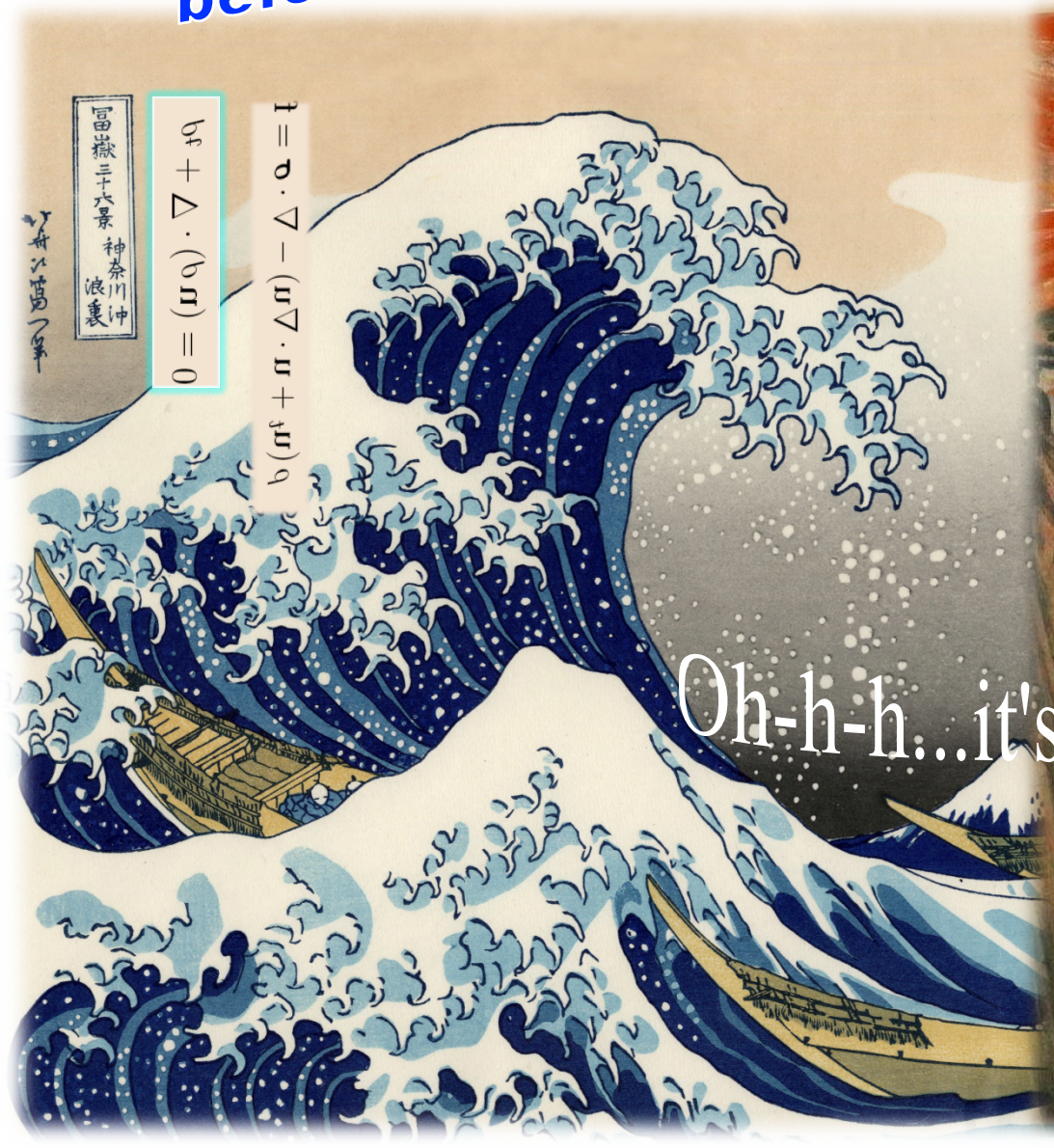
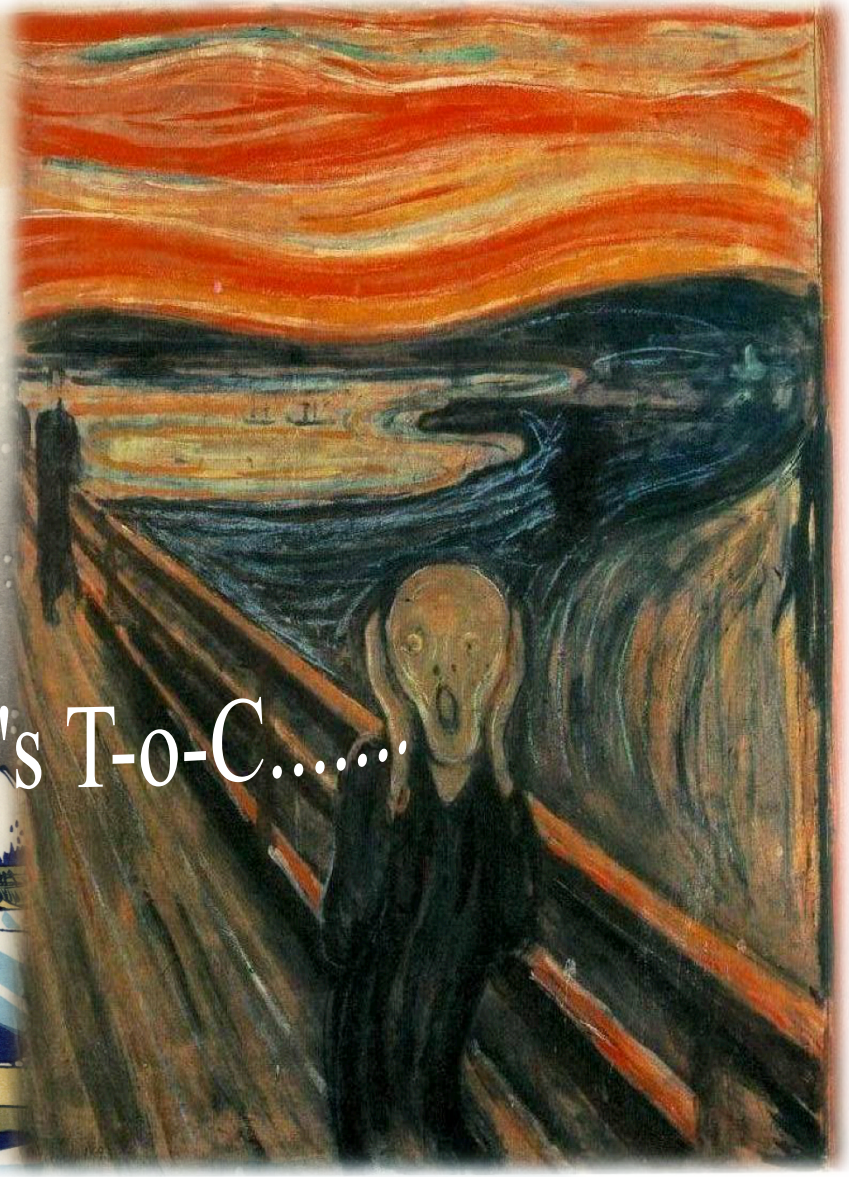
All you need to know about CS before you graduate

Intro to Theory of Computation

www. inf.unisi.ch\faculty\sharygina





Oh-h-h...it's T-o-C.....

Today's lecture

- Turing machine as a universal model of computation
  - History
  - Formalism
  - Demo

A bit of history

- Philosofical perspective and historical outline

  Let's recall the history of mathematics

Turing machine

- Fundamental concept in computer science

- Invented by Alan Turing (1912 – 1954) in 1936

Theory of Computation

- Computability Theory:

  What problems can(not) computers solve?

- Complexity Theory:

  What makes some problems computationally hard and others easy?

Theory of Computation

- Computability Theory:

  What problems can(not) computers solve?

- Complexity Theory:

  What makes some problems computationally hard and others easy?

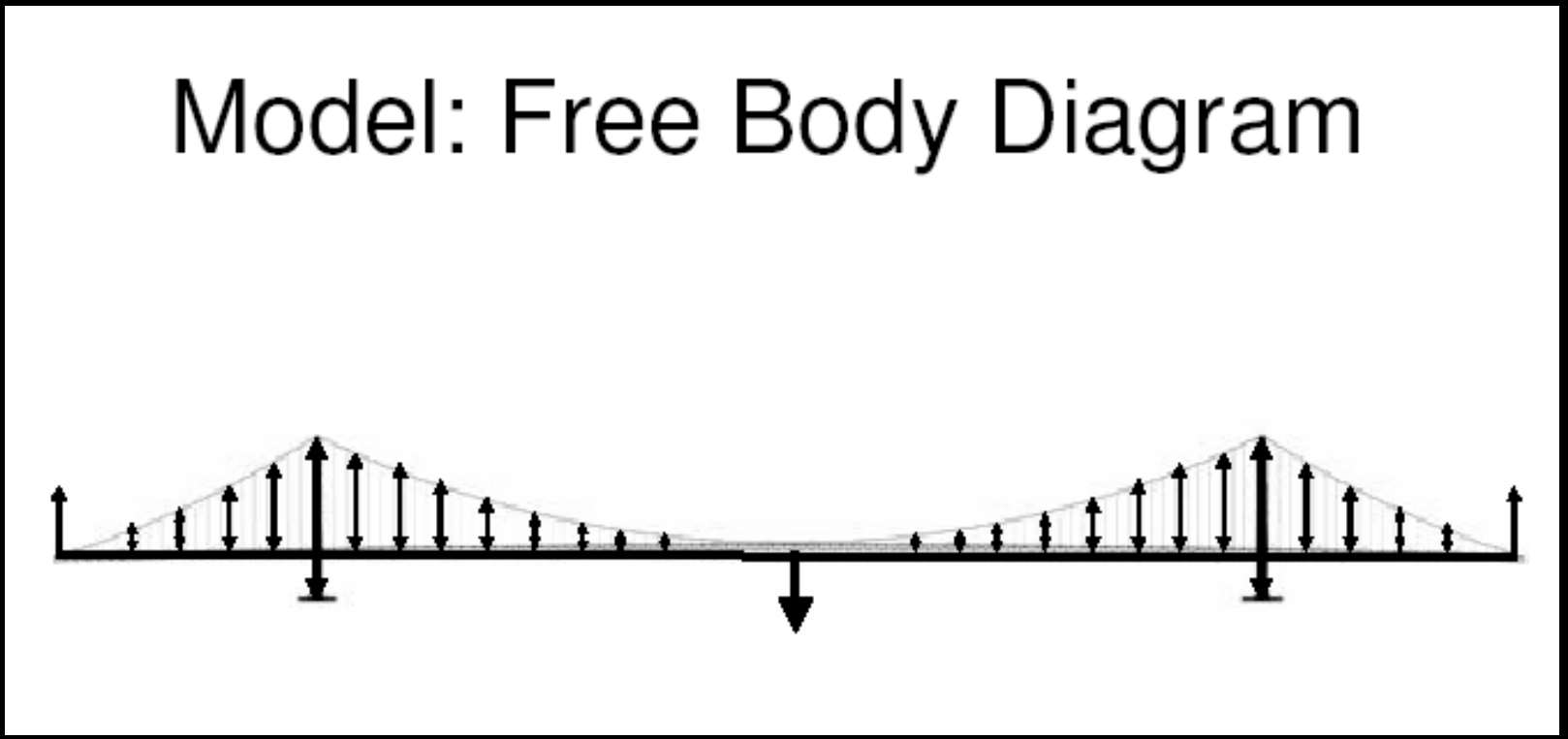
What is role of Turing and the purpose of Turing machine for above?

We need a device that could be used to compare problems and their solutions – Turing machine is this device in computing

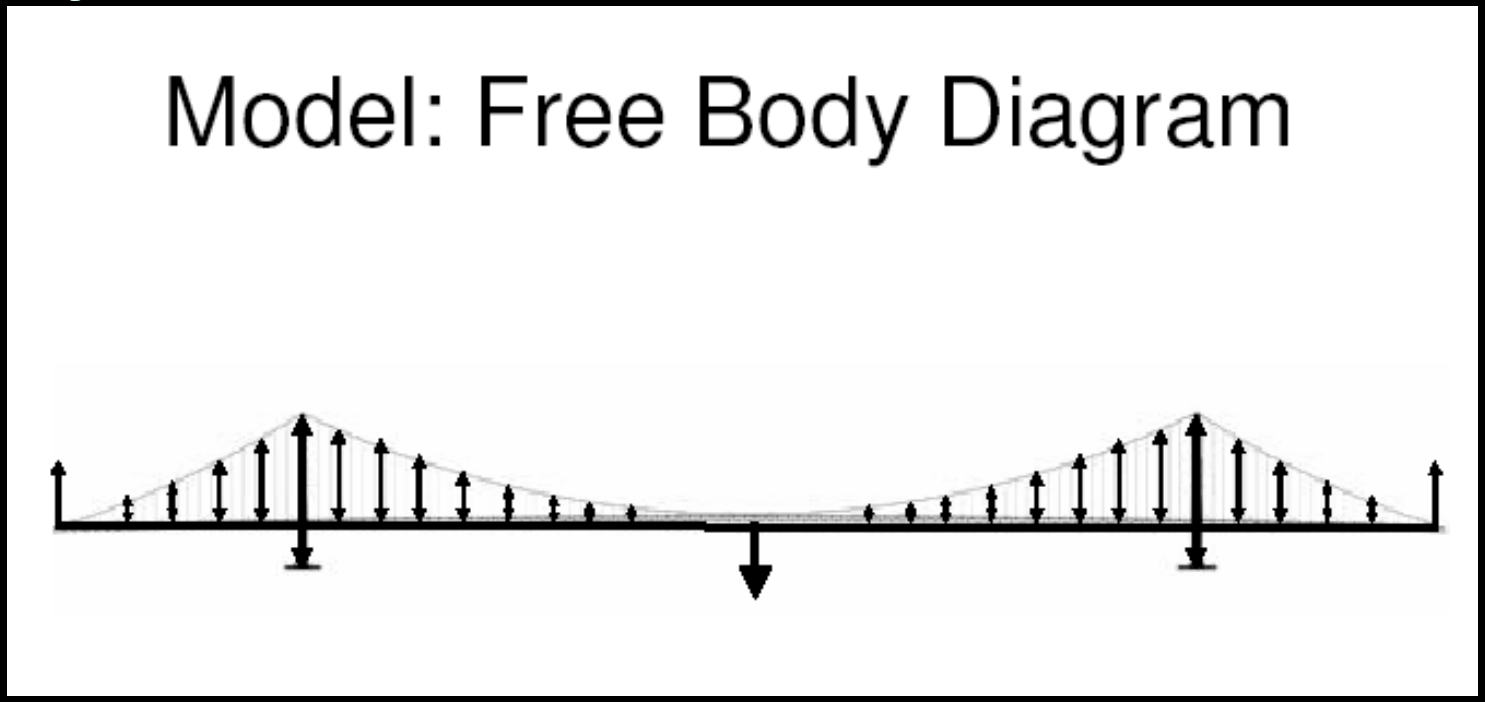
Why do we need models?

- Let's see connection with classical engineering disciplines

Why build a model?



Why build a model?



Analyzing a model helps to build reliable devices (e.g., bridges, etc)

In Computer Science we do the same with computers and computer programs

Why do we need models?

- Let's see connection with classical engineering disciplines
- Can you make a parallel to computing science?

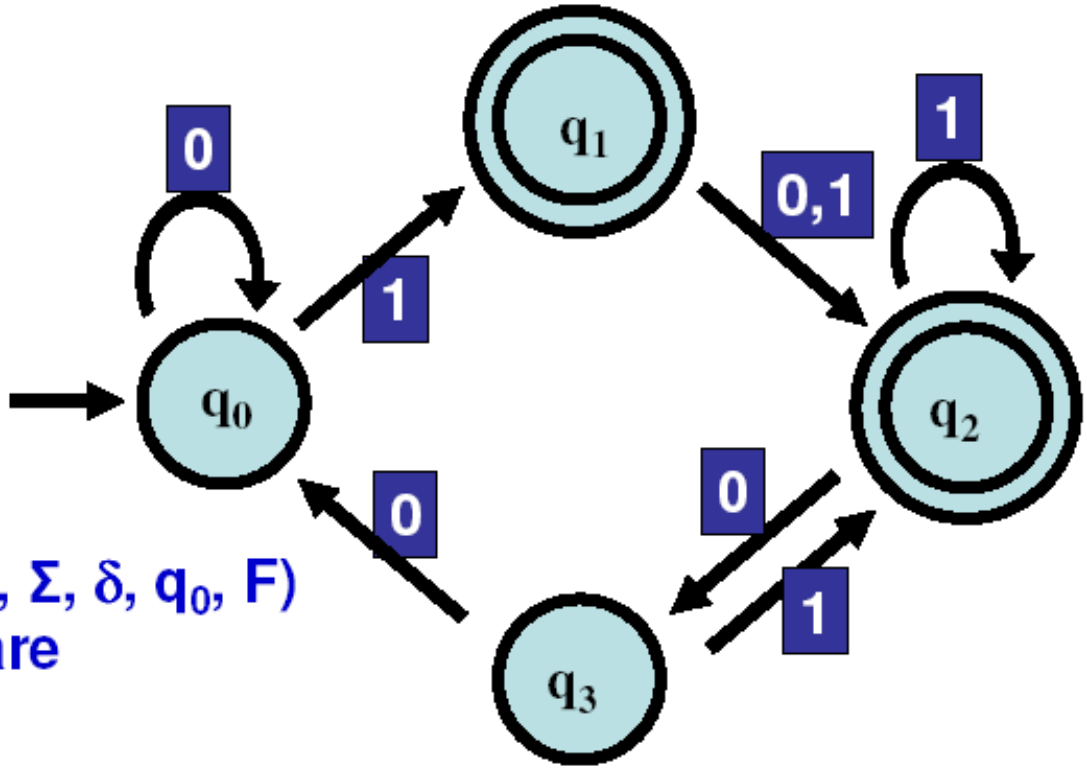
Programs and their models?

- Sequential programs can be modeled as pushdown automata
  - E.g., Microsoft Static Driver Verifier looks for bugs in device drivers after creating a pushdown automaton models for them
  - Etc… can you give any other examples?

What models do we know?

- Finite Automata
- Pushdown Automata

Let's recall DFA terminology



$M = (Q, \Sigma, \delta, q_0, F)$

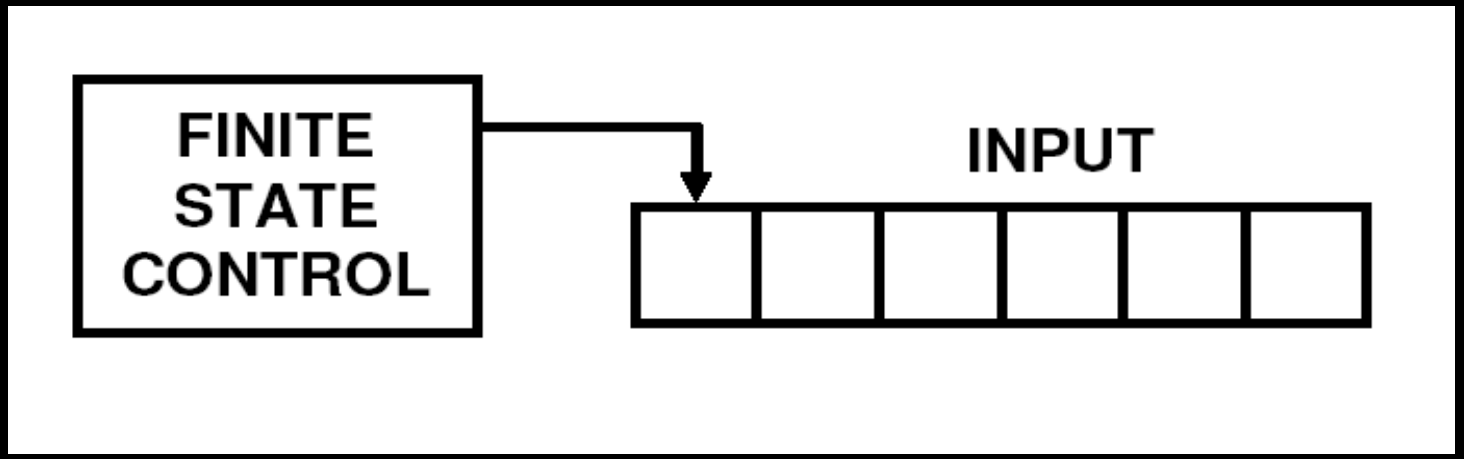
What are

1. Q
2. F
3. Σ
4. δ

Let's recall acceptance of DFA

- M is a DFA with alphabet Σ
- An input word (string of symbols) $w = w_1 w_2 \ldots w_n$, where each $w_i \in \Sigma$
- M accepts w "if after fully reading w it ends up in an accept (final) state"
  - if there is a sequence of states $s_0, s_1, \ldots, s_n$ such that
    - $s_0 = q_0$
    - $s_i = \delta(s_{i-1}, w_i)$
    - $s_n \in F$

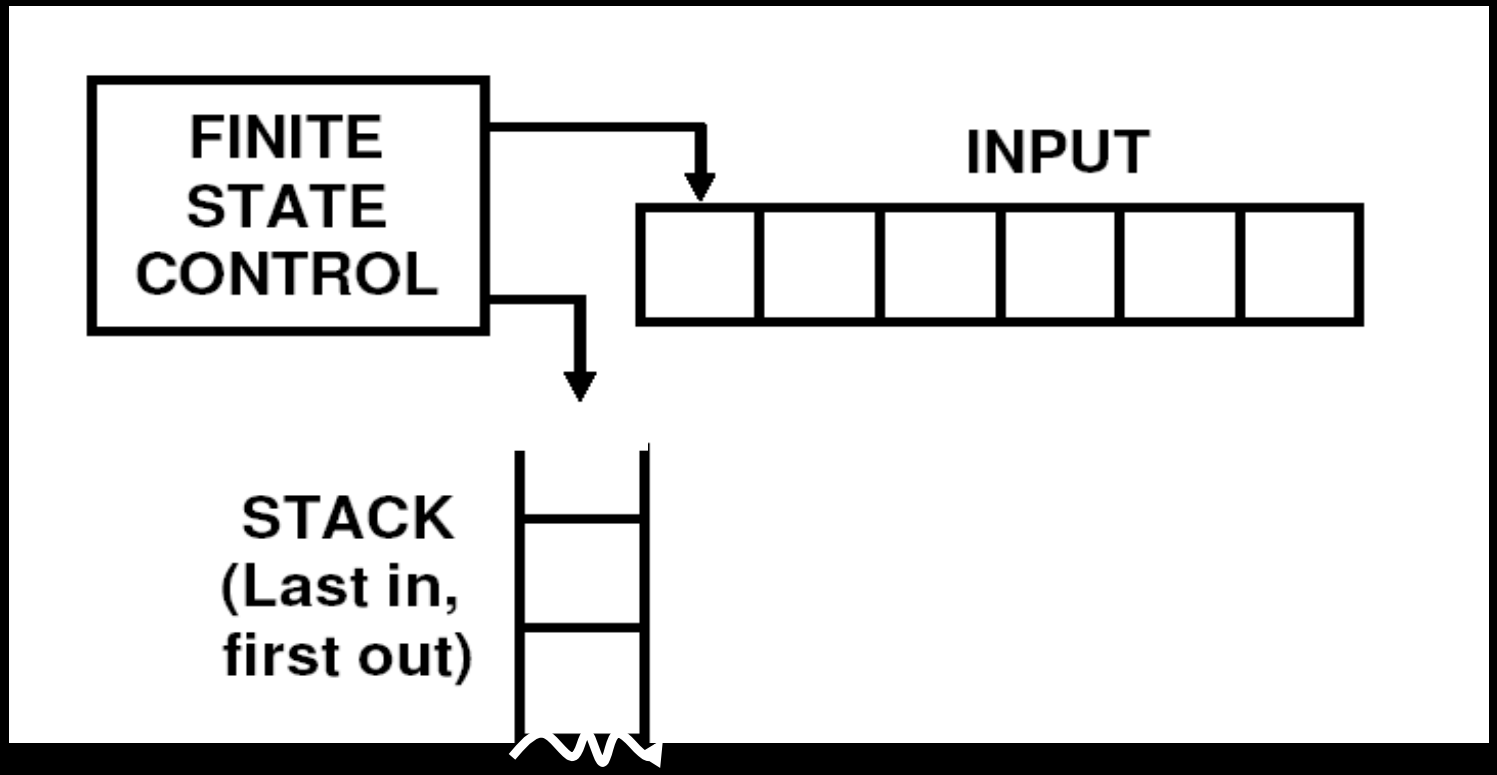
Finite Automaton



This is a computer with limited and restricted memory

Why memory is limited and restricted?

Pushdown Automaton



Let's recall Definition of PDA

Definition: A (non-deterministic) PDA is a tuple $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where:

Q is a finite set of states

Σ is the input alphabet

Γ is the stack alphabet

$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow 2^{Q \times \Gamma_\varepsilon}$ (non-determinism)

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accept states

2^Q is the set of subsets of Q

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}, \quad \Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$

Let's recall acceptance of PDA

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a word $w \in \Sigma^*$

where $w = w_1 w_2 w_3 \ldots w_m$ with $w_i \in \Sigma_\varepsilon$

if *there exists* a sequence

$(q_0, s_0) \rightarrow (q_1, s_1) \rightarrow (q_2, s_2) \rightarrow \ldots \rightarrow (q_m, s_m)$

where

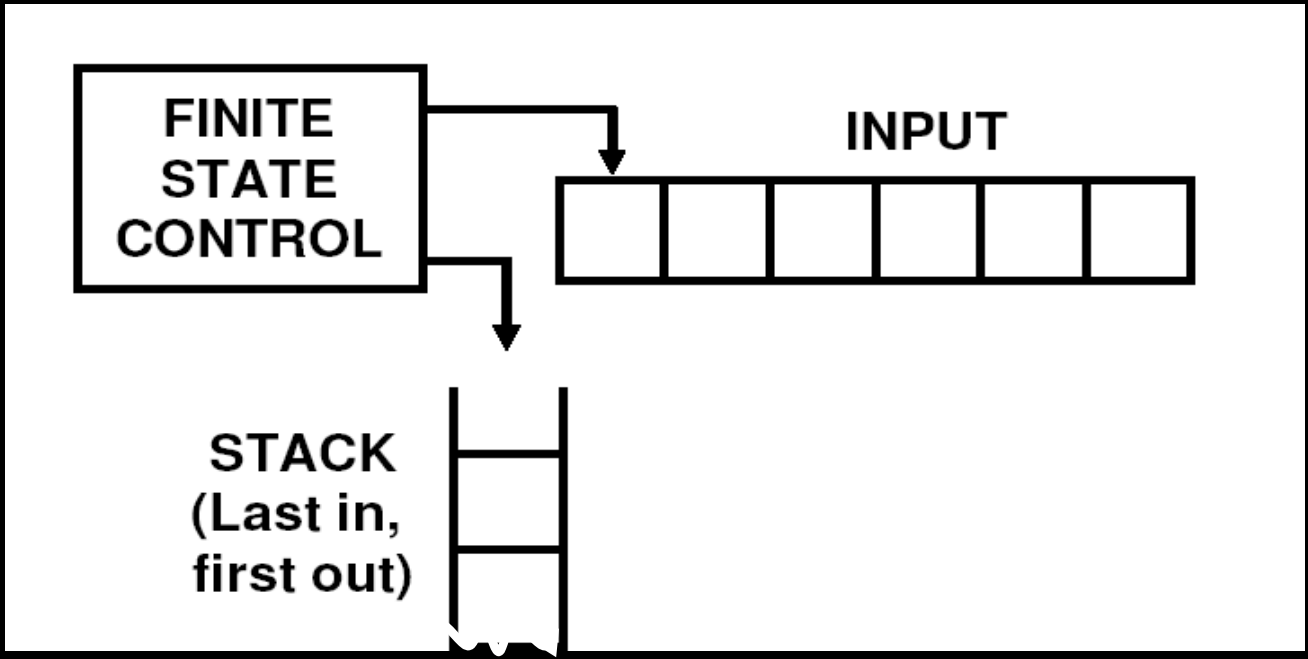
$s_i \in \Gamma^*$ (represent the stack), with $s_0 = \varepsilon$,

$q_m \in F$,

$(q_{i+1}, b) \in \delta(q_i, w_{i+1}, a)$

where $s_i = at$ and $s_{i+1} = bt$, $a,b \in \Gamma_\varepsilon$, $t \in \Gamma^*$

Pushdown Automaton



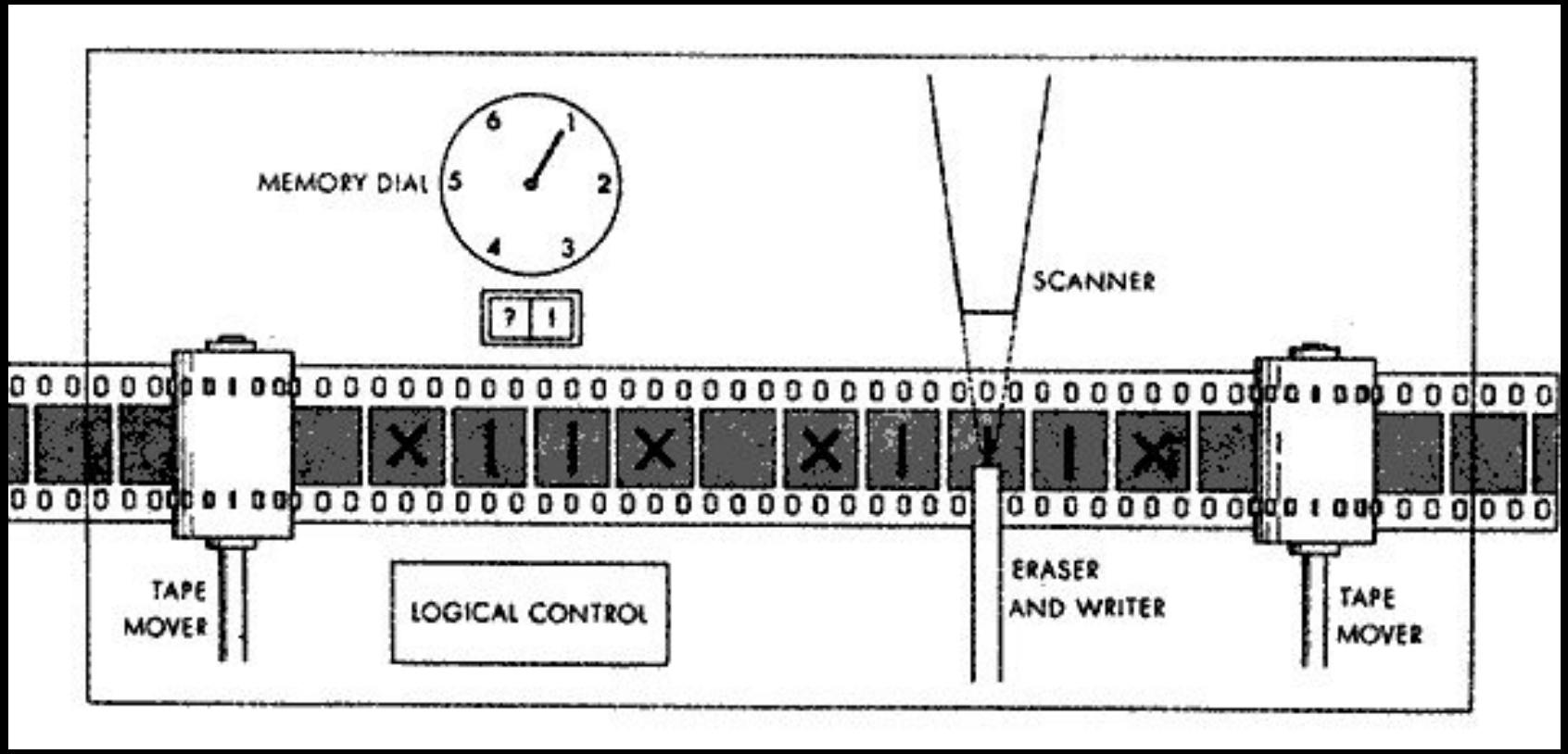
Why PDA is a more accurate model than FA?

What are the PDA restrictions?

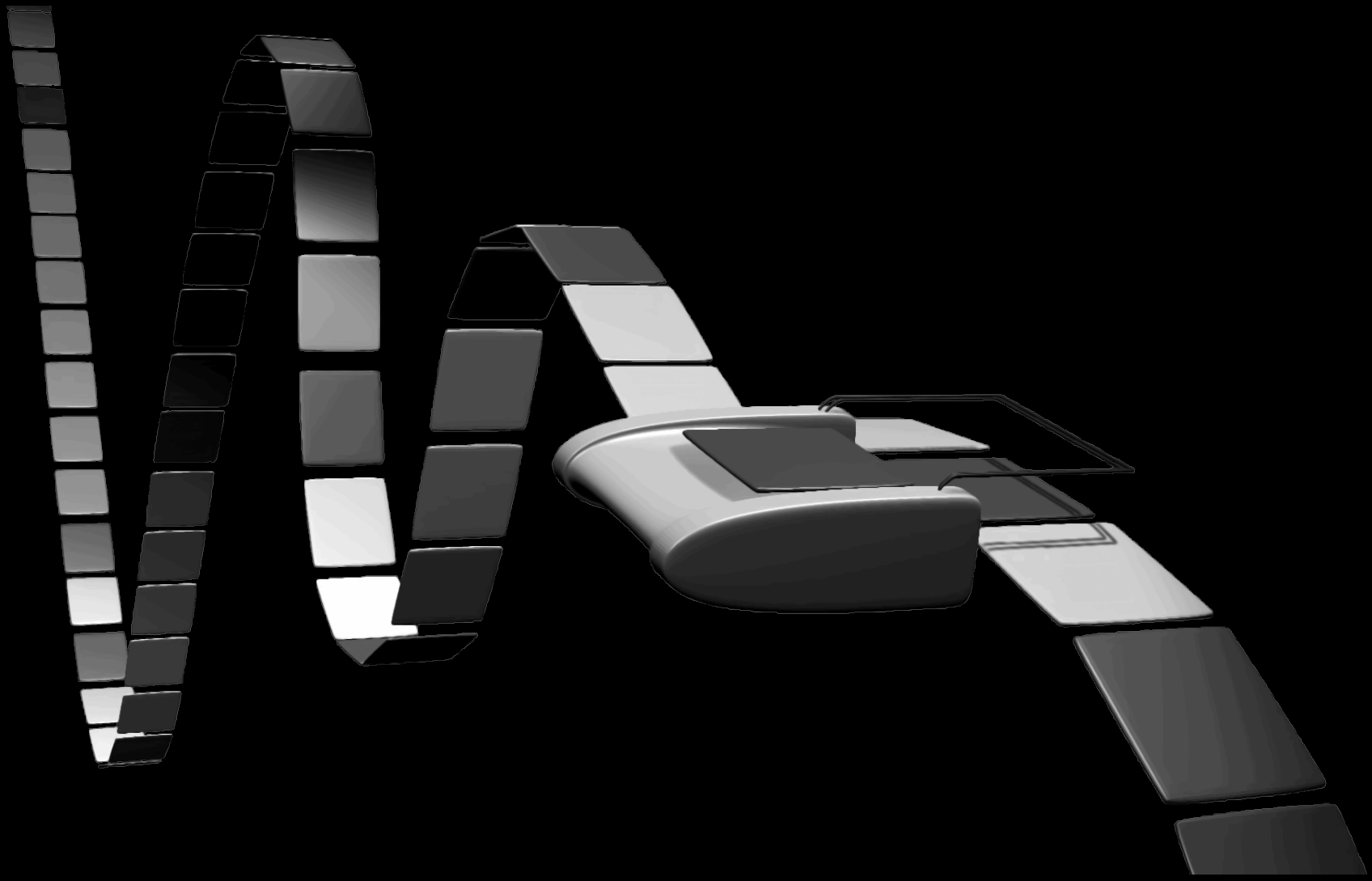
Turing Machine

- A Turing machine is a much more accurate model of a general purpose computer than FA or PDA
- A Turing machine can do everything that a real computer can do
- Even a Turing machine can not solve certain problems: in a very real sense, these problems are beyond the theoretical concepts of computation

Turing Machine



Turing Machine



What is different from FA?

- Input tape is infinite
- Input head can both READ and WRITE
- Input head can move LEFT and RIGHT
- Has both ACCEPT and REJECT states ***and accepts/rejects right away*** (does not need to reach the end of input)

Designing TM

- Let $L = \{ w\#w \mid w \in \{0,1\}^* \}$

Give a high-level description of a Turing Machine that accepts an input if it is in L, and rejects if not.

Let's try designing it all together