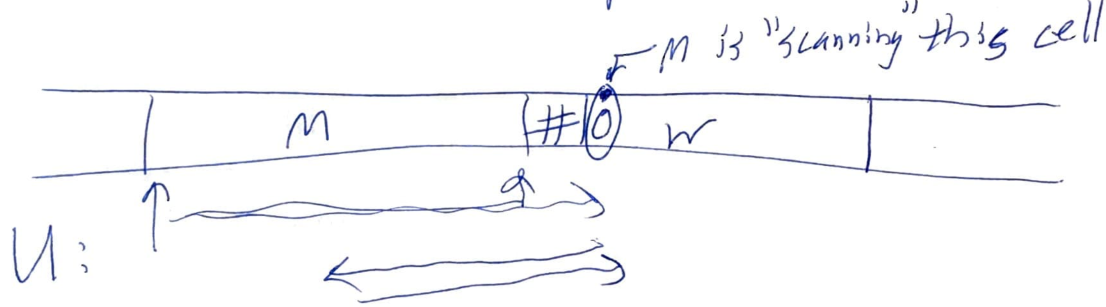
① # Decidability & Undecidability

CSCE 355
4/13/2022

<u>Last time</u>: Universal TM U

such that, for every TM M and input string w, U behaves (accept/reject/loop) on input $\langle M, w \rangle$ just as M behaves on input w.



$w \in \{0,1\}^*$ U's tape alphabet includes $0, 1, \dot{0}, \dot{1}$, etc.

<u>Def</u>: A TM is a <u>decider</u> if it halts on all inputs (never loops), and so it either accepts or rejects every input.

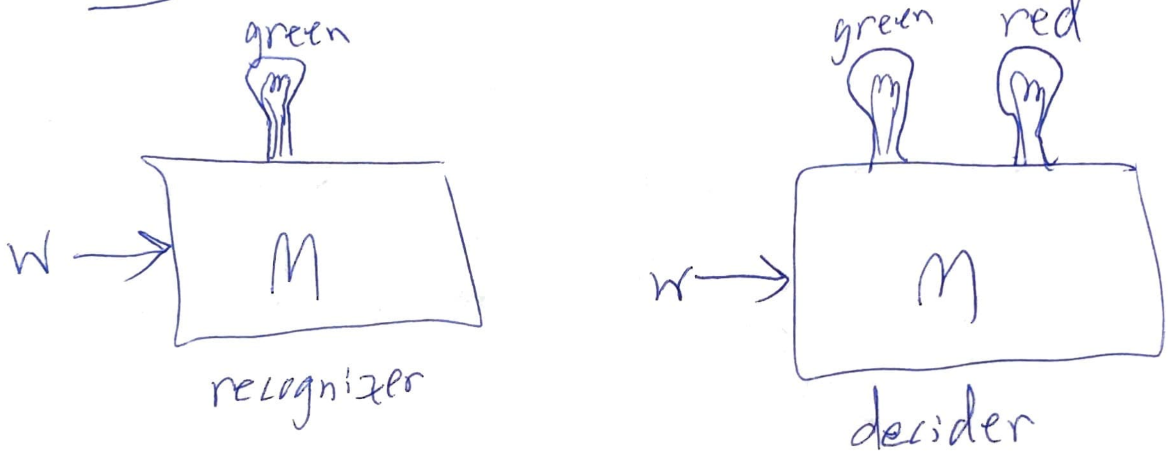
<u>Def</u>: Let TM M be arbitrary. The language <u>recognized</u> by M ($L(M)$) is

$$L(M) := \{ w \in \Sigma^* : M \text{ accepts } w \}$$

If $w \in L(M)$, M on input w eventually halts in accepting state

(2) If $w \notin L(M)$, then M on input w either halts in a rejecting state or runs forever (<u>loops</u>).

If M is a decider, then we say that M <u>decides</u> $L(M)$.



Say that a language L is <u>Turing-recognizable</u> (T-rec) if $L = L(M)$ for some TM M.

Say that L is <u>decidable</u> if $L = L(M)$ for some decider M.

Clearly, decidable $\Rightarrow$ T-rec.

but <u>not conversely</u>.

<u>Def</u>: The <u>halting problem</u> is the following language:

$$H := \{ \langle M, w \rangle : M \text{ is a TM, } w \text{ a string, and } M \text{ halts on input } w \}$$

③ <u>Theorem</u>: H is undecidable (i.e., not decidable).

<u>Proof</u>: Assume otherwise and suppose there exists a decider (TM) D that decides H.

Let F be a TM that implements the following algorithm:

F := "On input $\langle \overset{F}{M} \rangle$ where $\overset{F}{M}$ is a TM:

1. Builds the string $\langle \underbrace{M}_{TM}, \underbrace{\langle M \rangle}_{string} \rangle$ // $\langle F, \langle F \rangle\rangle$
2. Run D on input $\langle \overset{F}{M}, \overset{\langle F \rangle}{\langle M \rangle} \rangle$
3. If D accepts $\langle \overset{F}{M}, \langle \overset{F}{M} \rangle \rangle$ then loop ~~reject~~ // F ~~rejects~~ loops on its input $\langle M \rangle$
4. Otherwise // D rejects $\langle M, \langle M \rangle \rangle$ accept (halting) // F accepts $\langle M \rangle$"

Consider F running on input $\langle F \rangle$

<u>If</u> D accepts $\langle F, \langle F \rangle\rangle$ in step 3, then by assumption, F must halt on input $\langle F \rangle$, but F loops instead ☇

<u>If</u> D rejects $\langle F, \langle F \rangle\rangle$ then by assumption,

(4) F must loop ~~on~~ on input $\langle F \rangle$, but F accepts (halting on) $\langle F \rangle$ instead ↯

So D cannot be right on input $\langle F, \langle F \rangle \rangle$.

Thus no such D can exist,

$\therefore$ H is undecidable. ▨

Another way to look at it:

Let $M_1, M_2, M_3, \ldots$ be a list of all TMs

such that, given i, a program can recover M_i.

H: TMs	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	...	$\langle F \rangle$
M_1	halt	halt	loop		
M_2	loop	halt			
→ M_3			loop		
$\vdots$					
F					D is wrong

$\langle M_i, \langle M_i \rangle \rangle$

(5) **Def**: Define

$$A_{TM} := \{\langle M, w\rangle : M \text{ accepts } w\}$$

Thm: A_{TM} is undecidable.

Proof outline: Assume A_{TM} is decidable. Then we can use a decider for A_{TM} to decide H. $\lightning$ $\therefore A_{TM}$ is undecidable. Consider the following algo:

$G :=$ "On input $\langle M, w\rangle$ where M is a TM & w a string:

1. Modify M by making all its states accepting states, obtaining a TM M'.

   Comment $\begin{cases} M \text{ halts on } w, \text{ then } M' \text{ accepts } w \\ M \text{ loops on } w, \text{ then } M' \text{ loops on } w \end{cases}$

2. Run an assumed decider D for A_{TM} on input $\langle M', w\rangle$

3. If D accepts $\langle M', w\rangle$ then accept $\langle M, w\rangle$ else reject $\langle M, w\rangle$ // D rejects $\langle M', w\rangle$"

G correctly decides H:

6 $\langle M, w \rangle \in H \iff M$ halts on input w

$\iff M'$ accepts input w

$\iff \langle M', w \rangle \in A_{TM}$

$\iff G$ accepts $\langle M, w \rangle$

Furthermore G is a decider.

$\therefore$ H is decided by G $\lightning$

$\therefore$ A_{TM} is undecidable. $\boxtimes$

<u>Def</u>: Let

$$ALL_{CFG} := \{\langle G \rangle : G \text{ is a context-free grammar and } L(G) \text{ contains all strings over its terminal alphabet}\}$$

Will show: ALL_{CFG} is undecidable.