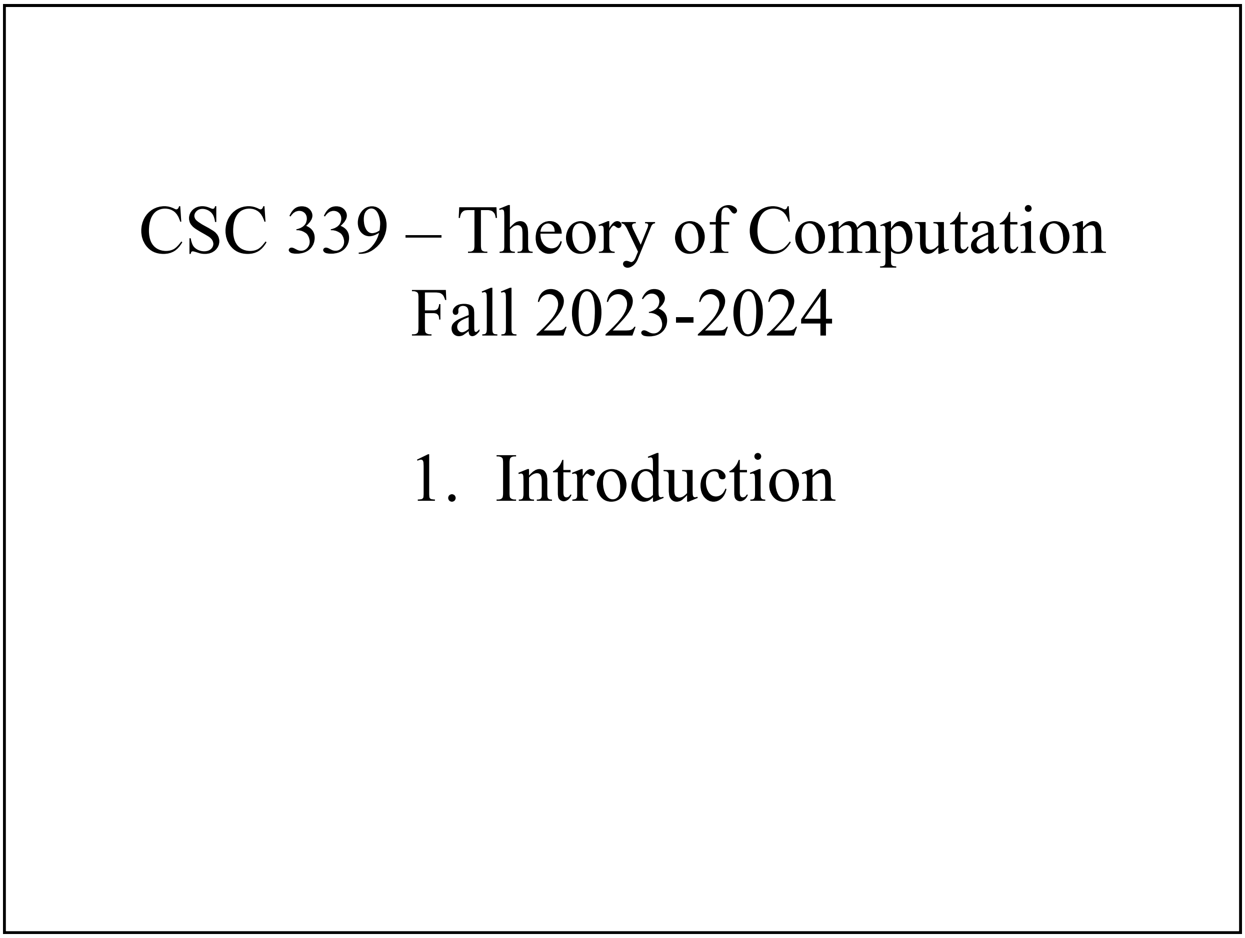
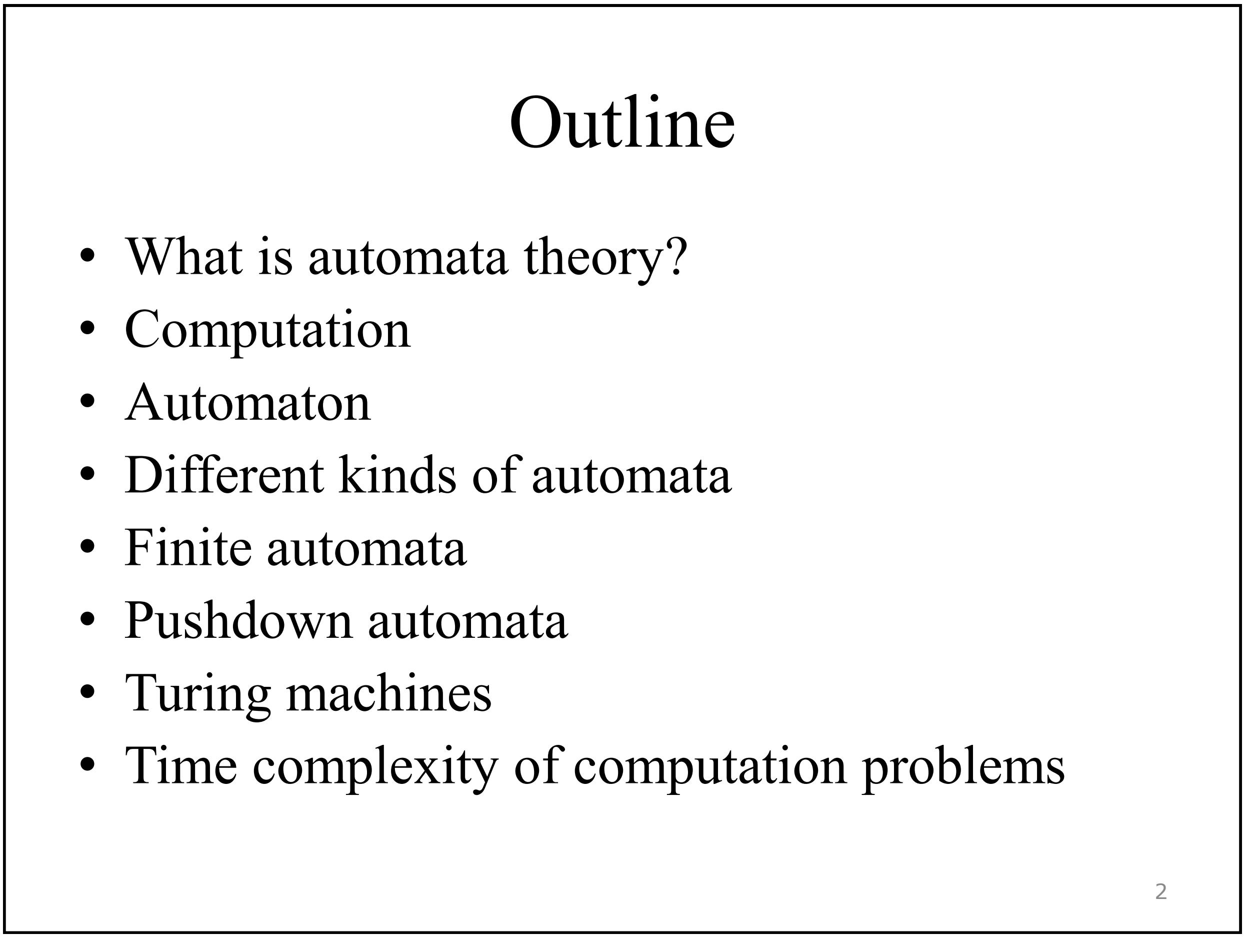


CSC 339 – Theory of Computation
Fall 2023-2024

1. Introduction



Outline

- What is automata theory?
- Computation
- Automaton
- Different kinds of automata
- Finite automata
- Pushdown automata
- Turing machines
- Time complexity of computation problems

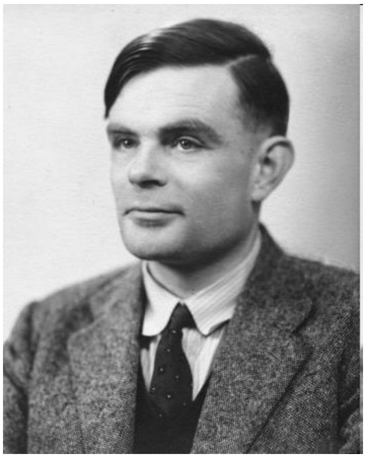
What is Automata Theory?

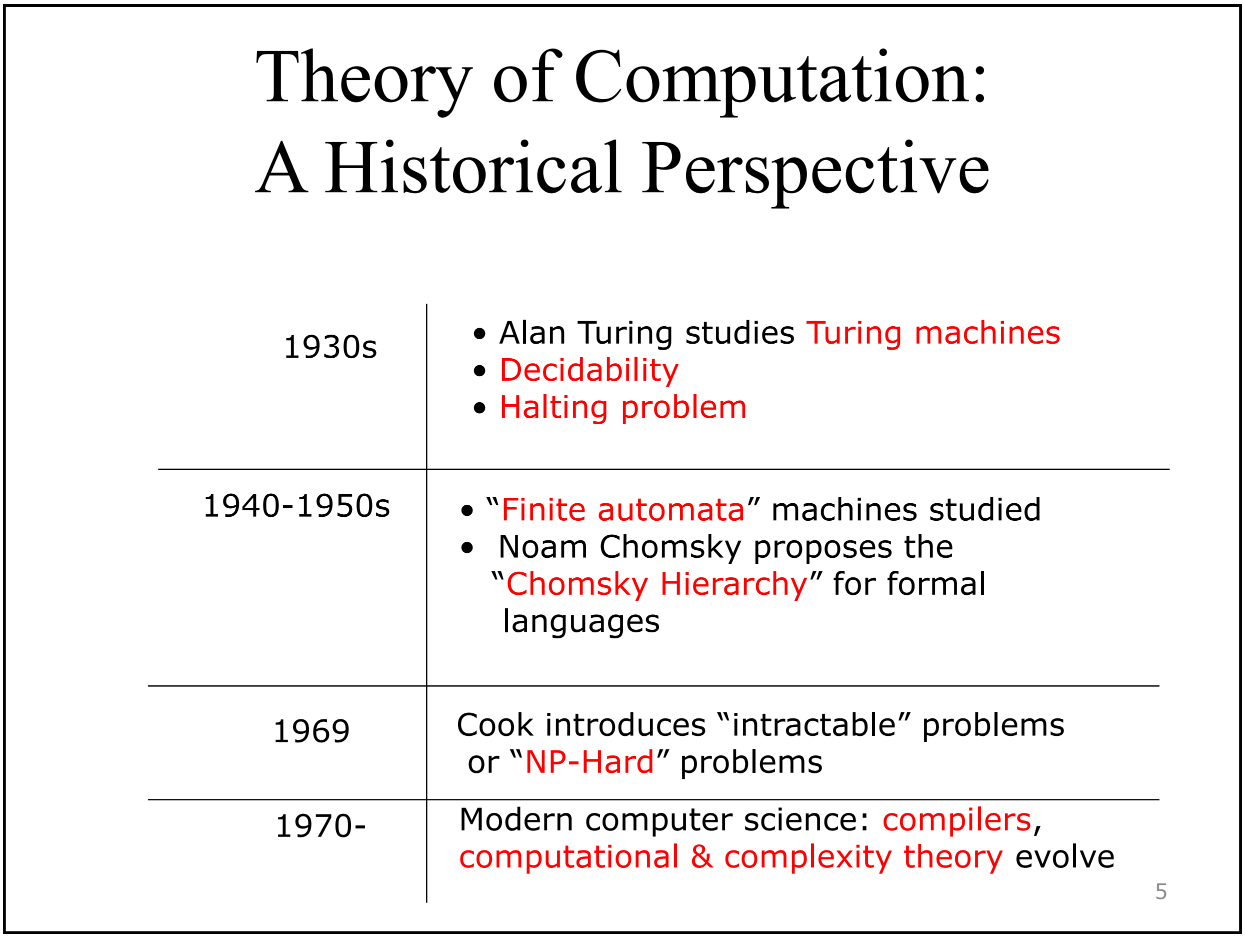
- *Study of abstract computing devices, or "machines"*
- Automaton = an abstract computing device
  - **Note:** A "device" need not even be physical hardware!
- A fundamental question in computer science:
  - Find out what different models of machines can and cannot do
  - The *theory of computation*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

Alan Turing (1912-1954)

(A pioneer of automata theory)

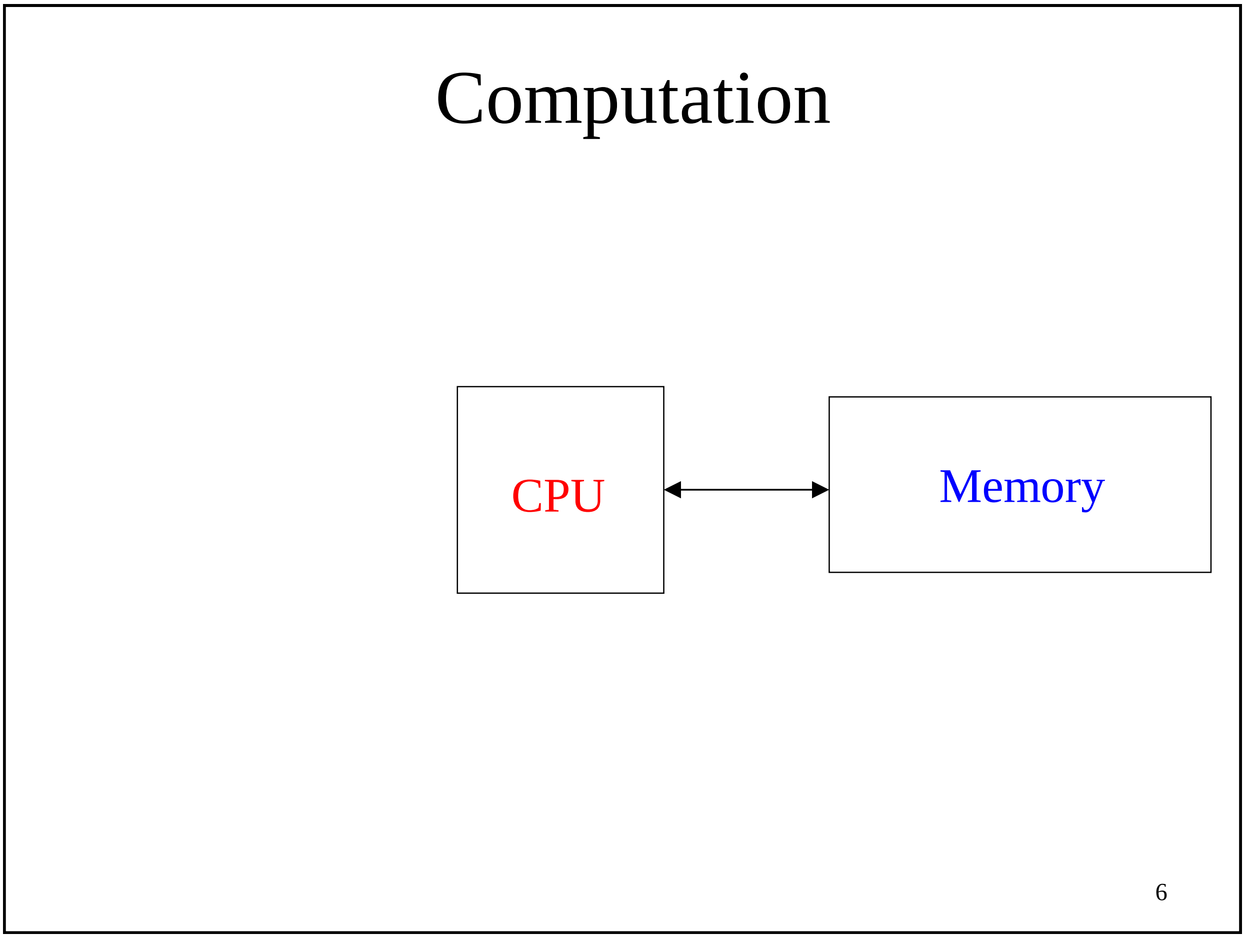
- Father of Modern Computer Science
- English mathematician
- Studied abstract machines called ***Turing machines*** even before computers existed
- The Turing test!



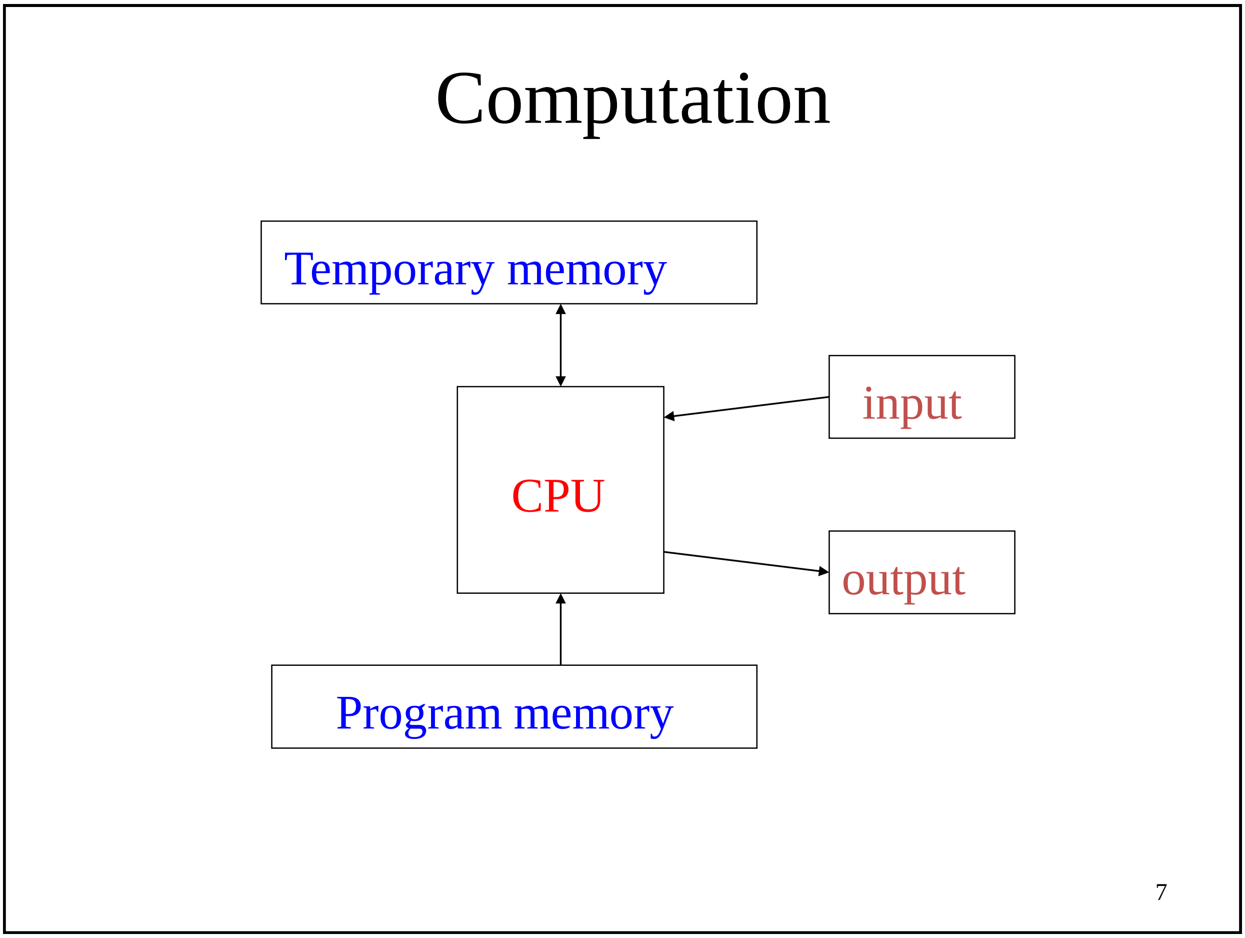


Theory of Computation: A Historical Perspective

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



Computation



Computation

Temporary memory

CPU

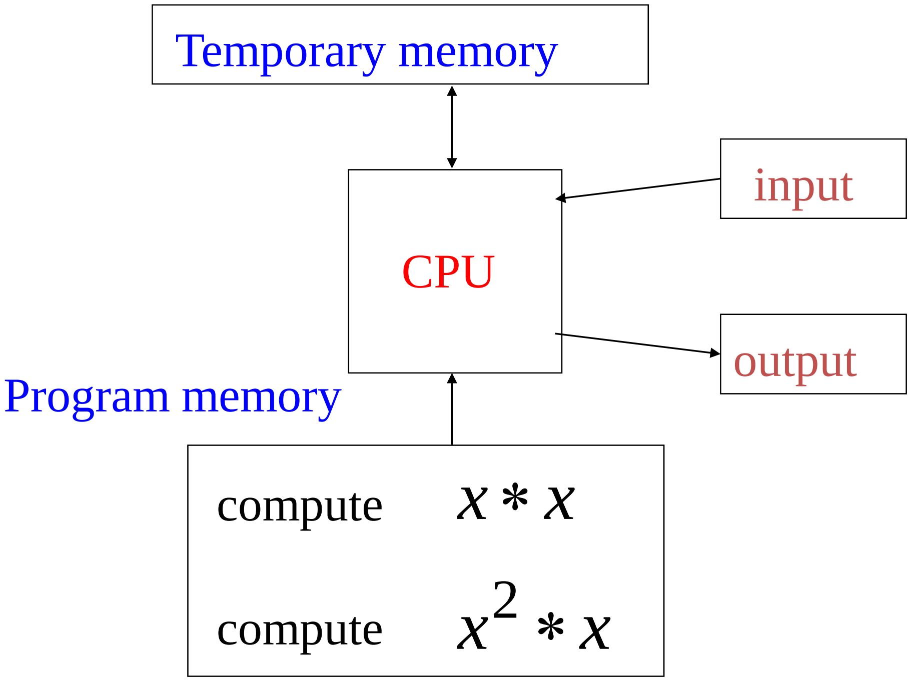
input

output

Program memory

Computation

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



Temporary memory

CPU

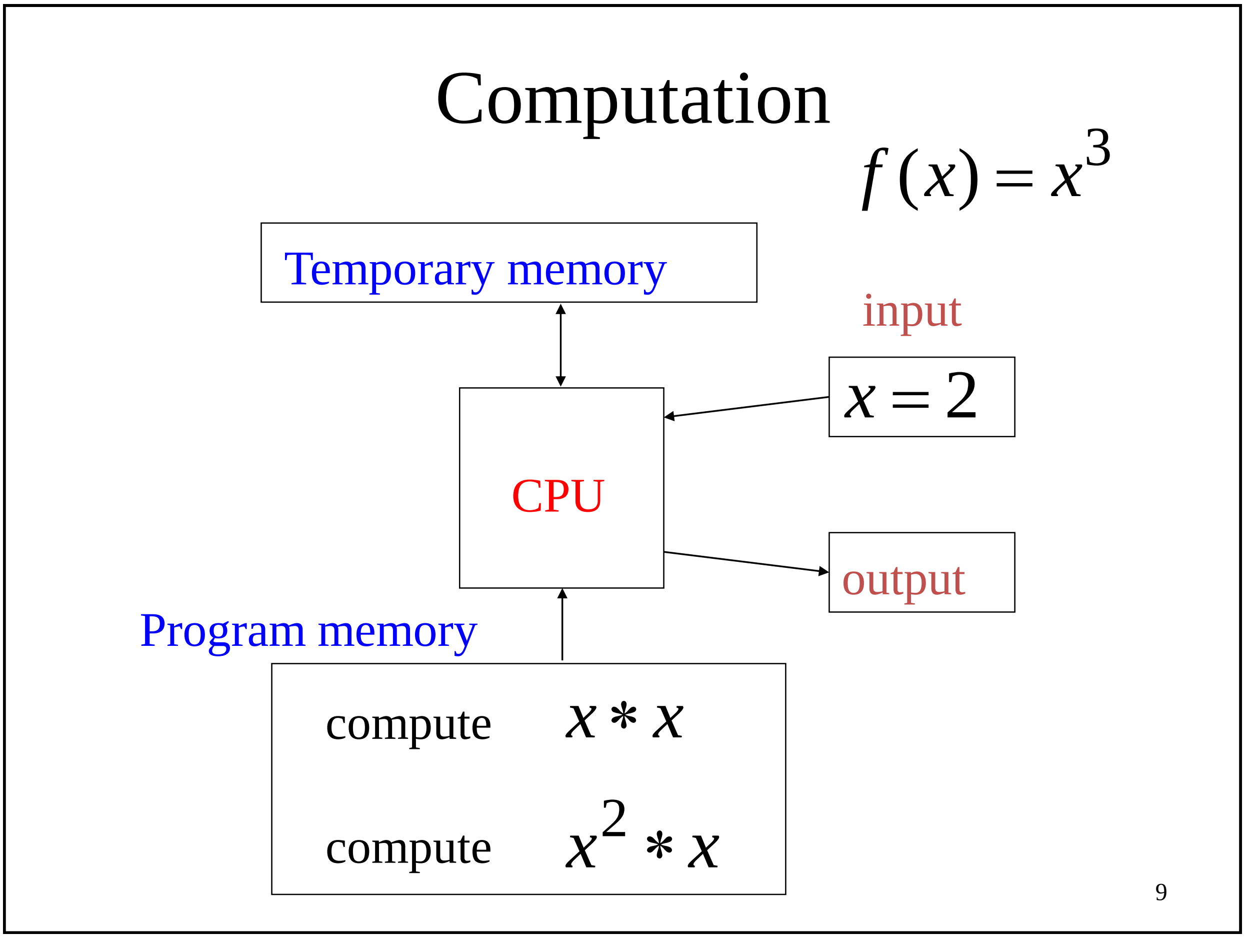
input

output

Program memory

compute $x * x$

compute $x^2 * x$



Computation

$f(x) = x^3$

Temporary memory

input

$x = 2$

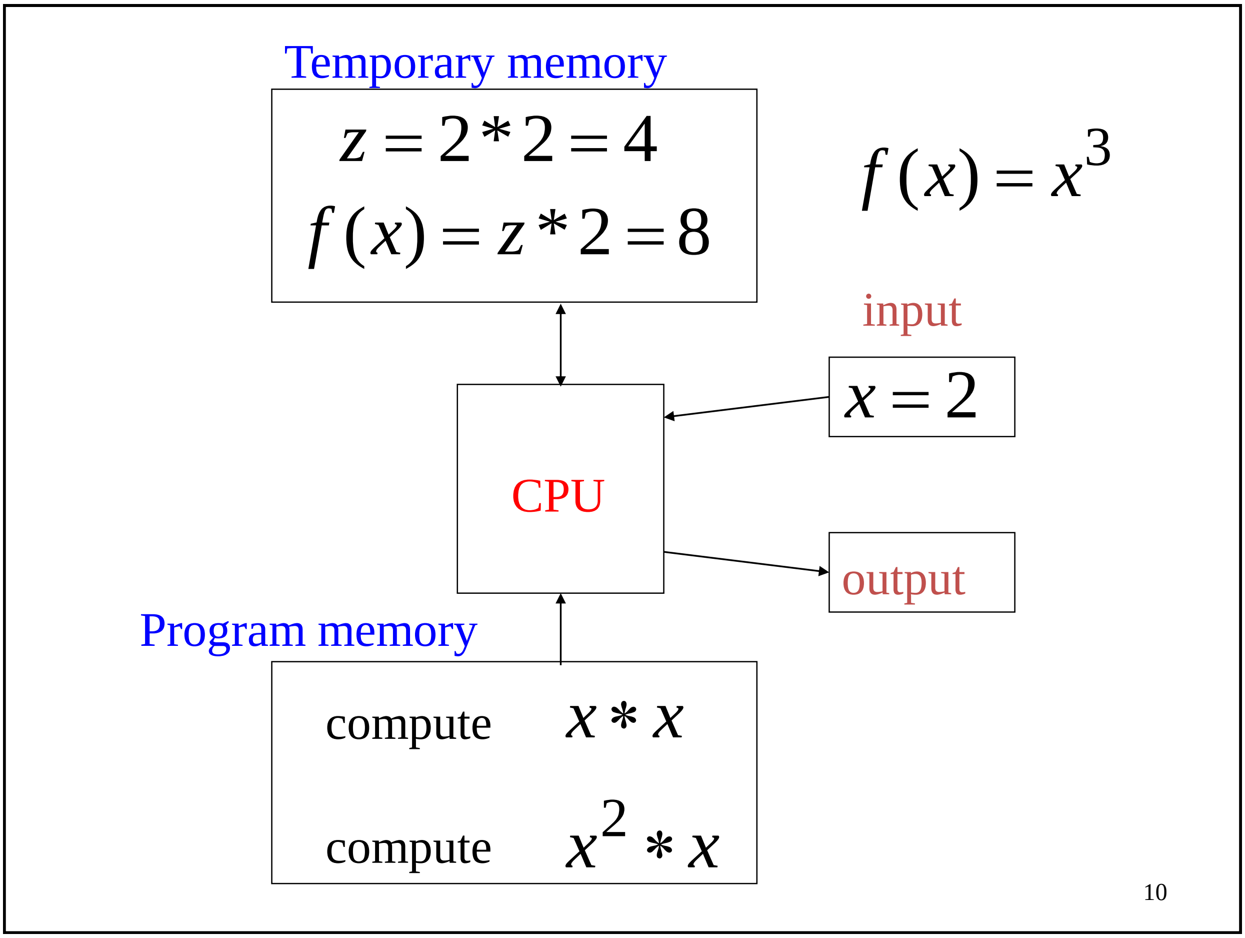
CPU

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

$$x = 2$$

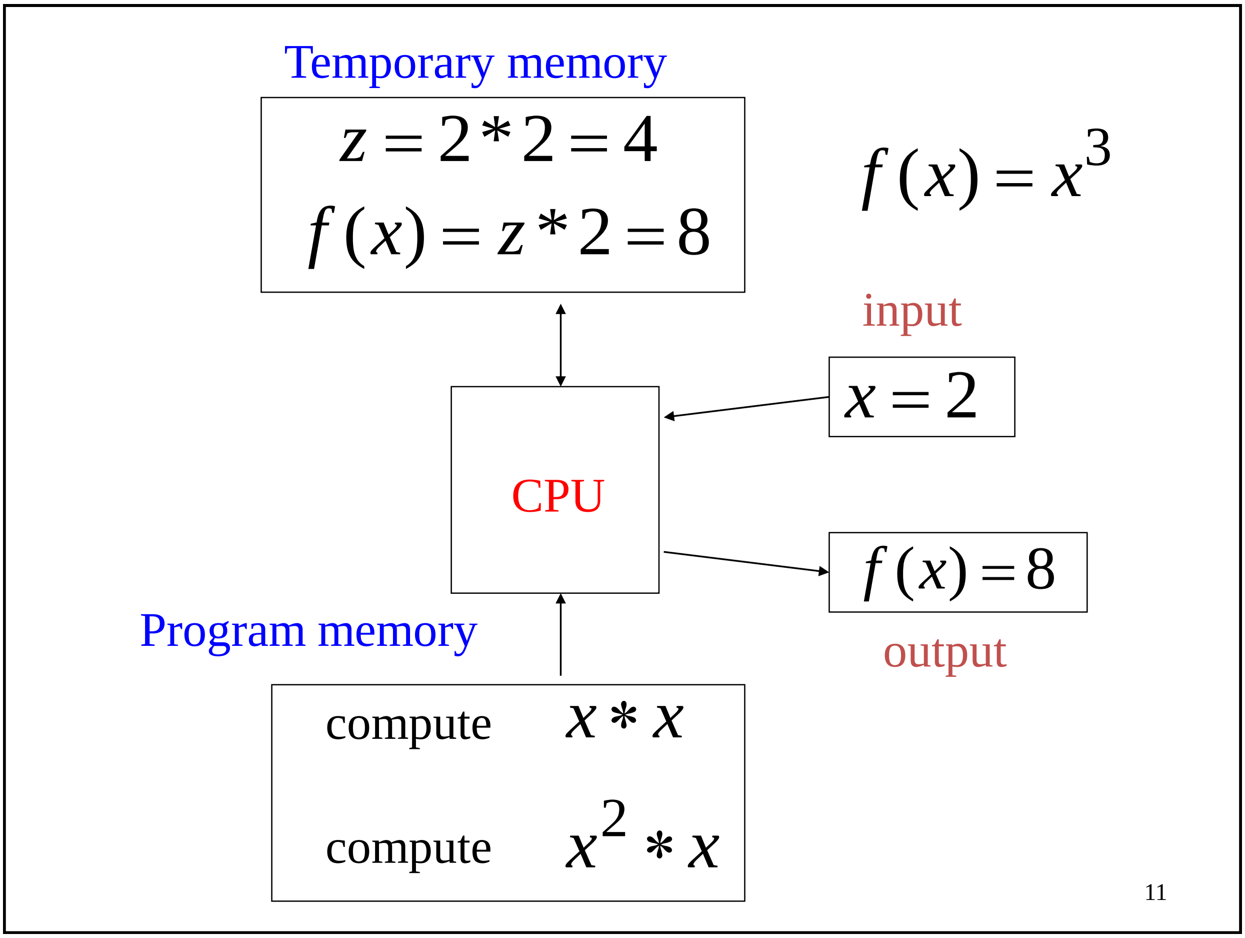
CPU

output

Program memory

compute $x*x$

compute x^2*x



Temporary memory

$z = 2 * 2 = 4$

$f(x) = z * 2 = 8$

$f(x) = x^3$

input

$x = 2$

CPU

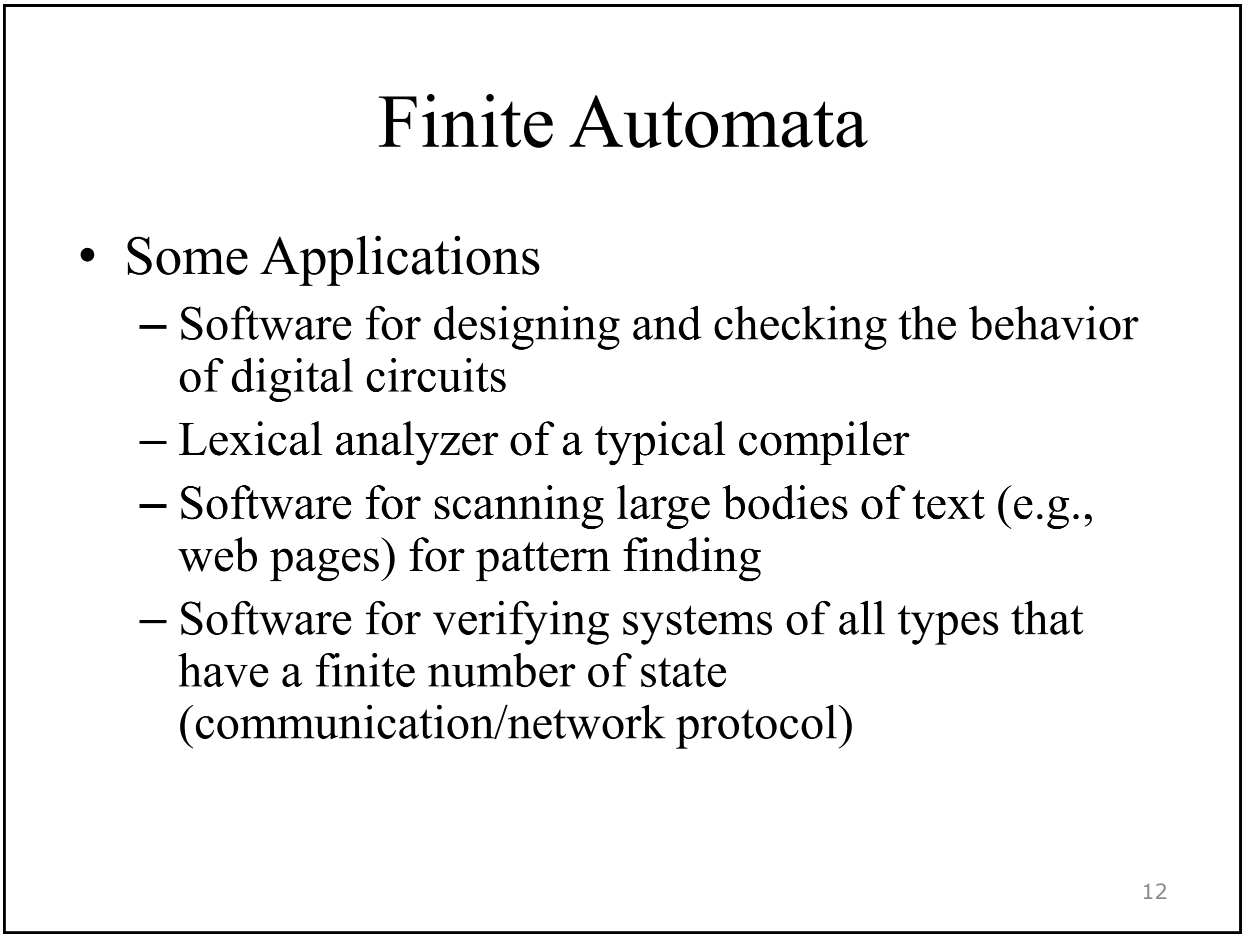
$f(x) = 8$

output

Program memory

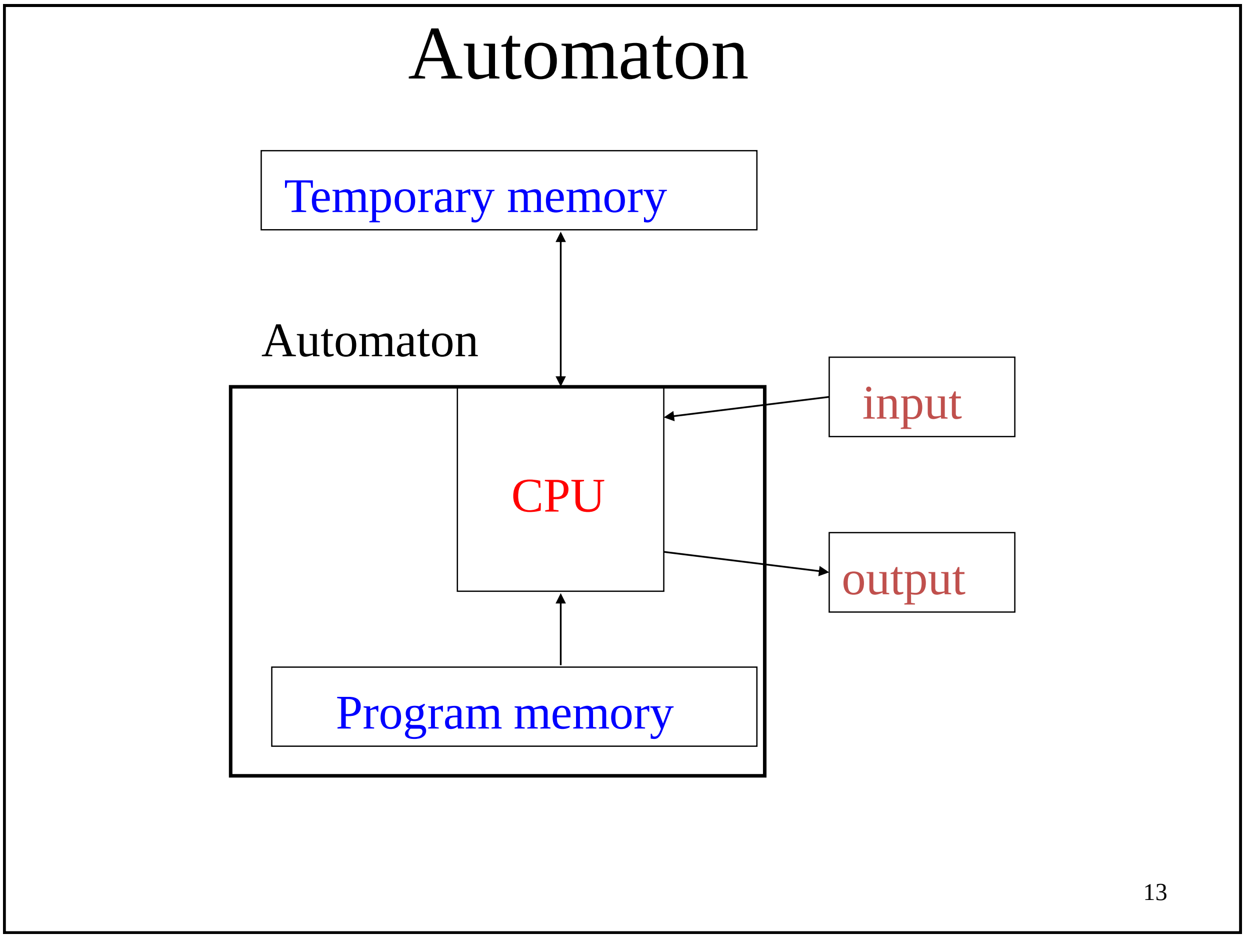
compute $x * x$

compute $x^2 * x$



Finite Automata

- Some Applications
  - Software for designing and checking the behavior of digital circuits
  - Lexical analyzer of a typical compiler
  - Software for scanning large bodies of text (e.g., web pages) for pattern finding
  - Software for verifying systems of all types that have a finite number of state (communication/network protocol)



Automaton

Temporary memory

Automaton

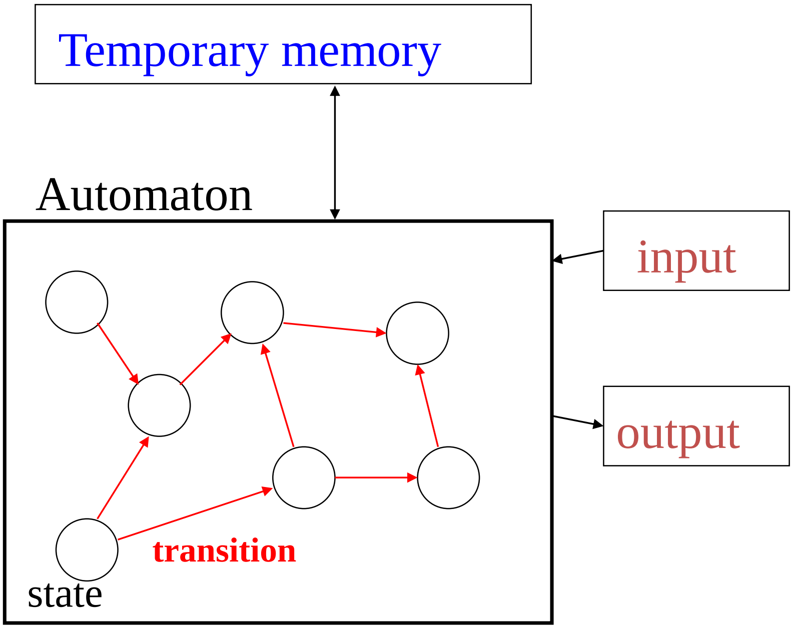
CPU

input

output

Program memory

Automaton



Temporary memory

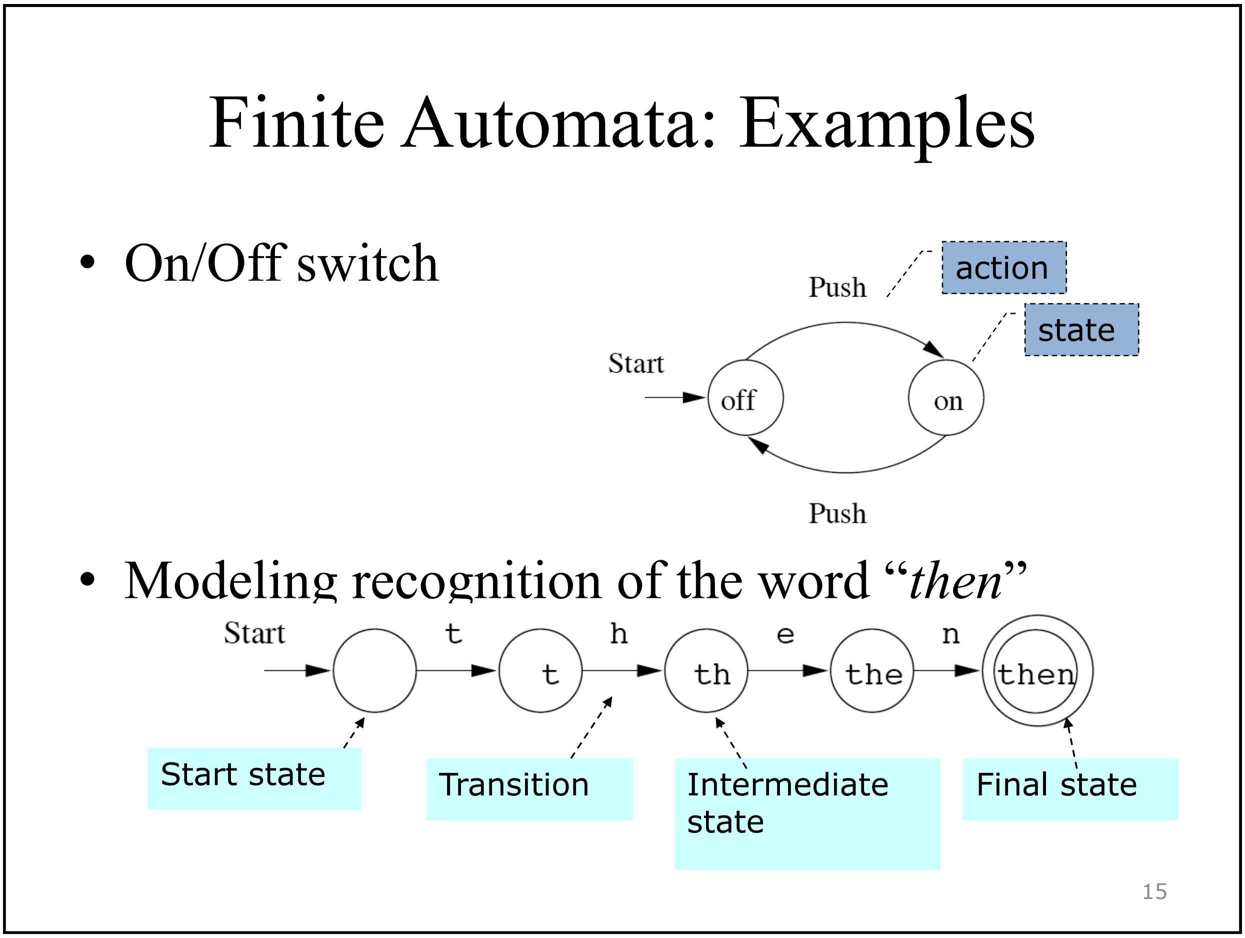
Automaton

input

output

transition

state



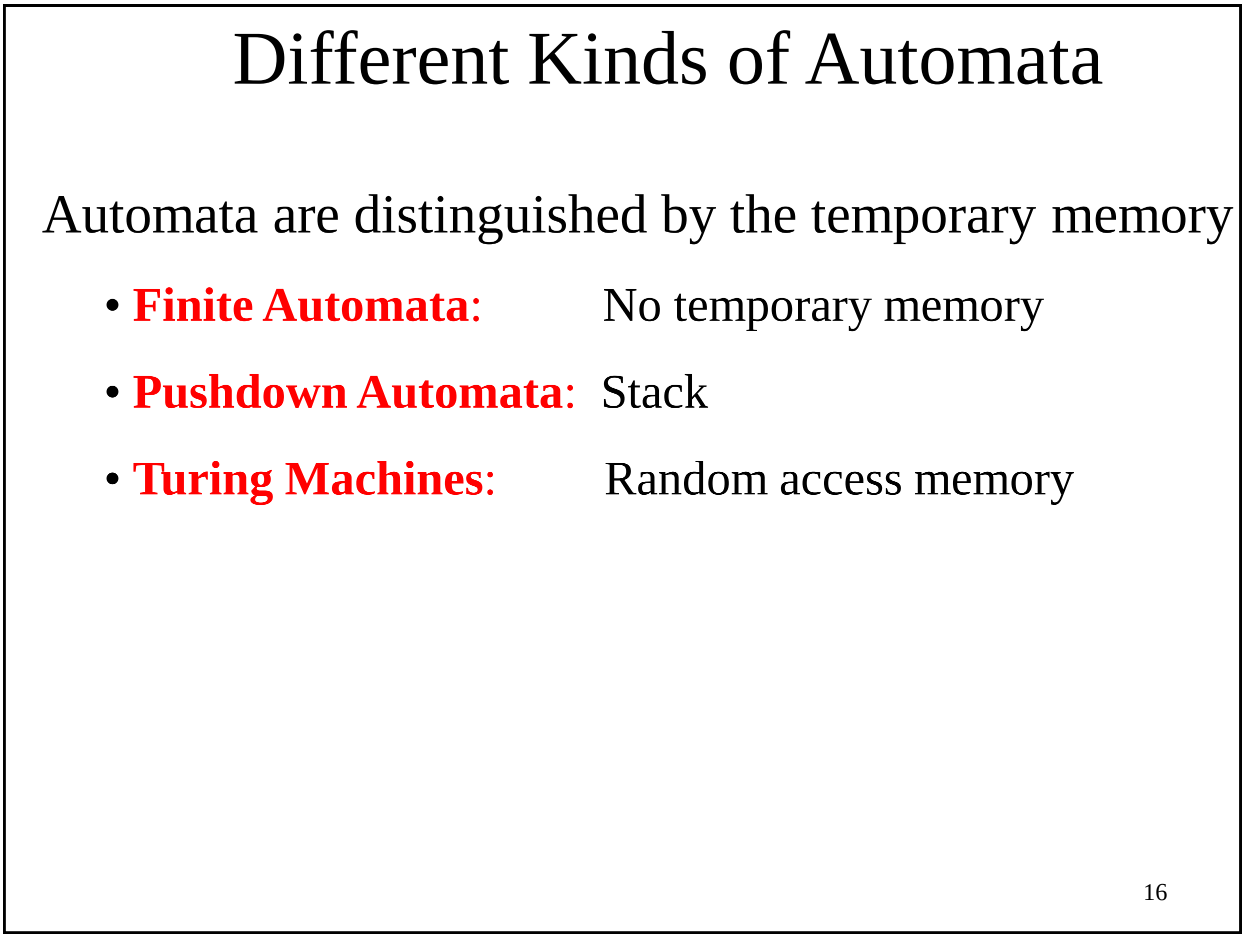
Finite Automata: Examples

- On/Off switch

Push
action
state
Start
off
on
Push

- Modeling recognition of the word “*then*”

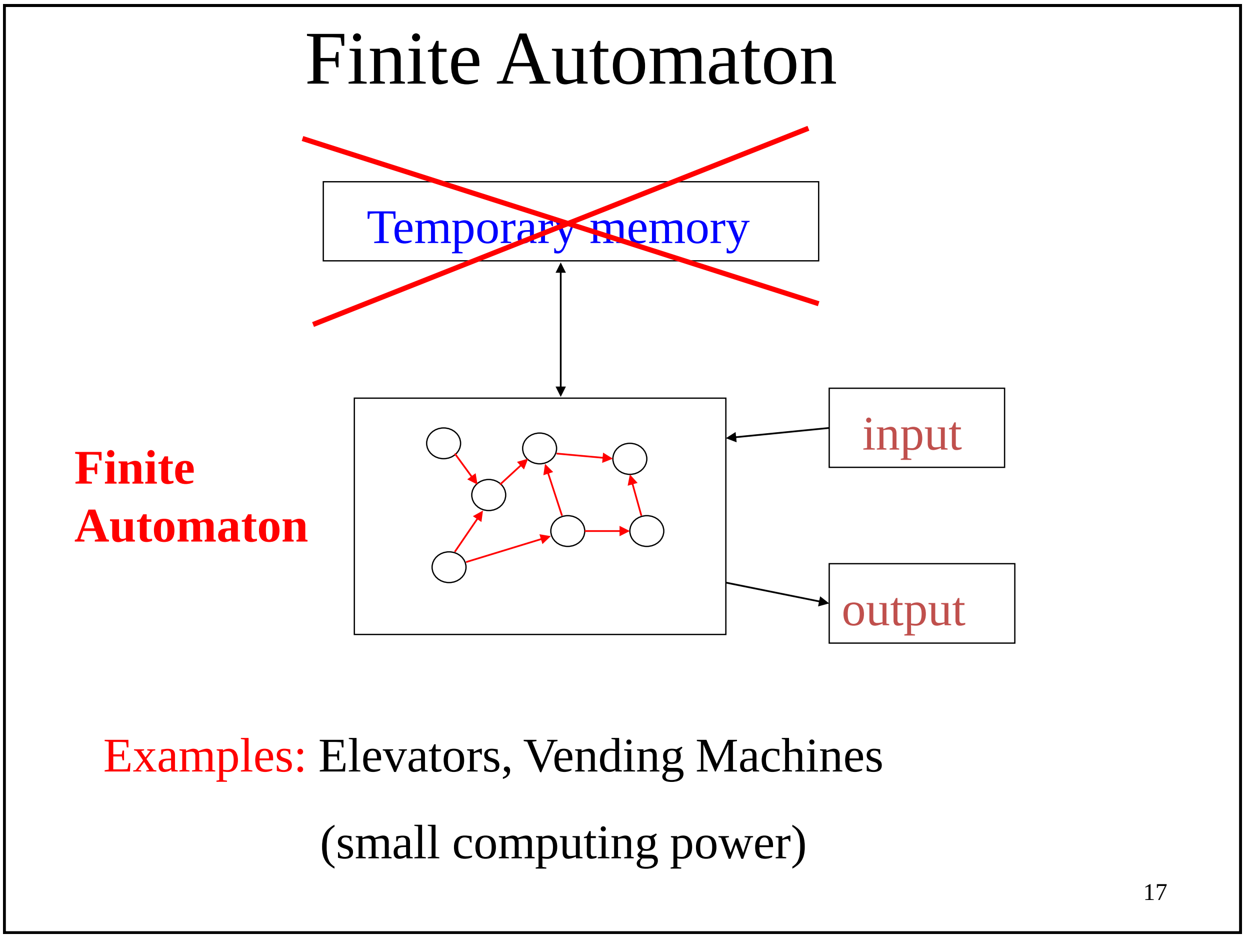
Start
t
h
e
n
t
th
the
then
Start state
Transition
Intermediate state
Final state



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

~~Temporary memory~~

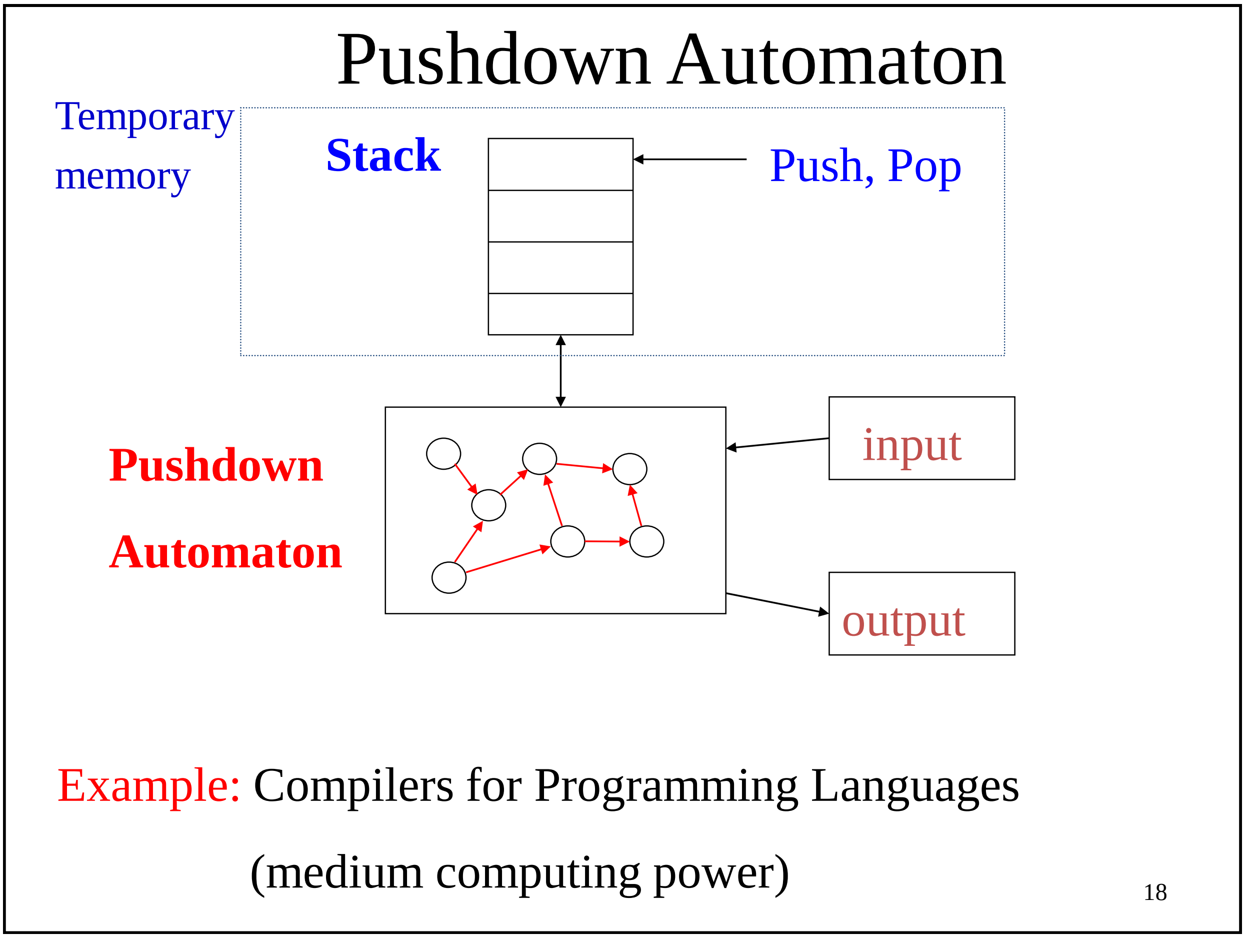
Finite Automaton

input

output

Examples: Elevators, Vending Machines

(small computing power)



Pushdown Automaton

Temporary memory

Stack

Push, Pop

Pushdown Automaton

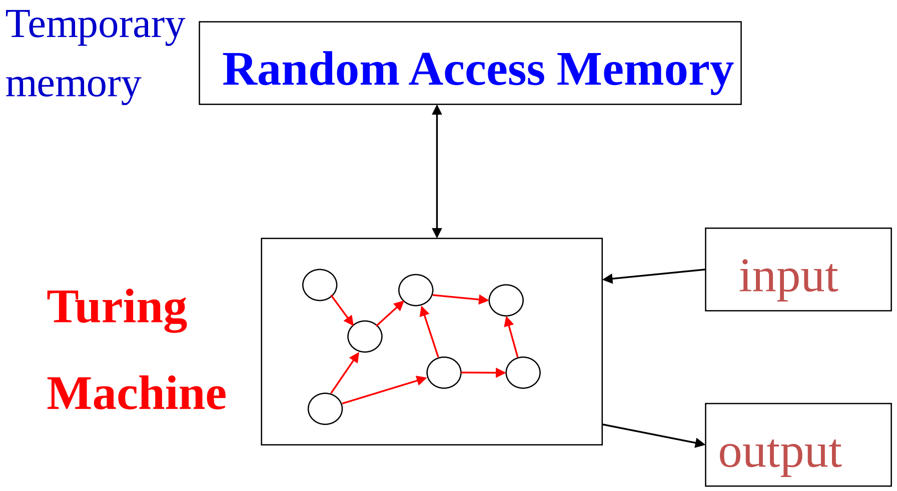
input

output

Example: Compilers for Programming Languages

(medium computing power)

Turing Machine



Temporary memory

Random Access Memory

Turing Machine

input

output

Example: Any Algorithm

(highest computing power)

Power of Automata

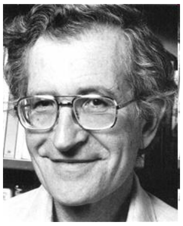
Simple problems | **More complex problems** | **Hardest problems**

Finite Automata < Pushdown Automata < Turing Machine

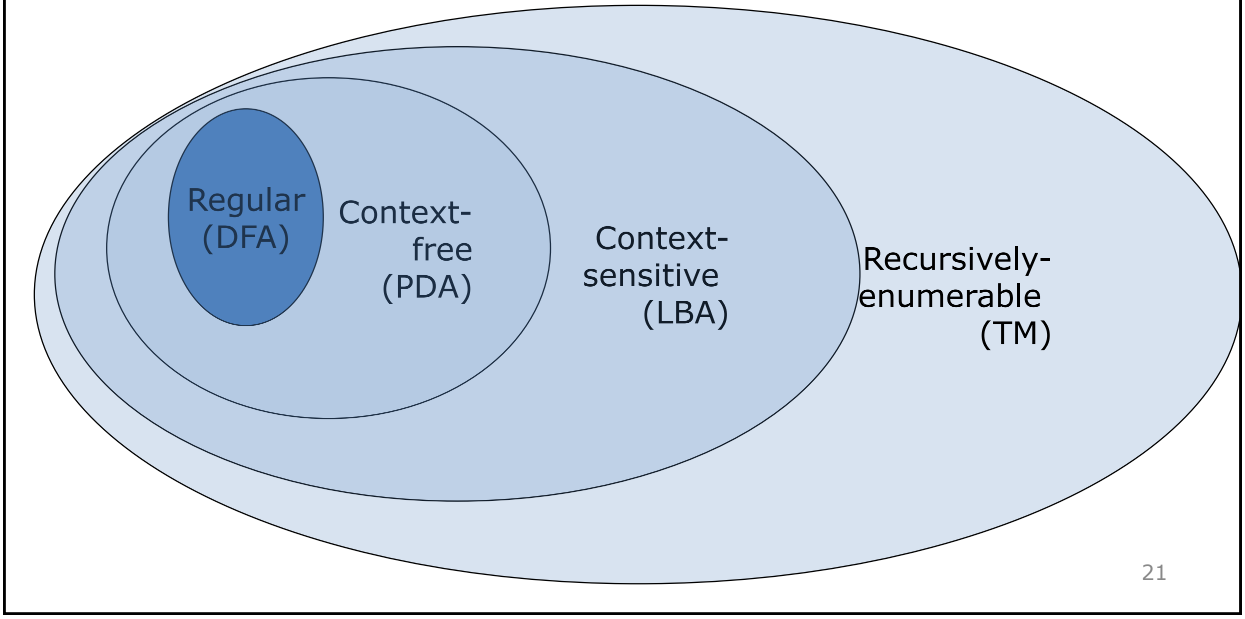
Less power → More power

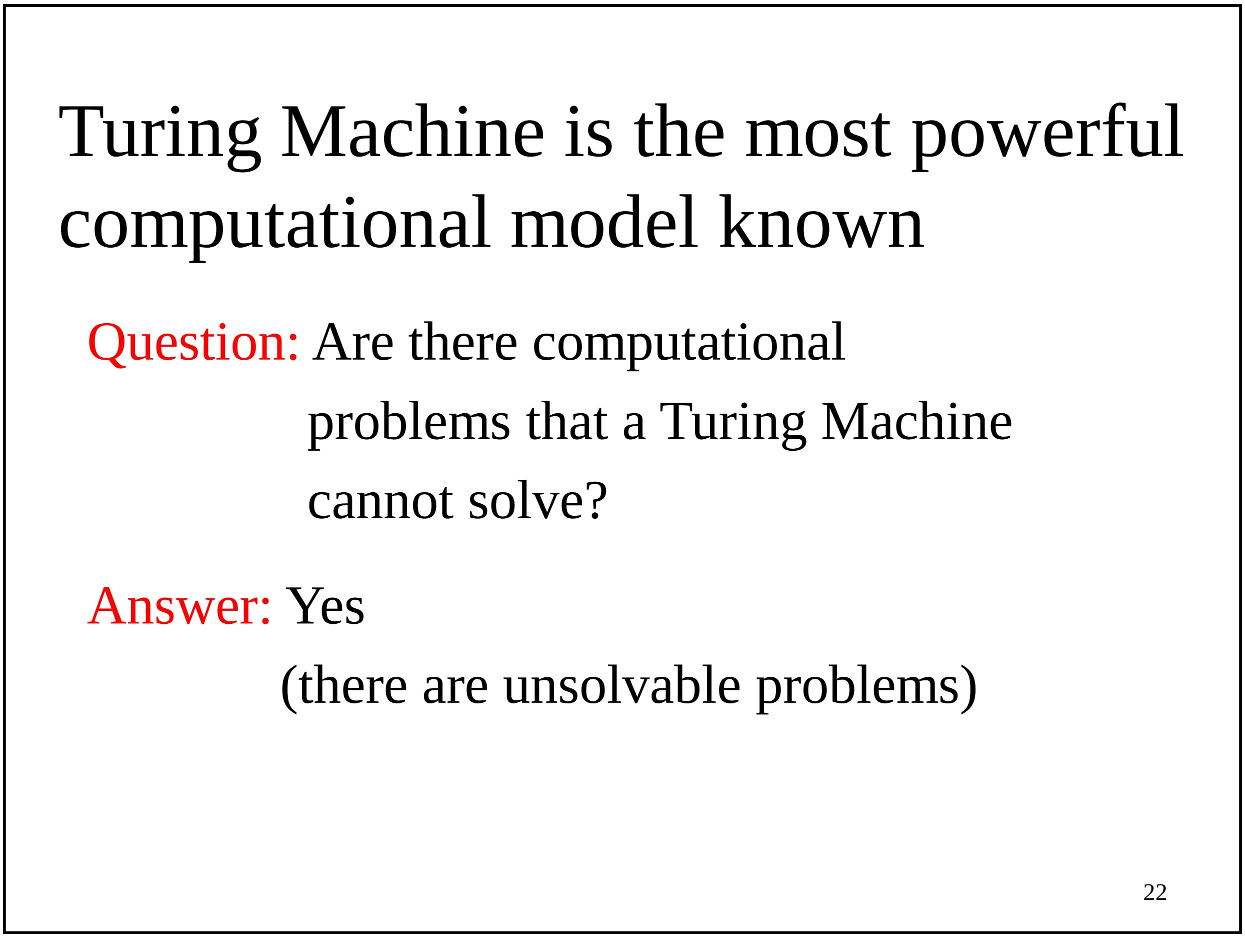
Solve more computational problems

The Chomsky Hierachy



- A containment hierarchy of classes of formal languages



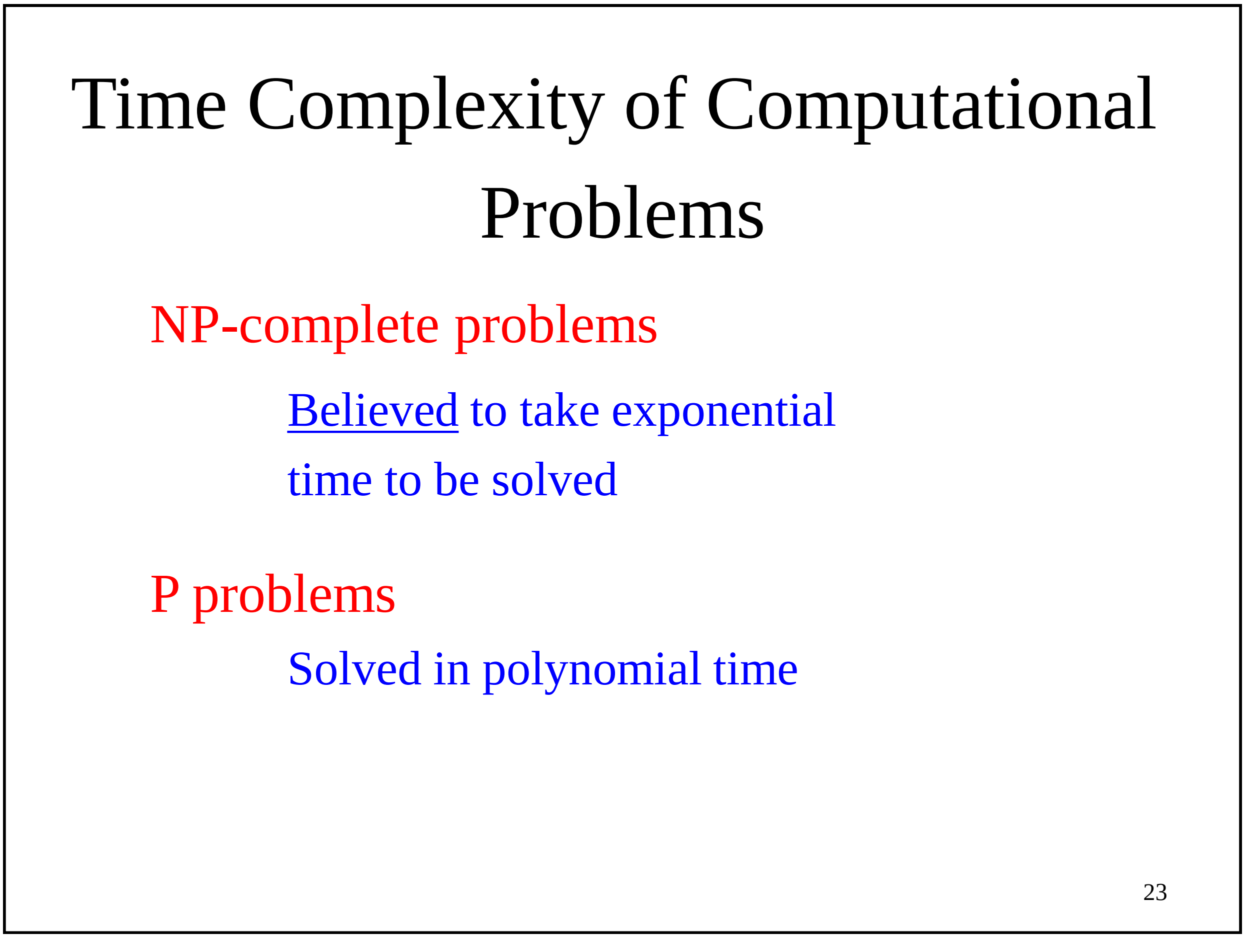


Turing Machine is the most powerful computational model known

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

Answer: Yes

(there are unsolvable problems)



Time Complexity of Computational Problems

NP-complete problems

Believed to take exponential time to be solved

P problems

Solved in polynomial time