1) "L" LANGUAGE; http://kilby.stanford.edu/~rvg/154/handouts/decidability.html (Theorem 1)

Decidable and Recognizable Languages

Recall: Definition

A Turing machine M is said to recognize a language L if $L = L(M)$.
A Turing machine M is said to decide a language L if $L = L(M)$ and M halts on every input.

L is said to be Turing-recognizable (or simply recognizable) if there exists a TM M which recognizes L. L is said to be Turing-decidable (or simply decidable) if there exists a TM M which decides L.

- Every finite language is decidable: For e.g., by a TM that has all the strings in the language "hard-coded" into it
- We just saw some example algorithms all of which terminate in a finite number of steps, and output yes or no (accept or reject). i.e., They decide the corresponding languages.

Decidable and Recognizable Languages

- But not all languages are decidable! In the next class we will see an example:
  - $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ is undecidable
- However A_{TM} is Turing-recognizable!

Proposition

There are languages which are recognizable, but not decidable

Recognizing A_{TM}

Program U for recognizing A_{TM}:

```
On input ⟨M, w⟩
    simulate M on w
    if simulated M accepts w, then accept
    else reject (by moving to q_rej)
```

U (the Universal TM) accepts $\langle M, w\rangle$ iff M accepts w. i.e.,

$$L(U) = A_{\text{TM}}$$

But U does not decide A_{TM}: If M rejects w by not halting, U rejects $\langle M, w\rangle$ by not halting. Indeed (as we shall see) no TM decides A_{TM}.

Deciding vs. Recognizing

Proposition

If L and $\overline{L}$ are recognizable, then L is decidable

Proof.

Program P for deciding L, given programs P_L and $P_{\overline{L}}$ for recognizing L and $\overline{L}$:

- On input x, simulate P_L and $P_{\overline{L}}$ on input x. Whether $x \in L$ or $x \notin L$, one of P_L and $P_{\overline{L}}$ will halt in finite number of steps.
- Which one to simulate first? Either could go on forever.
- On input x, simulate in parallel P_L and $P_{\overline{L}}$ on input x until either P_L or $P_{\overline{L}}$ accepts
- If P_L accepts, accept x and halt. If $P_{\overline{L}}$ accepts, reject x and halt.

Deciding vs. Recognizing

Proof (contd).

In more detail, P works as follows:

```
On input x
for i = 1,2,3,...
    simulate P_L on input x for i steps
    simulate P_L̄ on input x for i steps
    if either simulation accepts, break
if P_L accepted, accept x (and halt)
if P_L̄ accepted, reject x (and halt)
```

(Alternately, maintain configurations of P_L and $P_{\overline{L}}$, and in each iteration of the loop advance both their simulations by one step.) □

Deciding vs. Recognizing

So far:

- A_{TM} is undecidable (next lecture)
- But it is recognizable
- Is every language recognizable? No!

Proposition

$\overline{A_{\mathrm{TM}}}$ *is unrecognizable*

Proof.

If $\overline{A_{\mathrm{TM}}}$ is recognizable, since A_{TM} is recognizable, the two languages will be decidable too! □

Note: Decidable languages are closed under complementation, but recognizable languages are not.

Recursive Enumerability

- A Turing Machine on an input w either (halts and) accepts, or (halts and) rejects, or never halts.
- The language of a Turing Machine M, denoted as $L(M)$, is the set of all strings w on which M accepts.
- A language L is recursively enumerable/Turing recognizable if there is a Turing Machine M such that $L(M) = L$.

Decidability

- A language L is decidable if there is a Turing machine M such that $L(M) = L$ and M halts on every input.
- Thus, if L is decidable then L is recursively enumerable.

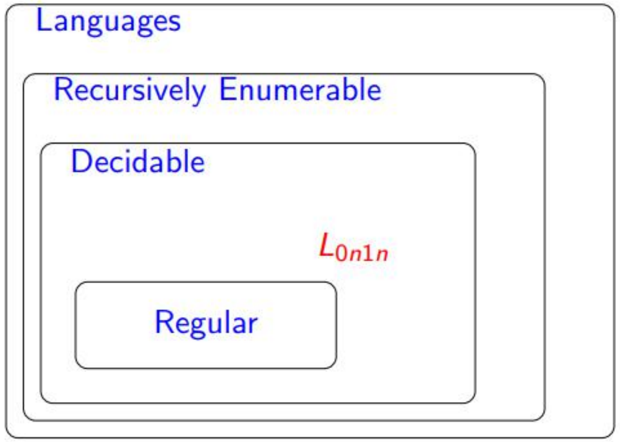
Undecidability

Definition

A language L is undecidable if L is not decidable. Thus, there is no Turing machine M that halts on every input and $L(M) = L$.

- This means that either L is not recursively enumerable. That is there is no turing machine M such that $L(M) = L$, or
- L is recursively enumerable but not decidable. That is, any Turing machine M such that $L(M) = L$, M does not halt on some inputs.

Big Picture



Relationship between classes of Languages

2) "Ld" LANGUAGE; https://courses.engr.illinois.edu/cs373/fa2012/Lectures/lec22.pdf **(Proposition 3)**

The Diagonal Language

Definition

Define $L_d = \{M \mid M \notin L(M)\}$. Thus, L_d is the collection of Turing machines (programs) M such that M does not halt and accept when given itself as input.

A non-Recursively Enumerable Language

Proposition

L_d is not recursively enumerable.

Proof.

Recall that,

- Inputs are strings over $\{0, 1\}$
- Every Turing Machine can be described by a binary string and every binary string can be viewed as Turing Machine
- In what follows, we will denote the ith binary string (in lexicographic order) as the number i. Thus, we can say $j \in L(i)$, which means that the Turing machine corresponding to ith binary string accepts the jth binary string.

Completing the proof

Diagonalization: Cantor

Proof (contd).

We can organize all programs and inputs as a (infinite) matrix, where the (i,j)th entry is Y if and only if $j \in L(i)$.

Inputs $\longrightarrow$

TMs ↓	1	2	3	4	5	6	7	⋯
1	$\boxed{\text{N}}$	N	N	N	N	N	N	
2	N	$\boxed{\text{N}}$	N	N	N	N	N	
3	Y	N	$\boxed{\text{Y}}$	N	Y	Y	Y	
4	N	Y	N	$\boxed{\text{Y}}$	Y	N	N	
5	N	Y	N	Y	$\boxed{\text{Y}}$	N	N	
6	N	N	Y	N	Y	$\boxed{\text{N}}$	Y	

Suppose L_d is recognized by a Turing machine, which is the jth binary string. i.e., $L_d = L(j)$. But $j \in L_d$ iff $j \notin L(j)$! □

Acceptor for L_d?

Consider the following program

```
On input i
    Run program i on i
    Output ''yes'' if i does not accept i
    Output ''no'' if i accepts i
```

Does the above program recognize L_d? No, because it may never output "yes" if i does not halt on i.

Recursively Enumerable but not Decidable

- L_d not recursively enumerable, and therefore not decidable. Are there languages that are recursively enumerable but not decidable?
- Yes, $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$

3) "Atm" LANGUAGE; https://courses.engr.illinois.edu/cs373/fa2012/Lectures/lec22.pdf **(Proposition 4)**

The Universal Language

Proposition

A_{TM} is r.e. but not decidable.

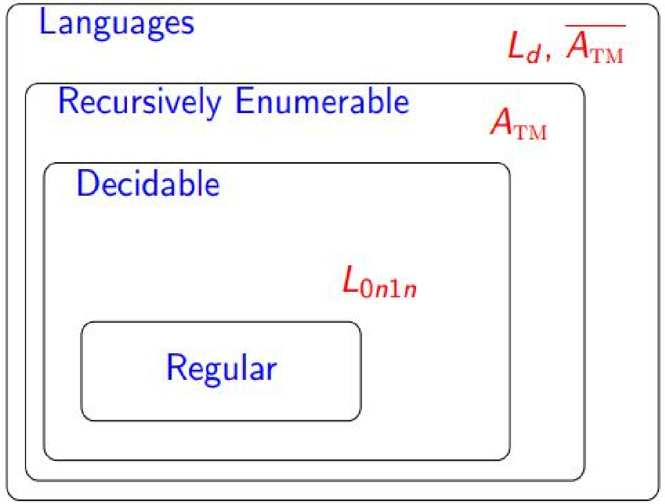
Proof.

We have already seen that A_{TM} is r.e. Suppose (for contradiction) A_{TM} is decidable. Then there is a TM M that always halts and $L(M) = A_{\mathrm{TM}}$. Consider a TM D as follows:

```
On input i
    Run M on input ⟨i, i⟩
    Output ``yes'' if i rejects i
    Output ``no'' if i accepts i
```

Observe that $L(D) = L_d$! But, L_d is not r.e. which gives us the contradiction. □

A more complete Big Picture



4) What is the 'Halting problem'?; http://bilgisayarkavramlari.sadievrenseker.com/2008/11/12/durma-problemi-halting-problem/

5) Let L be an undecidable language. Is it possible that both L and L¯ (the complement of L) are recognizable. If yes, give an example. If no, give a short proof?;

http://fuuu.be/polytech/INFOF408/Introduction-To-The-Theory-Of-Computation-Michael-Sipser.pdf **(Page 202 - Theorem 4.22)**