temporary memory
Automaton
CPU
input
output
Program memory

Automaton



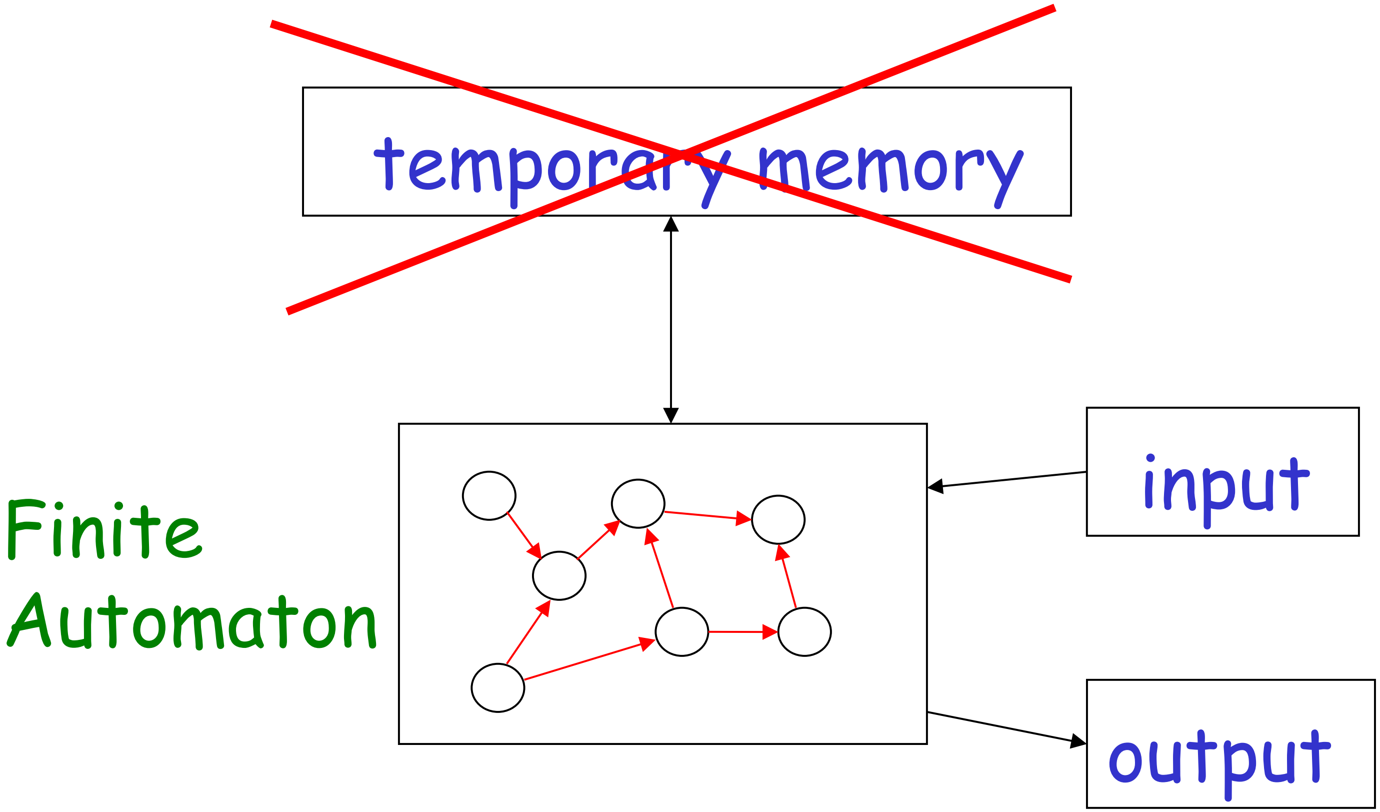
temporary memory
Automaton
input
output
transition
state

Different Kinds of Automata

Automata are distinguished by the temporary memory

- **Finite Automata:** no temporary memory
- **Pushdown Automata:** stack
- **Turing Machines:** random access memory

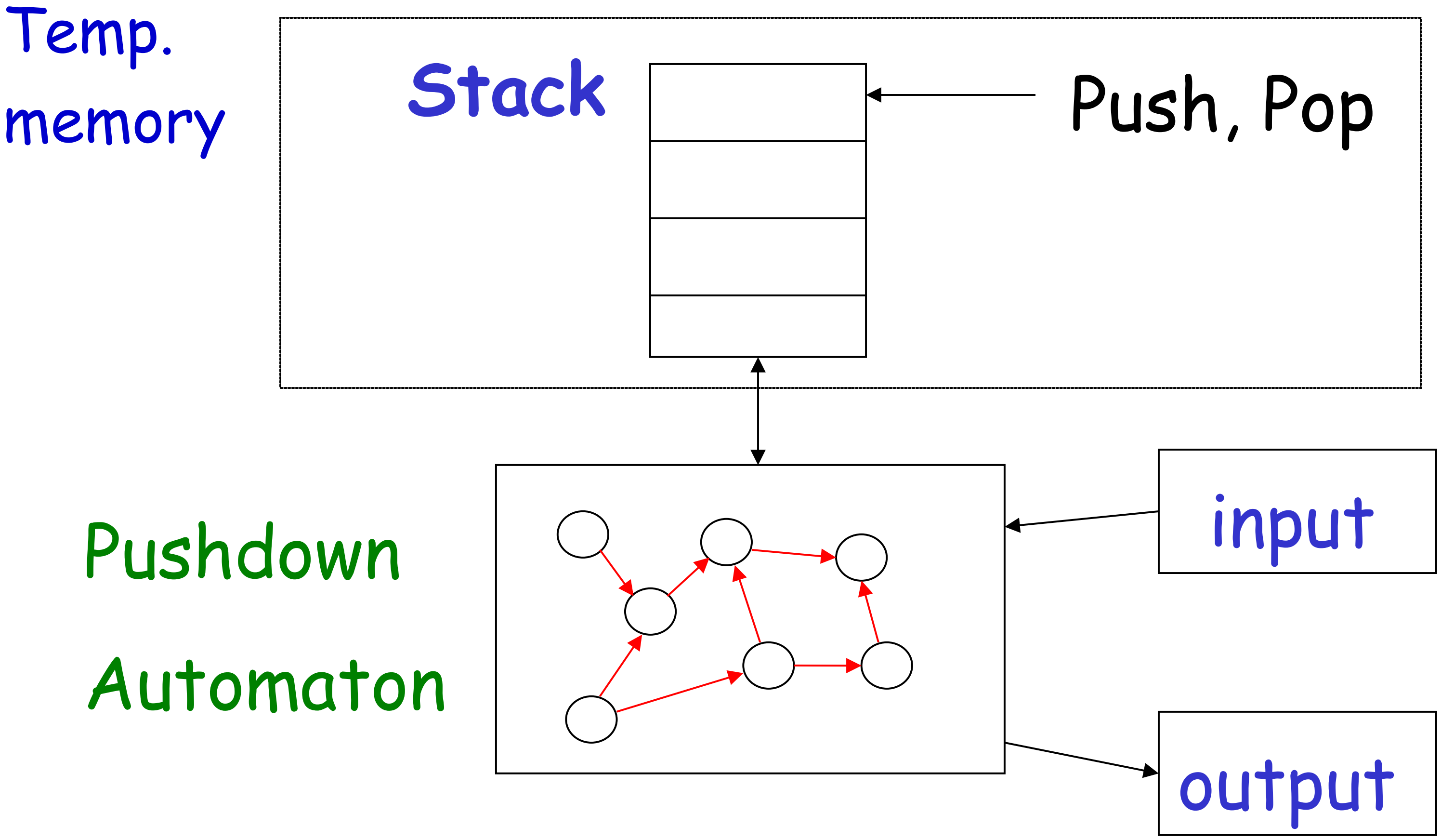
Finite Automaton



Example: Elevators, Vending Machines

(small computing power)

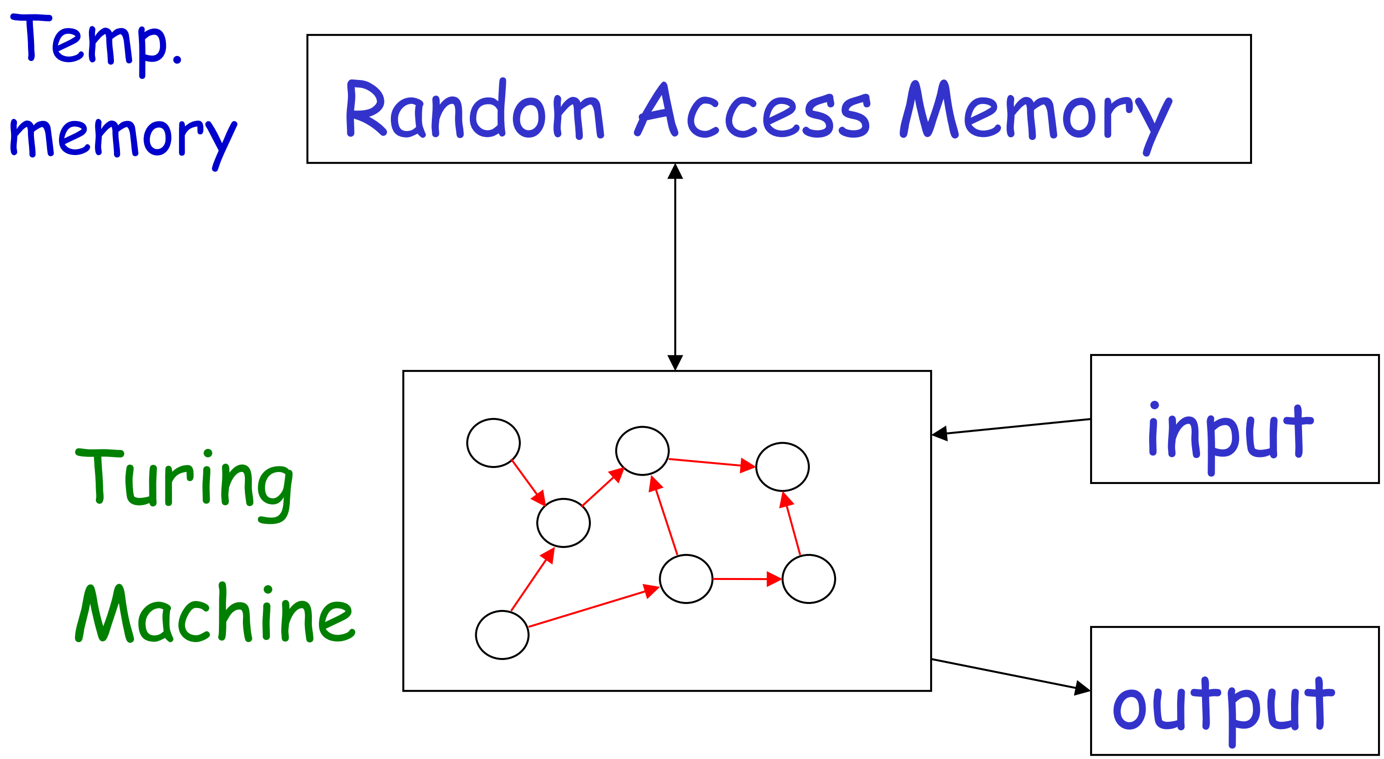
Pushdown Automaton



Example: Compilers for Programming Languages

(medium computing power)

Turing Machine



Examples: Any Algorithm

(highest computing power)

Power of Automata

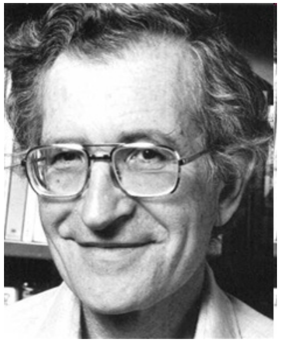
Simple problems | More complex problems | Hardest problems

Finite Automata < Pushdown Automata < Turing Machine

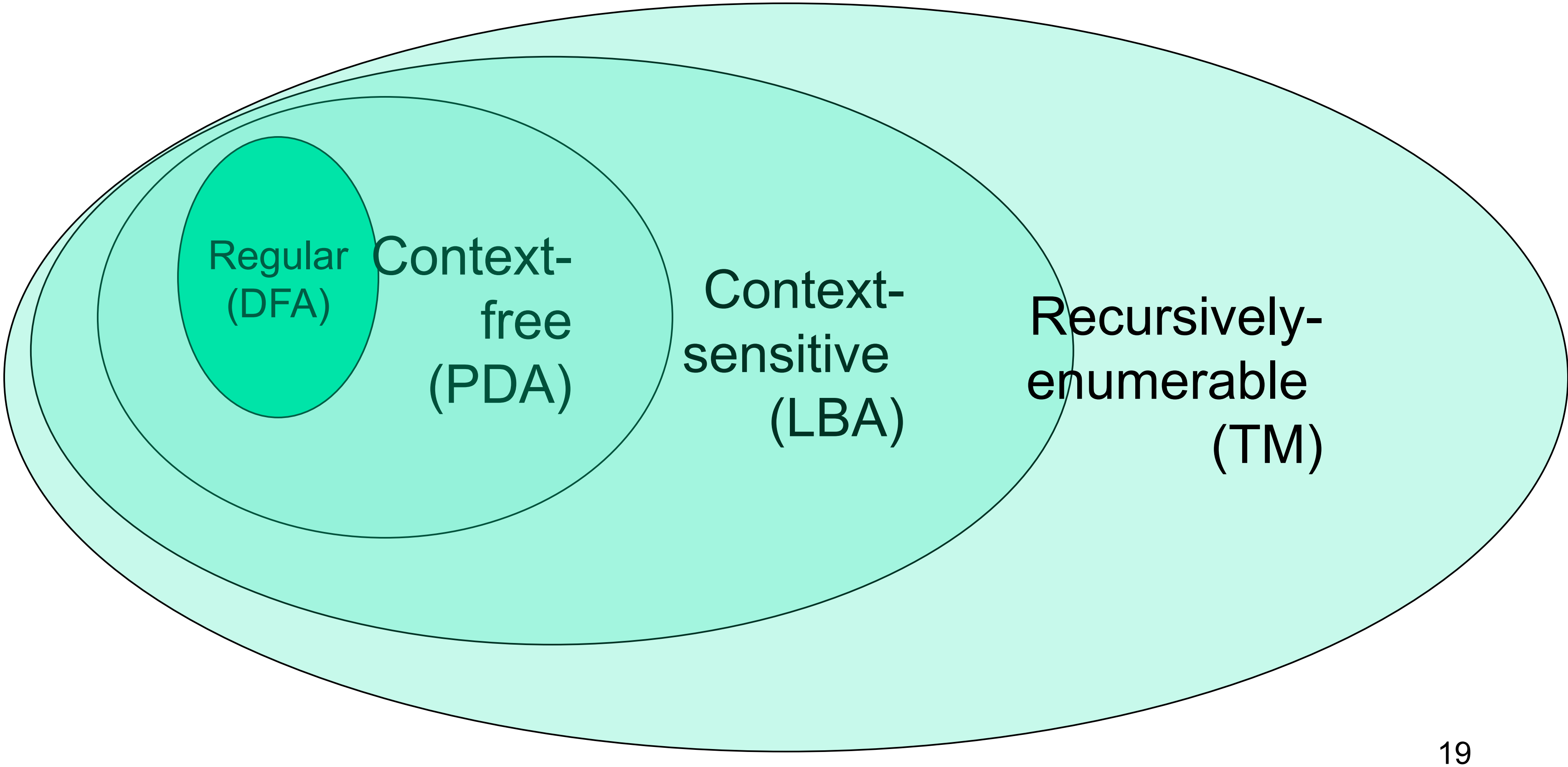
Less power → More power

Solve more computational problems

The Chomsky Hierarchy



- A containment hierarchy of classes of formal languages



Turing Machine is the most powerful computational model known

Question: Are there computational problems that a Turing Machine cannot solve?

Answer: Yes (unsolvable problems)

Time Complexity of Computational Problems:

NP-complete problems

Believed to take exponential time to be solved

P problems

Solved in polynomial time

Languages

Language: a set of strings

String: a sequence of symbols from some alphabet

Example:

Strings: cat, dog, house

Language: {cat, dog, house}

Alphabet: $\Sigma = \{a, b, c, \ldots, z\}$

Languages are used to describe computation problems:

$$PRIMES = \{2,3,5,7,11,13,17,\ldots\}$$

$$EVEN = \{0,2,4,6,\ldots\}$$

Alphabet: $\Sigma = \{0,1,2,\ldots,9\}$

Alphabets and Strings

An alphabet is a set of symbols

Example Alphabet: $\Sigma = \{a, b\}$

A string is a sequence of symbols from the alphabet

Example Strings

a

ab

$abba$

$aaabbbaabab$

$u = ab$

$v = bbbaaa$

$w = abba$

Decimal numbers alphabet $\Sigma = \{0,1,2,\ldots,9\}$

102345 567463386

Binary numbers alphabet $\Sigma = \{0,1\}$

100010001 101101111

Unary numbers alphabet $\Sigma = \{1\}$

Unary number:	1	11	111	1111	11111
Decimal number:	1	2	3	4	5

String Operations

$$w = a_1 a_2 \cdots a_n \qquad abba$$

$$v = b_1 b_2 \cdots b_m \qquad bbbaaa$$

Concatenation

$$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m \qquad abbabbbaaa$$

$$w = a_1 a_2 \cdots a_n \qquad ababaaabbb$$

Reverse

$$w^R = a_n \cdots a_2 a_1 \qquad bbbaaababa$$

String Length

$$w = a_1 a_2 \cdots a_n$$

Length: $|w| = n$

Examples:

$$|abba| = 4$$

$$|aa| = 2$$

$$|a| = 1$$

Length of Concatenation

$$|uv| = |u| + |v|$$

Example: $u = aab, \quad |u| = 3$

$$v = abaab, \quad |v| = 5$$

$$|uv| = |aababaab| = 8$$

$$|uv| = |u| + |v| = 3 + 5 = 8$$

Empty String

A string with no letters is denoted: λ **or** ε

Observations: $|\lambda| = 0$

$$\lambda w = w\lambda = w$$

$$\lambda abba = abba\lambda = ab\lambda ba = abba$$

Substring

Substring of string:

a subsequence of consecutive characters

String	Substring
$\underline{ab}bab$	ab
$\underline{abba}b$	$abba$
$ab\underline{b}ab$	b
$a\underline{bbab}$	$bbab$

Prefix and Suffix

$$abbab$$

Prefixes	Suffixes
λ	$abbab$
a	$bbab$
ab	bab
abb	ab
$abba$	b
$abbab$	λ

$$w = uv$$

prefix

suffix

Another Operation

$$w^n = \underbrace{ww\cdots w}_{n}$$

Example: $(abba)^2 = abbaabba$

Definition: $w^0 = \lambda$

$$(abba)^0 = \lambda$$

The * Operation

Σ^* : the set of all possible strings from alphabet Σ

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

The + Operation

Σ^{+} : the set of all possible strings from alphabet Σ except λ

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

$$\Sigma^{+} = \Sigma^* - \lambda$$

$$\Sigma^{+} = \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

Languages

A language over alphabet Σ

is any subset of Σ^*

Examples:

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$$

Language: $\{\lambda\}$

Language: $\{a, aa, aab\}$

Language: $\{\lambda, abba, baba, aa, ab, aaaaaa\}$

More Language Examples

Alphabet $\Sigma = \{a, b\}$

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

$$\left.\begin{array}{l} \lambda \\ ab \\ aabb \\ aaaaabbbbb \end{array}\right\} \in L$$

$$abb \notin L$$

Prime numbers

Alphabet $\Sigma = \{0,1,2,\ldots,9\}$

Language:

$$PRIMES = \{x : x \in \Sigma^* \text{ and } x \text{ is prime}\}$$

$$PRIMES = \{2,3,5,7,11,13,17,\ldots\}$$

Even and odd numbers

Alphabet $\Sigma = \{0,1,2,\ldots,9\}$

$$EVEN = \{x : x \in \Sigma^* \text{ and } x \text{ is even}\}$$

$$EVEN = \{0,2,4,6,\ldots\}$$

$$ODD = \{x : x \in \Sigma^* \text{ and } x \text{ is odd}\}$$

$$ODD = \{1,3,5,7,\ldots\}$$

Unary Addition

Alphabet: $\Sigma = \{1, +, =\}$

Language:

$$ADDITION = \{x + y = z \ : x = 1^n, y = 1^m, z = 1^k, \\ n + m = k\}$$

$$11 + 111 = 11111 \in ADDITION$$

$$111 + 111 = 111 \notin ADDITION$$

Squares

Alphabet: $\Sigma = \{1, \#\}$

Language:

$$SQUARES = \{x\#y \ : x = 1^n, y = 1^m, m = n^2\}$$

$$11\#1111 \in SQUARES$$

$$111\#1111 \notin SQUARES$$

Note that:

Sets $\emptyset = \{\ \} \neq \{\lambda\}$

Set size $|\{\ \}| = |\emptyset| = 0$

Set size $|\{\lambda\}| = 1$

String length $|\lambda| = 0$

Operations on Languages

The usual set operations

$$\{a, ab, aaaa\} \cup \{bb, ab\} = \{a, ab, bb, aaaa\}$$

$$\{a, ab, aaaa\} \cap \{bb, ab\} = \{ab\}$$

$$\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$$

Complement: $\overline{L} = \Sigma^* - L$

$$\overline{\{a, ba\}} = \{\lambda, b, aa, ab, bb, aaa, \ldots\}$$

Reverse

Definition: $L^R = \{w^R : w \in L\}$

Examples: $\{ab, aab, baba\}^R = \{ba, baa, abab\}$

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

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

Concatenation

Definition: $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

Example: $\{a, ab, ba\}\{b, aa\}$

$= \{ab, aaa, abb, abaa, bab, baaa\}$

Another Operation

Definition: $$L^n = \underbrace{LL \cdots L}_{n}$$

$$\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} =$$
$$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

Special case: $L^0 = \{\lambda\}$

$$\{a, bba, aaa\}^0 = \{\lambda\}$$

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

$$L^2 = \{ a^n b^n a^m b^m : n, m \geq 0 \}$$

$$aabbaaabbb \in L^2$$

Star-Closure (Kleene *)

All strings that can be constructed from L

Definition: $L^* = L^0 \cup L^1 \cup L^2 \cdots$

Example:

$$\{a, bb\}^* = \begin{Bmatrix} \lambda, \\ a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{Bmatrix}$$

Positive Closure

Definition: $L^+ = L^1 \cup L^2 \cup \cdots$

Same with L^* but without the λ

$$\{a, bb\}^+ = \left\{\begin{array}{l} a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array}\right\}$$