CSE 105
THEORY OF COMPUTATION

Spring 2018

http://cseweb.ucsd.edu/classes/sp18/cse105-ab/

Today's learning goals

Sipser Section 3.2

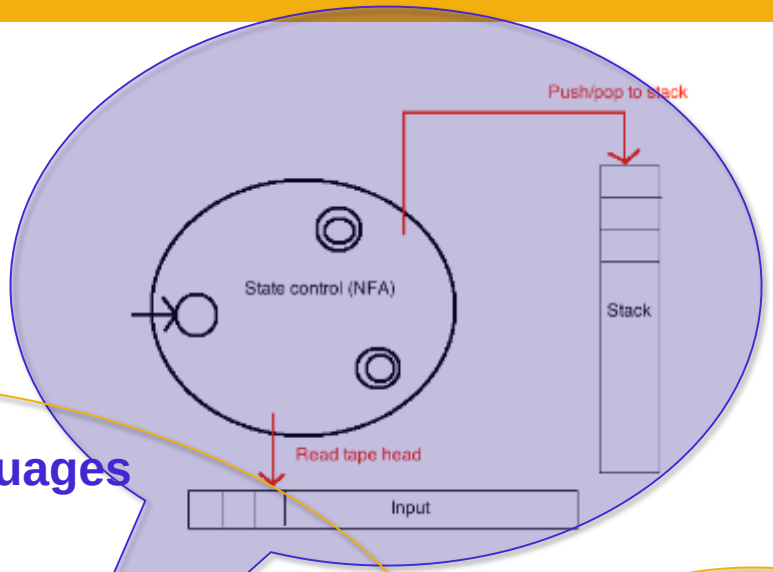
- Describe several variants of Turing machines and informally explain why they are equally expressive.
- State and use the Church-Turing thesis.

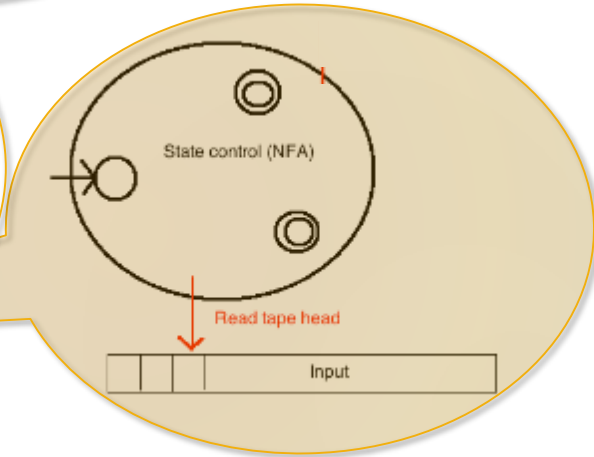
Turing recognizable languages

Turing decidable languages

Context-free languages

Regular languages





Why Turing machines …

- Why one-way infinite tape? JFLAP is different
- Why do we move the read-write head L/R at each step? Stay?
- Why no “jump to address”?
- Why deterministic? PDA, NFA have nondeterminism
- Why special q_{acc}, q_{rej} instead of F?
- Why recognize a language by checking inputs instead of producing it? CFG let us produce strings

Why Turing machines …

- Why one-way infinite tape?
- Why do we move the read-write head L/R at each step?
- Why no “jump to address”?
- Why deterministic?
- Why special q_{acc}, q_{rej} instead of F?
- Why recognize a language by checking inputs instead of producing it?

These are all conventions; can be changed without changing what it means for a language to be recognizable

Variants of TMs

- Scratch work, copy input, … **Multiple tapes**
- Parallel computation **Nondeterminism**
- Printing vs. accepting **Enumerators**
- More flexible transition function
  - Can "stay put"
  - Can "get stuck"
  - Can "goto" cell on tape
  - *lots of examples in exercises to Chapter 3*

All these models are equally expressive!

Also: wildly different models

- **λ-calculus, Post canonical systems, URMs, etc.**

"Equally expressive"

Model 1 is **equally expressive** as Model 2 iff every language recognized by a Model 1 machine is recognizable by a Model 2 machine **and** every language recognized by a Model 2 machine is recognizable by a Model 1 machine.

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.

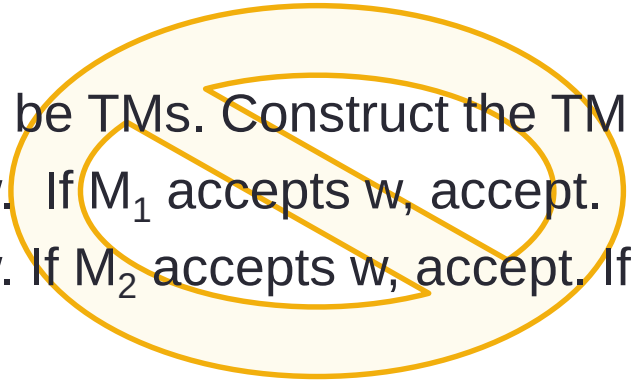
Why?

- Might be easier to build a machine to solve a problem if it can use additional features (e.g. different memory layout, nondeterminism).

- If these features are equally expressive to Turing machines, get easier proofs for closure of class of decidable / recognizable languages under certain operations

Example: union

Claim: The class of recognizable languages over fixed alphabet Σ is closed under union.



“Proof”: Let M_1 and M_2 be TMs. Construct the TM M = "On input w,

1. Run M_1 on input w. If M_1 accepts w, accept. If M_1 rejects, go to 2.
2. Run M_2 on input w. If M_2 accepts w, accept. If M_2 rejects, reject."

Problem: if M_i loops on w, can’t let it “hijack” M’s computation.

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.

Example: union

Claim: The class of recognizable languages over fixed alphabet Σ is closed under union.

Proof idea: Let M_1 and M_2 be TMs. Construct the two tape TM M = "On input w,

1. Copy w to second tape.
2. Simultaneously simulate the computations of M_1 and M_2 on w by using the two tapes.
3. If either computations halts and accept, accept."

Nondeterministic TMs

Sipser p. 178

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

Transition function

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

Nondeterministic machine accepts w iff there is a computation path that halts and accepts

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 braching.

Example: union

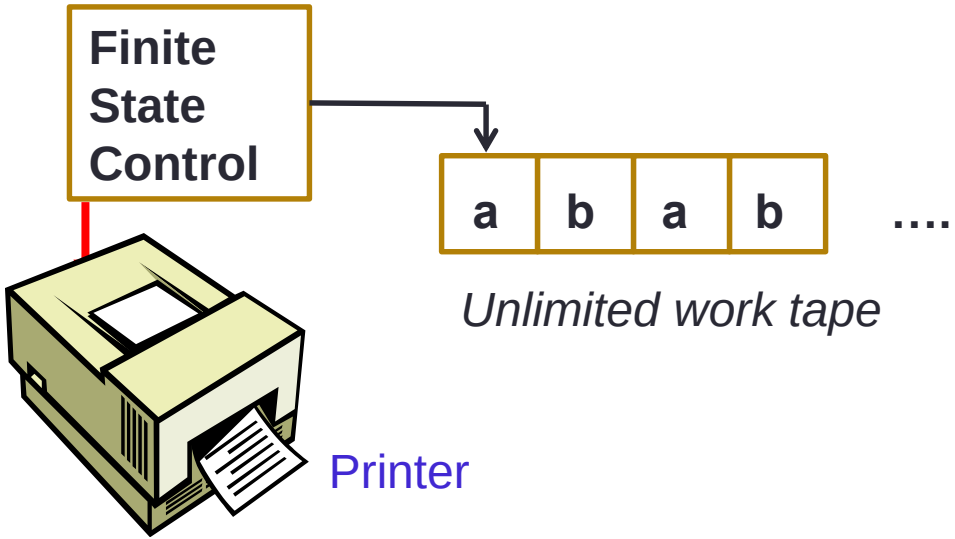
Claim: The class of recognizable languages over fixed alphabet Σ is closed under union.

Proof idea: Let M_1 and M_2 be TMs. Construct the nondeterministic TM
M = "On input w,

1. Nondeterministically choose M_i to be either M_1 or M_2
2. Run M_i on w. If it accepts, accept; if it rejects, reject."

Very different model: Enumerators *Sipser p. 180*

Produce language as output rather than recognize input

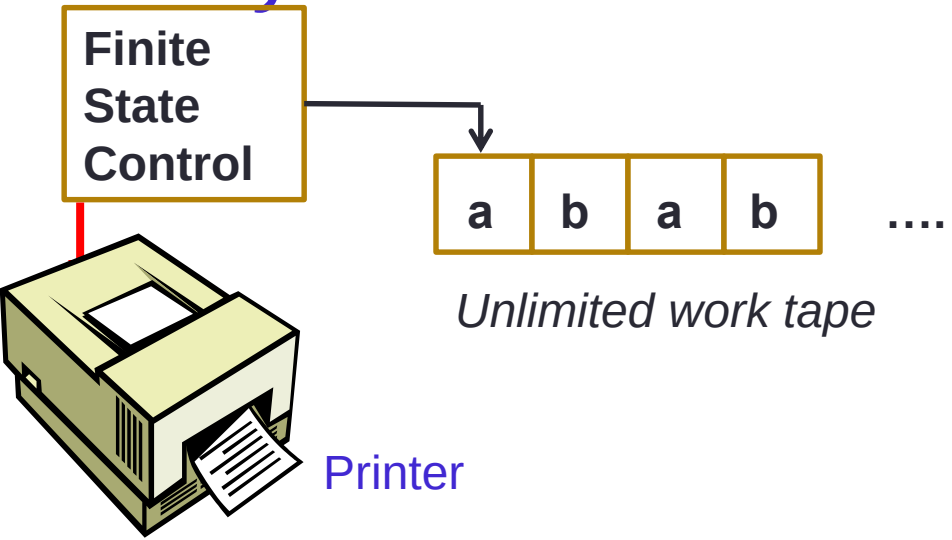


Computation proceeds according to transition function.

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

L(E) = { w | E eventually, in finite time, prints w}

Very different model: Enumerators *Sipser p. 180*



Computation starts with both tapes empty, apply δ over and over.

Whenever enter q_{print}, "print" the contents of the second tape, i.e. consider current word to be part of L(E).

Formally $(Q,\Sigma,\Gamma,\delta,q_0,q_{print})$ with

δ: $Q \times \Gamma \times \Gamma \rightarrow Q \times \Gamma \times \Gamma \times \{L,R\} \times \{L,R\}$

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, … be a list of all strings in Σ^* in standard string order

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

1. Run M for (at most) i steps on each input s_1, …, s_i
2. If any of the i computations of M accepts, print out the accepted string."

Correctness?

Example: union

Claim: The class of recognizable languages over fixed alphabet Σ is closed under union.

Proof idea: Let E_1 and E_2 be enumerators. Construct the enumerator E = ”<ignore input>:

1. Run E_1 and E_2 in parallel

   i.e. for i=1, 2, 3…

   1a. Run E_1 for i steps.

   1b. Run E_2 for i steps.

   Whenever any string is printed by either enumerator, print it.”

	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

Individual HW5 due Tuesday, May 15

GroupHW5 due Saturday, May 19

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