CSE 105
THEORY OF COMPUTATION

"Winter" 2018

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

Today's learning goals

Sipser Section 3.1

- Design TMs using different levels of descriptions.
- Determine whether a Turing machine is a decider.
- Prove properties of the classes of recognizable and decidable sets.

Announcements
- Group HW4 due Saturday
- Review Quiz "due" Sunday
- No class on Monday ← I'm still holding office hours
- Indiv HW5 due Tuesday
- Miles Jones subbing Wednesday + Friday

Describing TMs

Sipser p. 184-185

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

for loops, while,

Language of a TM

Sipser p. 170

L(M) = { w | M accepts w}

If w is in L(M) then the computation of M on w halts and accepts.

If the computation of M on w halts and rejects, then w is not in L(M).

If the computation of M on w doesn't halt, then w is not in L(M)

Deciders and recognizers *Sipser* p. 170 Defs 3.5 and 3.6

- L is **recognized** by Turing machine M if L(M) = ~~M~~ L.
- M is a **decider** if it is a Turing machine and halts on all inputs.
- L is **decided** by Turing machine M if M is a decider and L(M) = L.

{ deciders } $\subsetneq$ {TMs}

An example

Which of the following is an implementation-level description of a TM which **decides the empty set**? never loops

$L(M) = \emptyset$

M = "On input w:

A. reject." implementation + high level

~~B.~~ sweep right across the tape until find a non-blank symbol. Then, reject." not a decider $L(M) = \emptyset$

~~C.~~ If the first tape symbol is blank, accept. Otherwise, reject."

D. More than one of the above.

E. I don't know.

Extension

- Give an implementation-level description of a Turing machine which **recognizes** (but does not decide) the empty set.

- Give a high-level description of this Turing machine.

Another example

Suppose M_1 and M_2 are Turing machines

Consider the new TM M = "On input w,

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

What's L(M)?

$L(M_1) \cap L(M_2)$

Is M a decider?

if M_1, M_2 both are then yes

try 2nd machine for union

What kind of construction is this?

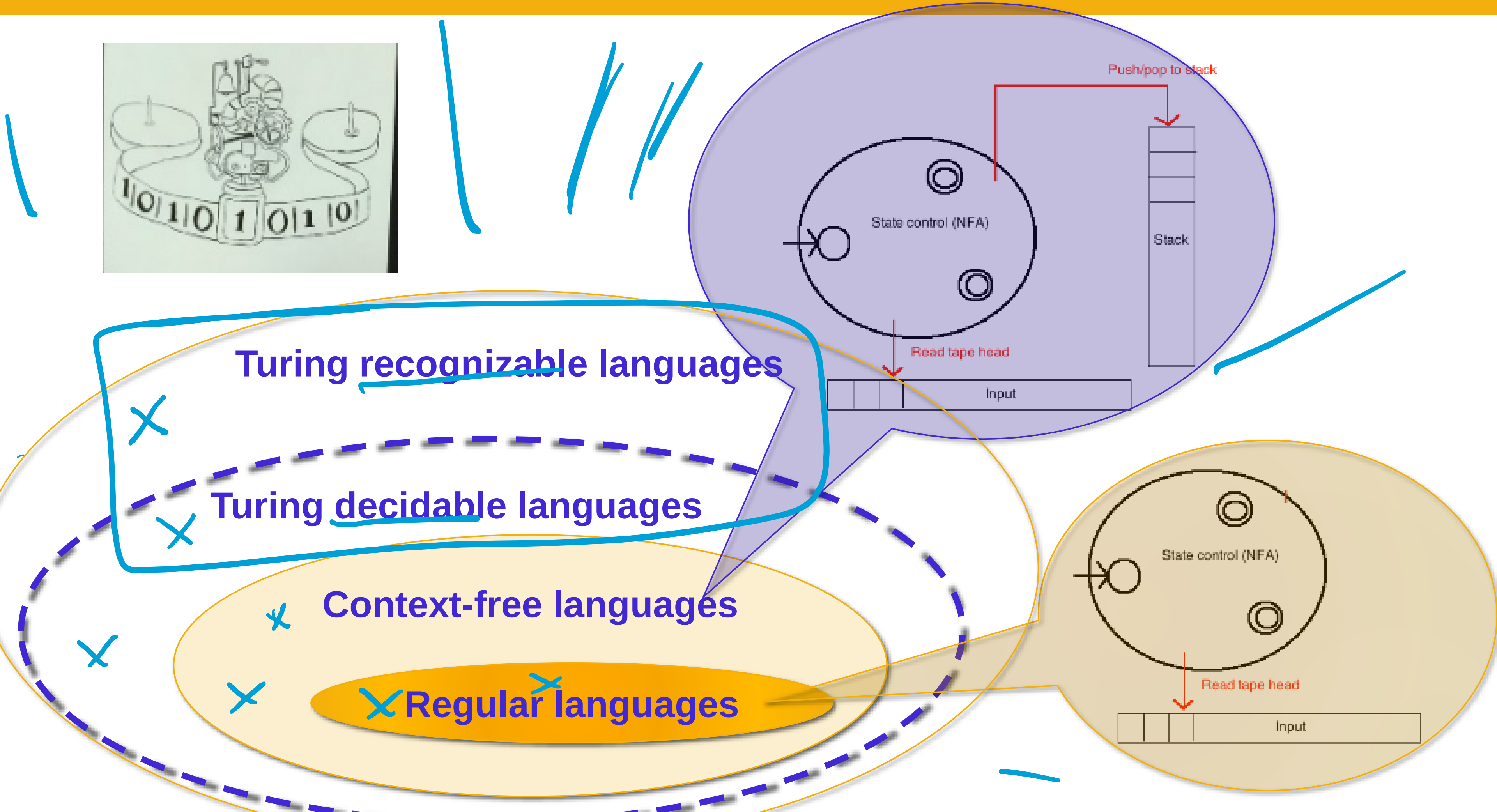
A. ~~Formal definition of TM~~
B. ~~Implementation-level description of TM~~
C. High-level description of TM
D. I don't know.

Classifying languages

A language L is

Turing-recognizable if there is a TM M such that L(M) = L
in other words, if there is some TM that recognizes it.

Turing-decidable if there is a TM M such that M is a decider and L(M) = L *in other words, if there is some TM that decides it.*


Push/pop to stack
State control (NFA)
Stack
Read tape head
Input
Turing recognizable languages
Turing decidable languages
Context-free languages
Regular languages
State control (NFA)
Read tape head
Input

Closure

$L_1 \rightarrow L_1 \cup L_2 \leftarrow L_2$

Theorem: The class of decidable languages over fixed alphabet Σ is closed under union.

Proof: Let …

WTS …

Closure

Theorem: The class of decidable languages over fixed alphabet Σ is closed under union.

Proof: Let L_1 and L_2 be languages over Σ and suppose M_1 and M_2 are TMs deciding these languages. We will **define** a new TM, M, via a high-level description. We will then show that **$L(M) = L_1 \cup L_2$ and that M always halts.**

Conclude: $L_1 \cup L_2$ is decidable

Closure

Theorem: The class of decidable languages over fixed alphabet Σ is closed under union.

Proof: Let L_1 and L_2 be languages and suppose M_1 and M_2 are TMs deciding these languages. Construct the TM M as "On input w,

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

Correctness of construction:

WTS $L(M) = L_1 \cup L_2$ and M is a decider.

1 2

Where do we use decidability?

Pf For an arbitrary string w

WTS ① Assume $w \in L_1 \cup L_2$. WTS M accepts w

Run M on w. That is, in step 1 we run M_1 on w. Computation halts b/c M_1 is decider

If $w \in L_1$, then b/c $L_1 = L(M_1)$, M_1 accepts w so step 1 says M also accepts :) Otherwise $w \notin L_1$, so $w \in L_2$ (b/c case is $w \in L_1 \cup L_2$) so M_1 rejects w, M moves to steps 2, runs M_2 on w and accepts :)

WTS ② Assume $w \notin L_1 \cup L_2$ WTS M rejects w

(yourself)

Closure

Good exercises – can't use without proof! (Sipser 3.15, 3.16)

The class of decidable languages is closed under	The class of recognizable languages is closed under
• Union ✓ • Concatenation • Intersection • Kleene star • Complementation	• Union • Concatenation • Intersection • Kleene star

For next time

GroupHW4 due Saturday, February 17

For Wednesday, pre-class reading: Section 3.2, pp. 181