18.404/6.840 Lecture 11

Last time:

- The Computation History Method for proving undecidability
- The Post Correspondence Problem is undecidable
- Linearly bounded automata, is decidable
- Configurations, Computation histories
- and are undecidable

Today:

- Self-reproducing machines and The Recursion theorem
- Short introduction to mathematical logic

No class Tuesday, October 13 (Monday schedule due to Indigenous Peoples' Day)
I will hold my office hours (4:00-5:30) on October 13.

Midterm exam Thursday, October 15.
Details next slide.

Midterm exam – October 15

Midterm exam Thursday, October 15, 2020.

90 minutes length + 20 minutes for printing/scanning/uploading.
Download and upload via Gradescope.
Flexible start time.

Sample problems and solutions have been posted.
Open book, postings, piazza, notes, and lecture videos, from this year.
Covers through Recursion Theorem presented today.
Will not include section on mathematical logic.

Not permitted: Communication with anyone except course staff, other materials, internet searching.

Not permitted: Providing information about the exam to anyone who hasn't completed it.

Please respect our honor system.

Self-reproduction Paradox

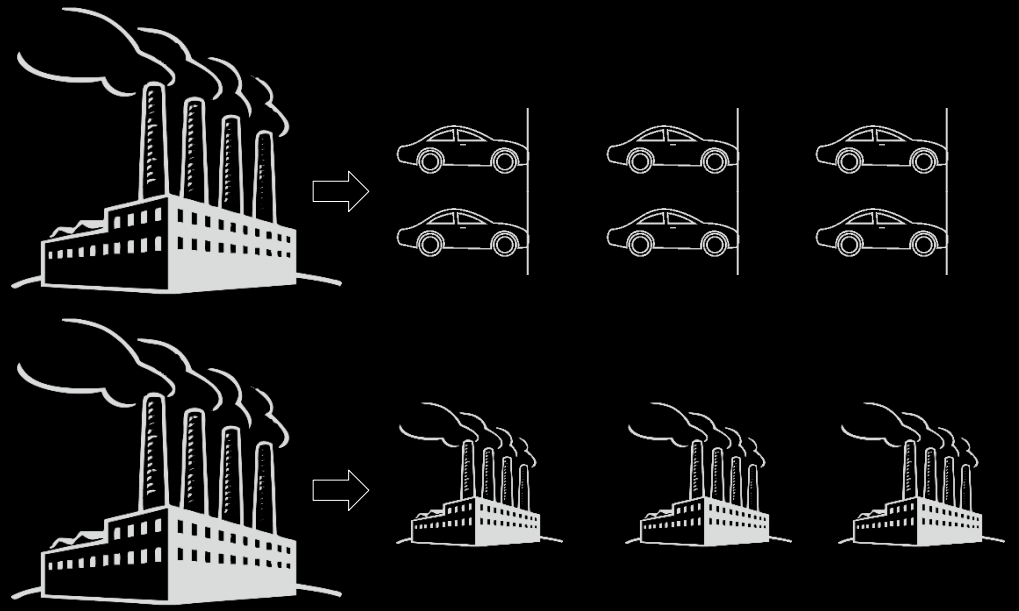
Suppose a Factory makes Cars
- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

Can a Factory make Factories?
- Complexity of Factory > Complexity of Factory?
- Seems impossible to have a self-reproducing machine

But, living things self-reproduce

How to resolve this paradox?

Self-reproducing machines are possible!



A Self-Reproducing TM

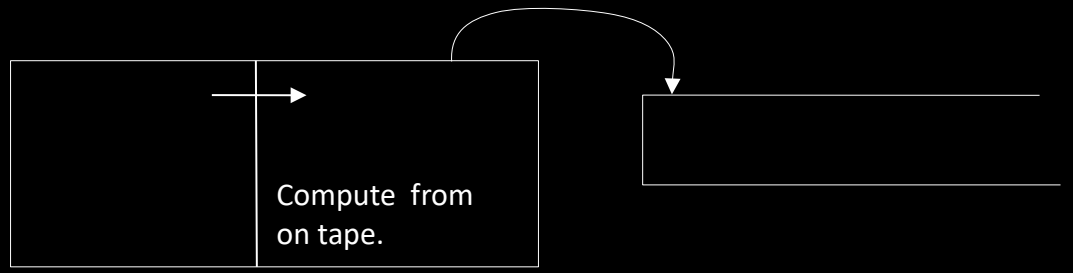
Theorem: There is a TM which (on any input) halts with on the tape.

Lemma: There is a computable function such that for every , where is the TM "Print on the tape and halt".
Proof: Straightforward.

Proof of Theorem: has two parts, and .

? NO, wou

"1. Compute (tape contents) to get .
2. Combine with to get .
3. Halt with on tape."



Can implement in any programming language.

English Implementation

Check-in 11.1

Implementations of the Recursion Theorem have two parts, a Template and an Action. In the TM and English implementations, which is the Action part?

(a) A and the upper phrase

(b) A and the lower phrase

(c) B and the upper phrase

(d) B and the lower phrase.

Write the following twice, the second time in quotes

"Write the following twice, the second time in quotes"

Write the following twice, the second time in quotes

"Write the following twice, the second time in quotes"

Note on Pset Problem 6: Don't need to worry about quoting.

Check-in 11.1

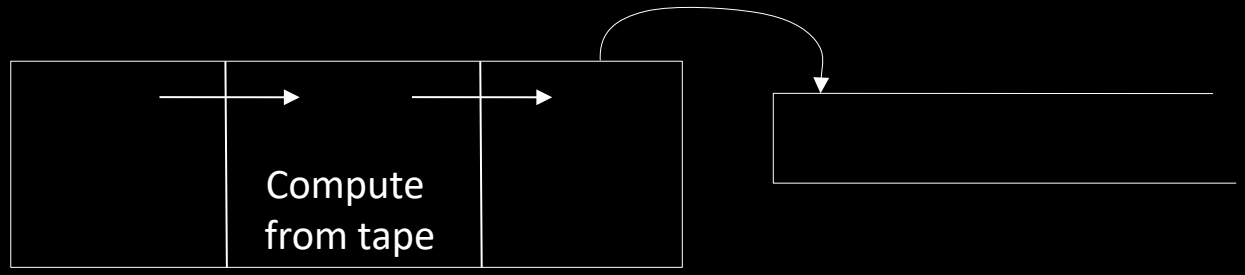
The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM there is a TM where for all
R on input operates in the same way as on input .

Proof of Theorem: has three parts: , , and .
is given

"1. Compute (tape contents after) to get .
2. Combine with to get .
3. Pass control to on input ."



Moral: You can use "compute your own description" in describing TMs.

Check-in 11.2

Can we use the Recursion Theorem to design a TM where ?

(a) Yes.

(b) No.

Ex 1: is undecidable - new proof

Theorem: is not decidable

Proof by contradiction: Assume some TM decides .

Consider the following TM :

"On input

1. Get own description .
2. Use on input to determine whether accepts .
3. Do the opposite of what says."


Coffee
Break

Ex 2: Fixed-point Theorem

Theorem: For any computable function ,
there is a TM such that where .

In other words, consider to be a program transformation function.
Then for some program , its behavior is unchanged by .

Proof: Let be the following TM.

"On input

1. Get own description .
2. Compute and call the result .
3. Simulate on ."

Ex 3: is T-unrecognizable

Defn: is a minimal TM if .
Thus, a minimal TM has the shortest description among all equivalent TMs.

Let is a minimal TM.
Theorem: is T-unrecognizable.
Proof by contradiction: Assume some TM enumerates .

Consider the following TM :
"On input
1. Get own description .
2. Run enumerator until some TM appears, where .
3. Simulate on ."

Thus and so isn't minimal, but , contradiction.

Check-in 11.3

Let be an infinite subset of .
Is it possible that is T-recognizable?

(a) Yes.

(b) No.

Check-in 11.3

Other applications

1. Computer viruses.
2. A true but unprovable mathematical statement due to Kurt Gödel:
   “This statement is unprovable.”

Intro to Mathematical Logic

Goal: A mathematical study of mathematical reasoning itself.
Formally defines the language of mathematics, mathematical truth, and provability.

Gödel's First Incompleteness Theorem:
In any reasonable formal system, some true statements are not provable.

Proof: We use two properties of formal proofs:
1) Soundness: If has a proof then is true.
2) Checkability: The language is a proof of statement is decidable.

Checkability implies the set of provable statements $\{\langle\phi\rangle \mid \phi$ has a proof$\}$ is T-recognizable.

SImilarly, if we can always prove when it is true, then is T-recognizable (false!).

Therefore, some true statements of the form are unprovable.

Next, we use the Recursion Theorem to give a specific example of a true but unprovable statement.

A True but Unprovable Statement

Implement Gödel statement "This statement is unprovable."
Let be the statement where is the following TM:

"On any input

1. Obtain and use it to obtain .
2. For each possible proof
   Test if is a proof that is true.
   If yes, then *accept*. Otherwise, continue."

Theorem: (1) has no proof
(2) is true

Proof:

(1) If has a proof TM

(2) If is false R

Quick review of today

1. Self-reference and The Recursion Theorem
2. Various applications.
3. Sketch of Gödel's First Incompleteness Theorem in mathematical logic.