

CSE 105—Theory of Computability

Lecture 14—May 18, 2006
Turing Machines
Undecidability

Instructor: Neil Rhodes

Recognizing vs. Deciding

L is Turing-recognizable (recursively enumerable if)

- There exists a TM, M where every string s in L
 - is accepted by M

L is Turing-decidable (recursive) if

- There exists a TM, M where, for every string s:
 - If S in L, M accepts L
 - If S not in L, M rejects L
- That is, M (eventually) halts on all inputs

Complement of Turing-Decidable languages

If a language is Turing-decidable (recursive), its complement is:

If a language and its complement are both recursively enumerable, they are both:

Closure properties of Turing-decidable languages

Union

Intersection

Concatenation

Reverse

Closure properties of Turing-recognizable languages

Union

Intersection

Concatenation

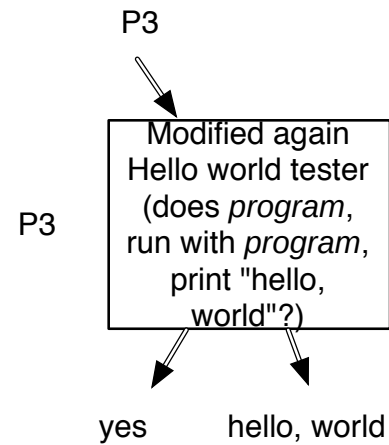
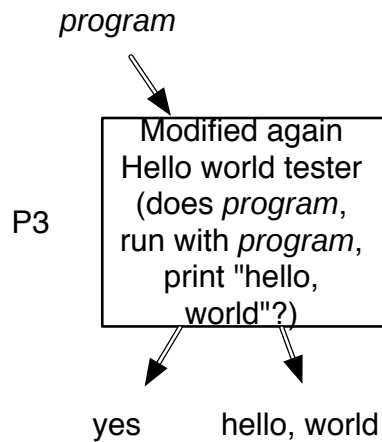
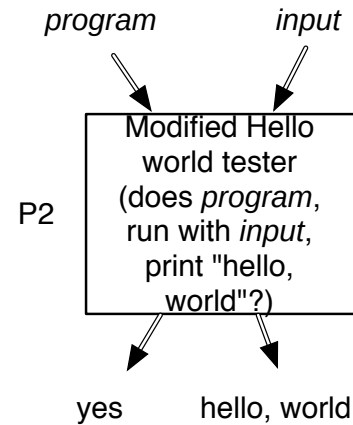
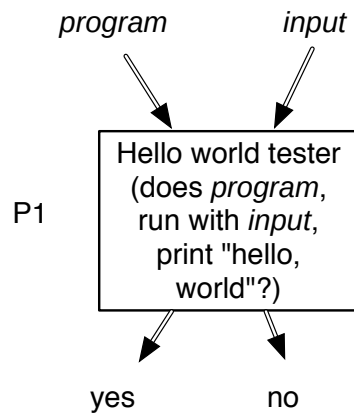
Reverse

Barber paradox

The only barber in town shaves everyone who doesn't shave themselves.

- Who shaves the barber?

Impossible-to-write program



Diagonalization

The set of real numbers is uncountable (no bijection with the natural numbers)

There are many languages which have no TM
(not recursively enumerable)

How large is set of TMs?

How large is set of languages?

Bijection possible?

A language that isn't recursively enumerable

Encode Turing machines as integers

- Any integer describing an invalid Turing machine we'll interpret as a TM that halts and rejects
- TM_i = Turing Machine described by integer i

Make table

- Columns are i th binary string
- Rows are j th Turing machine
- Each entry (i, j) is:
 - 1 if TM_j accepts input i
 - 0 if TM_j doesn't accept input i

L_D = opposite of (i, i) for each i

- Diagonalization language not recognized by *any* TM

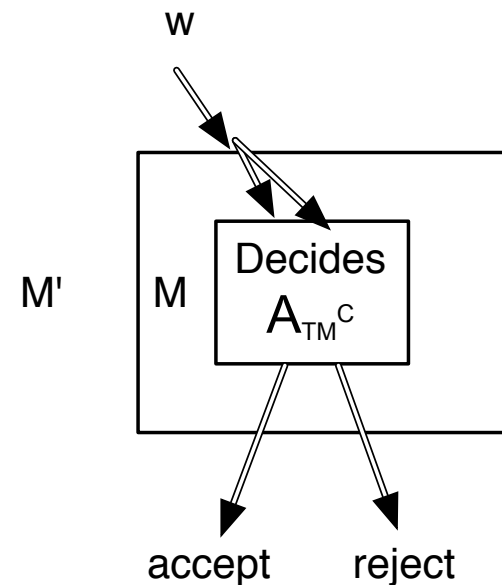
	1	2	3	4	5	6	7	8
TM_1								
TM_2								
TM_3								
TM_4								
TM_5								
TM_6								
TM_7								
TM_8								

Universal Turing Machine language A_{TM} is undecidable

$ATM = \{(M, w) \mid M \text{ is a TM and } M \text{ accepts } w\}$

Assume recursive (decidable)

- Therefore, A_{TM}^C is recursive
- There exists machine M which decides A_{TM}^C



Reductions

We have an algorithm that converts instances of problem P_1 to instances of problem P_2 with the same answer

- We have reduced P_1 to P_2
- P_2 is at least as hard as P_1
 - Because if we have an algorithm for P_2 , we have an algorithm for P_1

Start with a known hard problem P_1 for which no machine exists

- Like, does M on w accept?

Assume there's a TM, M_2 , that answers some other question P_2

- Does M accept the empty language?

Show a way to create a Turing Machine, M' that:

- Takes the inputs for P_1 .
- Converts them into inputs for M_2 .
- Takes the answer from M_2 and computes answer for M'
- But, since M' can't exist, M_2 can't exist either.

Reducing M accepts W to halting problem

Halting problem:

- Given M on w, does M halt (accept or reject)?
- $\text{HALT}_{\text{TM}} = \{(M, w) \mid M \text{ is a TM and } M \text{ halts on input } w\}$

Assume there exists machine M_{Halt} that solves the halting problem

Create M'

- Given M and w,
- Call M_{Halt} on M with w
 - If rejects, reject
 - If accepts, simulate M on w and when it halts, accept or reject appropriately

M' decides $A_{\text{TM}} = \{(M, w) \mid M \text{ is a TM and } M \text{ accepts input } w\}$

- But this is undecidable!
- So, M_{Halt} doesn't exist
- Therefore, Halting problem is undecidable

$\text{EMPTY}_{\text{TM}} = \{M \mid M \text{ is a TM and } M \text{ accepts the empty language}\}$

Is EMPTY_{TM} decidable?

Is $\text{EMPTY}_{\text{TM}}^c$ decidable?

$\text{REGULAR}_{\text{TM}} = \{M \mid M \text{ is a TM and } M \text{ accepts a regular language}\}$

Is $\text{REGULAR}_{\text{TM}}$ decidable?

Is $\text{REGULAR}_{\text{TM}}^c$ decidable?