(Aside: $E_{DFA} = \{ \underbrace{(\{q_0\}, \{a,b\}, \{\}, q_0, \emptyset \})}_{\text{one DFA that accepts no strings}}, \underbrace{(\{q_0\}, \{a,b\}, \ldots)}_{\text{another one, ...}} \ldots \}$

Decidability (cont'd)

$E_{DFA} = \{ \langle A \rangle \mid A \text{ is a DFA, } L(A) = \emptyset \}$

Theorem: E_{DFA} is decidable.

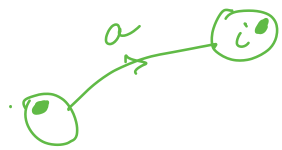
Proof: E_{DFA} is decided by TM T where:

T = "on input $\langle A \rangle$, where A is a DFA:

1. Mark the start state of A.
2. Until no new states get marked (ie in this iteration), do:
   2.1 Mark any unmarked state that has a transition into it from a marked state.
3. If no accept state is marked, ACCEPT. Otherwise, REJECT."

terminates

The loop in step 2 will terminate
- # iterations cannot be greater than number of states in A.



Correctness

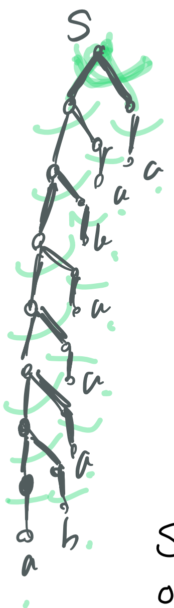
$\exists$ a string accepted by A $\iff$ $\exists$ a directed path from start state to an accept state in A.
$\iff$ an accept state gets marked by T. $\boxtimes$

$EQ_{DFA} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs, and } L(A) = L(B) \}$

$A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG, that generates string } w \}$

Theorem 5.1.4: A_{CFG} is decidable.

Proof: The TM S decides A_{CFG}:



S = " on input $\langle G, w \rangle$ where G is a CFG and w a string:

1. Convert G to CNF.
2. List all derivations of length $2n-1$ where $|w| = n$ (or if $w = \varepsilon$, where $n = 1$).
3. If any of the derivations derive w, ACCEPT. otherwise, REJECT. ".

S always halts because $\exists$ a finite # of derivations of length $2n-1$. It is correct, because it only accepts if it finds a derivation of w. $\blacksquare$

$E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \}$.

$\emptyset \neq \{\varepsilon\}$

eg

$D \to ABC$

$A \to Aa \mid BB$

$B \to bB \mid b$

$C \to Dc$

Theorem: E_{CFG} is decidable.

Proof: E_{CFG} is decided by TM R :

R = " on input $\langle G \rangle$, where G is a CFG :

1. mark all terminals in G (on RHS of rules)
2. Repeat until no new variables are marked :
   2.1 Mark any variable A where A has a rule $A \rightarrow \underbrace{\sigma_1 \sigma_2 \cdots \sigma_n}_{\text{all marked}}$.
3. If start symbol is not marked, ACCEPT. otherwise REJECT. "

we argued as to why this is correct; it terminates because at least one variable is marked at each iteration – we run out of variables in finite # of steps]

$\blacksquare$

Eg R accepts $(\{S\}, \{a,b\}, \{(S \rightarrow SS)\}, S)$

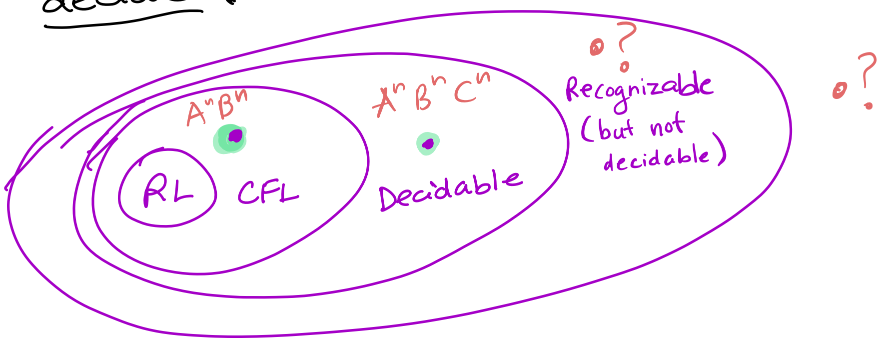
R rejects $(\{S\}, \{a\}, \{(S \rightarrow a)\}, S)$

$S \rightarrow AB$
$A \rightarrow 01 \mid 0A$
$B \rightarrow 0A1 \mid 1A0$

Chapter 5.14 ... <u>Undecidability</u>

$\exists$? a problem / language that a computer (TM) cannot recognize?

$\exists$? a problem / language that a computer (TM) cannot <u>decide</u>?



Claim: The language TMencodings $= \{\langle M\rangle \mid M \text{ is a TM over } \Sigma\}$ is countably infinite.

Proof: Sketch: 1. Each TM has an encoding over the alphabet $\gamma = \{ (,), ;, 0, 1, \{, \} \} \cup \Sigma$.

2. The strings over γ are enumerable (eg in shortlex order)

3. Remove the strings that are NOT a TM encoding, and you are left with a loose enumeration of TM encodings.

Theoretically, we could enumerate all the TM encodings in shortlex order. $\langle M_1\rangle, \langle M_2\rangle, \langle M_3\rangle, \ldots$

correspondingly we have a list

$M_1, M_2, M_3, \ldots$ $\blacksquare$

Claim: The set of languages over Σ is uncountable.

Proof:

shortest enum of strings over Σ.

enum ? of the language over Σ.

	ε	a	b	aa	ab	ba	bb	aaa	...
L_1	1	0	0	1	1	1	1	0	...
L_2	0	1	1	0	1	0	1	1	...
L_3	1	1	1	0	1	0	1	1	...
L_4									...
⋮									

← inclusion vector

$\blacksquare$

Corollary: $\exists$ languages that are not Turing recognizable.

Do $\exists$ languages that are recognizable but not decidable?

Let us define a table S "encoding acceptance"

TM \ input	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	...
M_1	0	1	0	0	.
M_2	1	0	1	1	1
M_3	0	0	1	0	0
M_4	1	0	1	1	0
M_5	1	0	0	..	

"0" here means M_3 does not accept $\langle M_2\rangle$

← here, we are only interested in what TMs due when their input is the encoding of some TM!

Let $D[i] = S[i][i]$

if $D[i] = \begin{cases} 1 & \text{then } M_i \text{ accepts } \langle M_i \rangle \\ 0 & \text{then } M_i \text{ does not accept } \langle M_i \rangle \end{cases}$

$\text{SelfAcc} = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \langle M \rangle \}$

SelfAcc is the language of TMencodings that are encodings of TMs that would halt-accept when run on their own encoding as input.

$\text{SelfNotAcc} = \{ \langle M \rangle \mid M \text{ is a TM that does not accept } \langle M \rangle \}$

Note that $\text{SelfNotAcc} = \overline{\text{SelfAcc}} \cap \text{TM encodings}$

We have seen that TMencodings is decidable.

The following theorem is useful:

Theorem: The class of decidable languages is closed under complement, $\cap$, $\cup$.

Proof: $\cup$, $\cap$ left as an exercise.

Complement: let L be any decidable language, and let M be a TM that decides L. Then we can construct a new TM $\bar{M}$ that decides $\bar{L}$ as follows:

$\bar{M}$ = " on input $\langle w \rangle$

1. Run M on $\langle w \rangle$;
   if M accepts, REJECT.
   if M rejects, ACCEPT."

1 Always terminates.

2. Correct.

Mar 21

Theorem: SelfNotAccept is <u>not decidable</u>

Proof: We have seen that the TM encodings are enumerable, for example in shortlex order. Then we can construct the following table:

Table S

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	$\langle M_6 \rangle$	$\langle M_7 \rangle$	$\cdots$	$\langle M_{1059} \rangle$	$\cdots$	$\langle M_{29467} \rangle$
M_1	0	1	0	1	1	0	1	$\cdots$			
M_2	1	1	0	1	1	1	0	$\cdots$			
M_3	0	0	1	1	1	1	1	$\cdots$			
M_4				0				$\cdots$			
$\vdots$											
M_{1059} = SelfAcc	0	1	1	0	1	0	1	$\cdots$			
M_{29467} = SelfNotAccept								$\cdots$			

Each TM M_i can be fed input $\langle M_j \rangle$, and M_i either accepts or does not.

$$\text{SelfAcc} = \{ \langle M_i \rangle \mid S[i][i] = 1 \}.$$

and the inclusion vector of SelfAcc is the diagonal of Table S.

The inclusion vector of SelfNotAcc is the "bit flip" of the diagonal of Table S.

$0 \to 1$
$1 \to 0$

But then where is TM SelfNotAccept in our enumeration?

It's inclusion vector is the flip of the diagonal, so this inclusion vector is NOT in the table itself — it is not a TM that can exist!

Another way of putting it is the following:

Theorem 4: SelfNotAccept is not decidable.

Proof: BWOC. Suppose $\underbrace{\text{SelfNotAccept}}_{\text{a language}}$ is decided by some TM $\underbrace{X}_{\text{a TM}}$.

Note: $\forall M_i$, $\boxed{M_i \text{ does not accept } \langle M_i \rangle} \Leftrightarrow \boxed{M_i \in \text{SelfNotAccept}}$

If X accepts $\langle X \rangle$, then

$\Rightarrow X$ is self-accepting

$\Rightarrow \langle X \rangle \notin$ SelfNotAccept

$\Rightarrow X$ does not accept $\langle X \rangle$

$\Rightarrow\Leftarrow$

If X does not accept $\langle X \rangle$

$\Rightarrow X$ is not self-accepting

$\Rightarrow \langle X \rangle \in$ SelfNotAccept

$\Rightarrow X$ accepts $\langle X \rangle$

$\Rightarrow\Leftarrow$

Either way, there is a contradiction.

$\therefore X$ does not exist, and SelfNotAccept is undecidable. $\blacksquare$

Theorem: SelfAccept is not decidable.

Proof: BWOC. Suppose SelfAccept is decided by some TM Y.

Then we can construct a TM X that decides SelfNotAccept as follows:

X = " on input $\langle M \rangle$ where M is a TM

1. Run Y on $\langle M \rangle$
   - if Y accepts, REJECT.
   - if Y rejects, ACCEPT."

X clearly returns the opposite answer to Y, accepting TM-encodings that reject themselves, and rejecting those that accept their own encodings — i.e., X decides SelfNotAcceptance, which contradicts Thm 4, above. $\blacksquare$

Note: we skipped the part where we reject the input if it is not a proper TMencoding. Perhaps it should read...

X = " on input <w> :

0. Run T on <w>, where T is a TM that decides whether <w> is a properly formatted TM-encoding. If T rejects, REJECT. otherwise, go to step 1.

1. Run Y on <w>.
   If Y accepts <w>, REJECT.
   If Y rejects <w>, ACCEPT. "

It is our practice to just represent the above step 0 as " on input <M>, where M is a TM "