CSE322 Theory of Computation (L18)

L is recognizable if there is a DTM that recognizes L.

if M decides L, M also recognizes L.

DTM accepts x if DTM goes to accept state on input x

if L is decidable then L is recog.

DTM halts

DTM rejects x if DTM goes to reject state on input x

DTM recognizes L if DTM accepts (goes to qaccept) x iff x is in L   L is recognizable

=> for strings not in L, may reject or may loop

DTM decides L if DTM accepts x if x is in L and rejects (goes to qrej) x if x is not in L

=> DTM never loops   L is decidable

L is recognizable if there is a DTM that recognizes L.

Q: L is (DTM)-recognizable iff L is NDTM-recognizable?

Q: L is (DTM)-decidable iff L is NDTM-decidable?

NDTM accepts x if some non-deterministic choice leads to accept state on input x

NDTM recognizes L if NDTM(x) goes to qaccept for some choice iff x is in L

=> for strings not in L, may reject or may loop on every choice

NDTM decides L if ...

NDTM(x) halts on all choices

for x in L, NDTM(x) goes to qacc for some choice

for x not in L, NDTM(x) goes to qrej for all choices ⟷ if N goes to qacc for some choice then $x \in L$.

If L is decided (recognized) by a DTM
then L is decided (recognized) by an NDTM.

If L is recognized by an NDTM, there L is recognized by a DTM.

Proof by construction: Let L be recognized by NDTM N. We will construct DTM D to recognize L.

$x \in L$ iff N(x) goes to qacc on some nondet. branch.

Use D = DTM simulator done in class for N. Now, we will prove that D recognizes L, i.e., D accepts only strings in L and all strings in L.

(a) Show that if x in L (i.e., N accepts x), then D accepts x.

(b) Show that if D accepts x, then x in L (i.e., L accepts x).

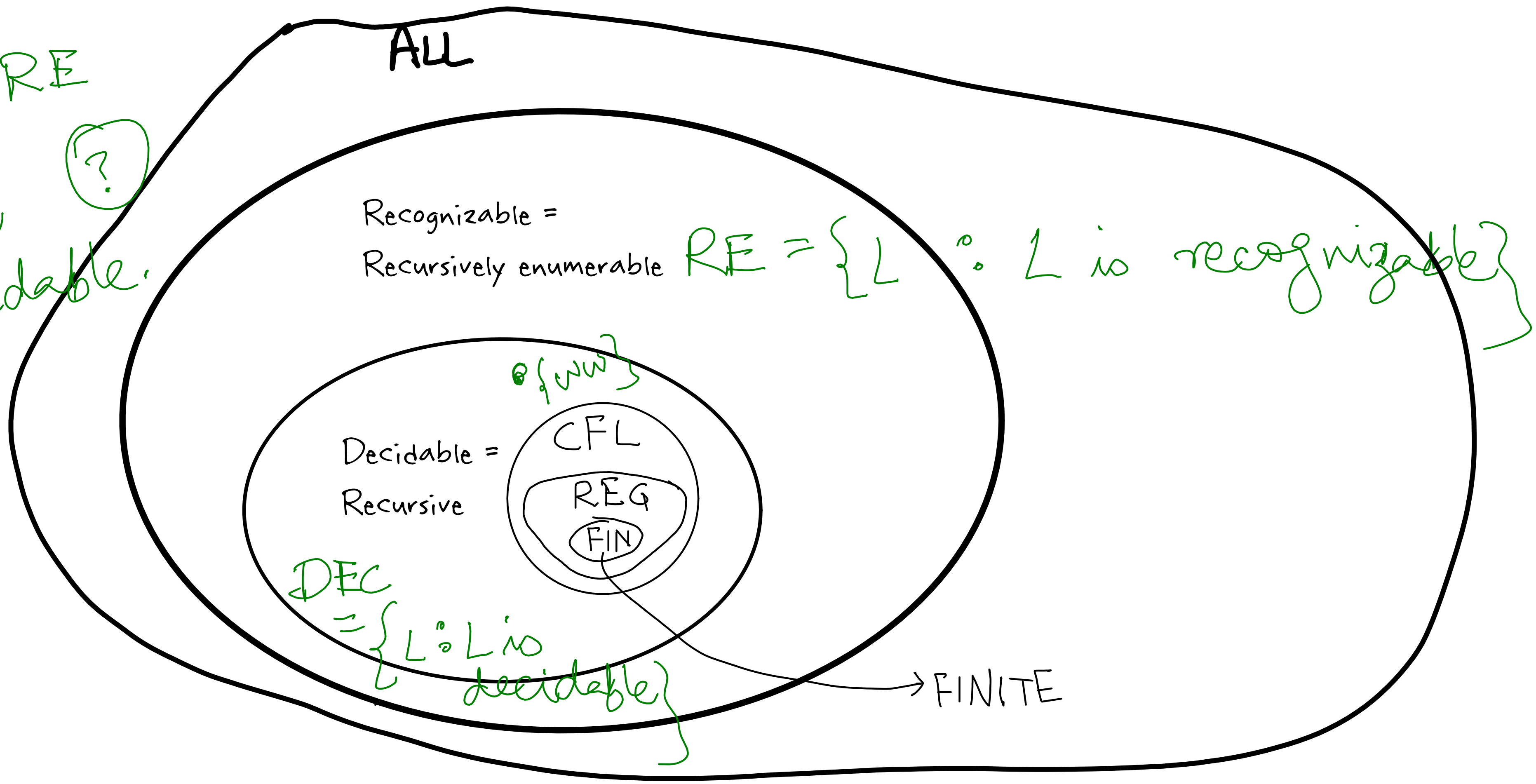
Both (a) and (b) follow from the property of the DTM simulator.

If L is decided by an NDTM, there L is decided by a DTM.

Similar proof as above. (Exercise)

DEC $\subseteq_?$ RE

① if L is a CFL, then L is decidable. ?



L is recognizable if there is a DTM or NDTM whose language is L.

L is decidable if there is a DTM that always halts and whose language is L.

Lemma: L is decidable if there is an NDTM that always halts on any non-deterministic branch and whose language is L.

TM for $L = \{ a^i b a^j : 0 <= i < j \}$

$\cancel{a}\,\cancel{a}\,a\,a\,a \ldots \quad b \quad \cancel{a}\,\cancel{a}\,a\,a\,a \quad a$

$\cancel{a}\,\cancel{a}\,a\,b\,\cancel{a}\,a\,b\,b\,b \rightarrow$

0. Right-shift the input

1. Make a left-to-right pass and check that input string is of the form a^*baa^*. If not, reject. Otherwise, put head back at left end.

2. Move right and put head on first non-blank symbol.

3. if input symbol is b: scan right to find any a. If a is found, accept. Else, reject.
(write a, move R)

4. if input symbol is a: replace it with blank, move right (reading past a's), read b, move right (reading past blanks), read a, replace it with blank, move left and put head back at left end.

$\sqcup \sqcup \quad \sqcup \sqcup \rightarrow$

$\cancel{a}\,\cancel{a}\,b\,a\,a\,\sqcup\,\sqcup\,\sqcup\,\sqcup$

Trace the TM on aabaaaa

$q_0\,aabaaaa \vdash^* q_0\,\$\,aabaaaa \vdash^* \cdots \vdash^* \$\sqcup\sqcup b \sqcup\sqcup a\,q_a\,a$

$L(TM) = L$!

Trace the TM on aba — TM does not halt!

Trace the TM on bab — TM rejects

$L = \{ w : \text{TM accepts } w \}$

Universal TM

→ simulator, VM

$\langle \{qa, qr\}, \{0,1\}, \{0,1\}, (qa, 0, qa, 0, L), (qa, 1, qa, 1, L), (\quad), (\quad), (\quad), qa, qa, qr \rangle$

string over alphabet (eg. ASCII)

$\langle M \rangle$: Encoding of a TM M, in some suitable alphabet

$\langle M, w \rangle$: Encoding of a TM M & w, in some suitable alphabet

Design a DTM U s.t.

U(x): "Hello" → reject.

if x is not of the form $\langle M, w \rangle$ for some M, w: reject

Run M on w and do what M does

```
def simulate(desc, inp):
    // simulates TM on 0010
```

desc → of a TM

inp → eg. 0010

$\langle M \rangle ; w$

```
$ python a.py 1 2 "Hello"
```

a.py: program

1 2 "Hello": input to program

Challenge: "Best" universal TM?

Write an algorithm/TM to produce a list of ALL TMs (maybe, with duplicates).

scratch/rough

Output

after every TM, enters into a special state q_{out}

a b c d . . . z [ ] < > { , ; . .

aa ab ac . . .

(1) program to generate all strings over ASCII

\+ (2) filtering program to check if a string encodes a TM

$$ALLTM = \{\langle M\rangle : M \text{ is a TM}\} \in RE.$$