

# CSE 105

# THEORY OF COMPUTATION

---

"Winter" 2018

<http://cseweb.ucsd.edu/classes/wi18/cse105-ab/>

# Today's learning goals

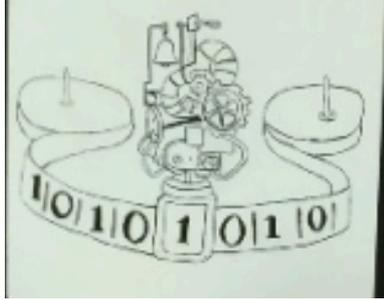
Sipser Section 3.2

- Describe several variants of Turing machines and informally explain why they are equally expressive.
- State and use the Church-Turing thesis.

# Describing TMs

*Sipser p. 184-185*

- **Formal definition:** set of states, input alphabet, tape alphabet, transition function, start state, accept state, reject state.
- **Implementation-level definition:** English prose to describe Turing machine head movements relative to contents of tape.
- **High-level description:** Description of algorithm, without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

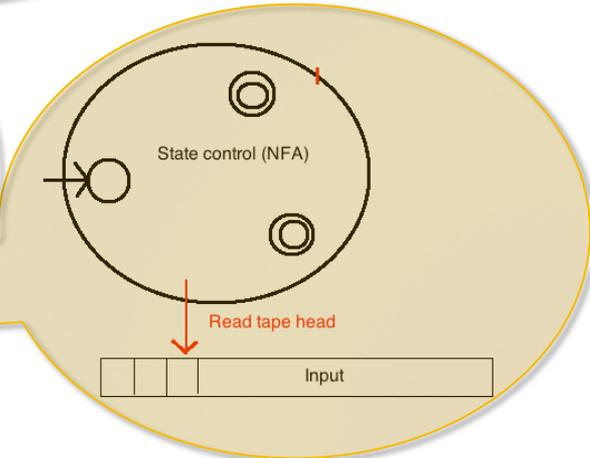
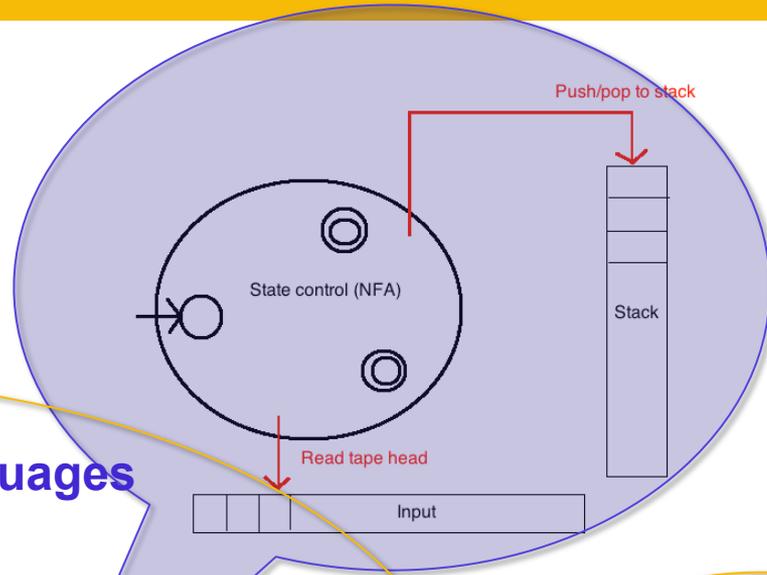


**Turing recognizable languages**

**Turing decidable languages**

**Context-free languages**

**Regular languages**



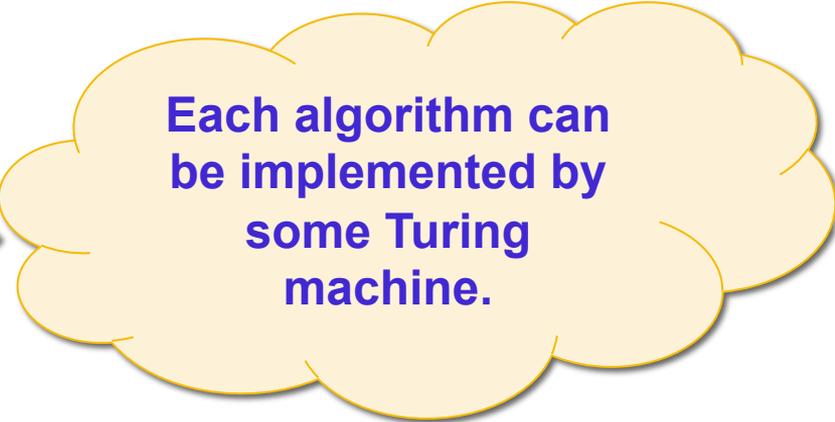
# High-level description = Algorithm

- **Wikipedia** "self-contained step-by-step set of operations to be performed"
- **CSE 20 textbook** "An algorithm is a finite sequence of precise instructions for performing a computation or for solving a problem."

**Church-Turing thesis**



**Each algorithm can  
be implemented by  
some Turing  
machine.**



# Variants of TMs

- Scratch work, copy input, ...
- Parallel computation
- Printing vs. accepting
- More flexible transition function
  - Can "stay put"
  - Can "get stuck"
  - Can "goto" cell on tape
  - *lots of examples in exercises*

**Multiple tapes**

**Nondeterminism**

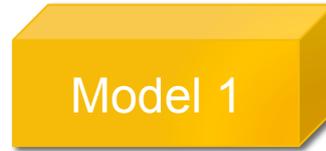
**Enumerators**

**All these models are  
equally expressive!**

**Also: wildly different models**

- **$\lambda$ -calculus, Post canonical systems, URMs, ...**

# "Equally expressive"



Model 1 is **equally expressive** as Model 2 iff

- every language recognized by some machine in Model 1 is recognizable by some machine in Model 2, **and**
- every language recognized by some machine in Model 2 is recognizable by some machine in Model 1.

**Which of the following statements is true?**

- A. NFAs and PDAs are equally expressive because they may both be nondeterministic.
- B. PDAs and Turing machines are equally expressive because they can both write (to a stack or the tape).
- C. NFAs and DFAs are equally expressive because they can be translated to one another.
- D. None of the above.

# Multitape TMs

Sipser p. 176

- As part of construction of machine, declare some finite number of tapes that are available.
- Input given on tape 1, rest of the tapes start blank.
- Each tape has its own read/write head.
- Transition function

$$Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$$

## Sketch of proof of equivalence:

*To simulate multiple tapes with one tape:* Use delimiter to keep tape contents separate, use special symbol to indicate location of each read/write head.

# Nondeterministic TMs

*Sipser p. 178*

At any point in the computation, machine may proceed according to several possibilities.

Transition function

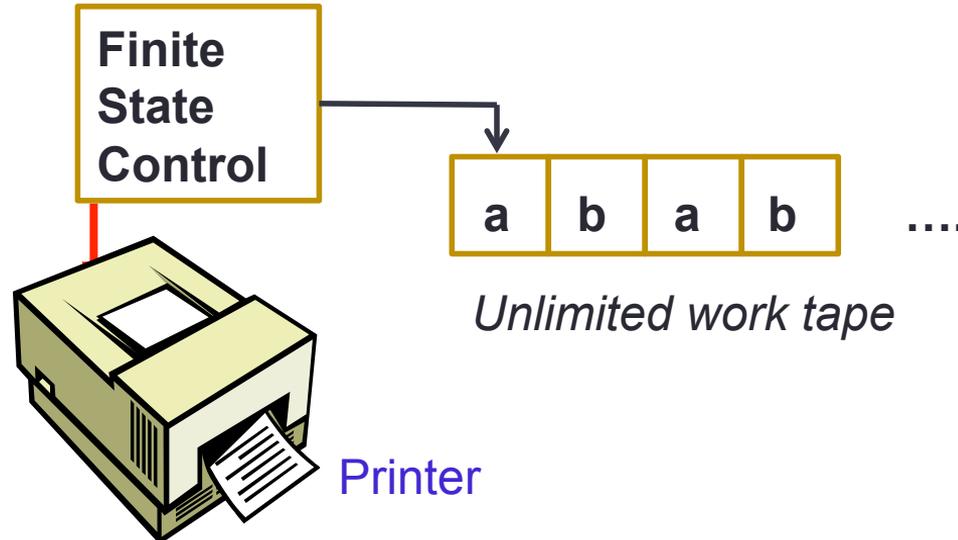
$$Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L,R\})$$

## **Sketch of proof of equivalence:**

*To simulate nondeterministic machine:* Use 3 tapes to do breadth-first search of computation tree: "read-only" input tape, simulation tape, tape tracking nondeterministic branching.

# Very different model: Enumerators Sipser p. 180

Produce language as output rather than recognize input



Computation proceeds according to transition function.

At any point, machine may "send" a string to printer.

$L(E) = \{ w \mid E \text{ eventually, in finite time, prints } w \}$

# Enumerators

Which of the following is a high level description for an enumerator that enumerates the set  $\{0\}$ ?

- A. "On input  $w$ , if  $w = 0$  accept, otherwise reject."
- B. "Ignore input. If  $w = 0$  accept, otherwise reject."
- C. "On input  $w$ , reject."
- D. "Ignore input. Print the string 0."
- E. None of the above.

# Recognition and enumeration Sipser Theorem 3.21

**Theorem:** A language  $L$  is Turing-recognizable iff some enumerator enumerates  $L$ .

Proof:

1. Assume  $L$  is Turing-recognizable. WTS some enumerator enumerates it.
2. Assume  $L$  is enumerated by some enumerator. WTS  $L$  is Turing-recognizable.

# Recognition and enumeration Sipser Theorem 3.21

**2. Assume the enumerator E enumerates L. WTS L is Turing-recognizable.**

We'll use E in a subroutine for high-level description of Turing machine M that will recognize L.

Define M as follows: M = "On input w,

1. Run E. For each string x printed by E
  - If  $x = w$ , accept. Otherwise, continue."

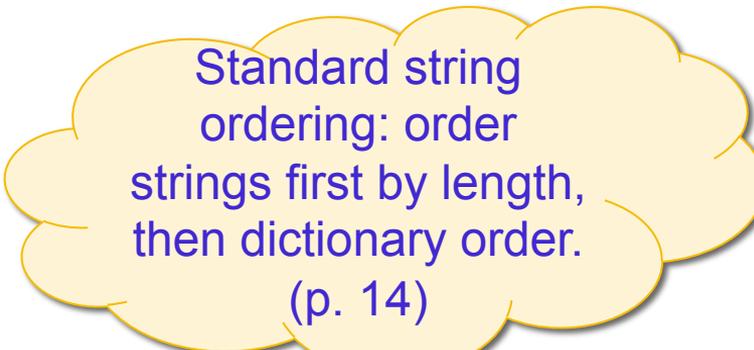
**Correctness?**

# Recognition and enumeration Sipser Theorem 3.21

**1. Assume  $L$  is Turing-recognizable. WTS some enumerator enumerates it.**

Let  $M$  be a TM that recognizes  $L$ . We'll use  $M$  in a subroutine for high-level description of enumerator  $E$ .

**Idea: check each string in turn to see if it is in  $L = L(M)$ .**



Standard string ordering: order strings first by length, then dictionary order.  
(p. 14)

# Recognition and enumeration Sipser Theorem 3.21

**1. Assume  $L$  is Turing-recognizable. WTS some enumerator enumerates it.**

Let  $M$  be a TM that recognizes  $L$ . We'll use  $M$  in a subroutine for high-level description of enumerator  $E$ .

Let  $s_1, s_2, \dots$  be a list of all strings in  $\Sigma^*$  in standard string order

$E =$  "Ignore any input. Repeat the following for  $i=1,2,3\dots$

1. Run  $M$  for  $i$  steps on each input  $s_1, \dots, s_i$
2. If any of the  $i$  computations of  $M$  accepts, print out the accepted string."

**Correctness?**

**Suppose M is TM  
that recognizes L**

**Suppose D is TM  
that decides L**

**Suppose E is  
enumerator that  
enumerates L**

If string  $w$  is in  $L$   
then ...

If string  $w$  is not in  
 $L$  then ...

# For next time

**GroupHW5 due Saturday, February 24**

For Friday, pre-class reading: pp. 185 (middle)