Undecidability

COMPSCI 3331

Outline

- Undecidability: definitions.
- Undecidability by diagonalization.
- Basic undecidability result: Halting problem.
- Universal TMs.
- Reductions.
- PCP and CFG undecidability.

Undecidability

- A problem is undecidable if there is no TM which solves it.
- The fact that **there exist undecidable problems** is one of the major contributions of theoretical computer science.
- We show that new problems are undecidable from existing undecidable problems by **reduction**.

Problems and Languages

- A **problem** P assigns each word $w \in \Sigma^*$ a yes/no answer. Let $P(w)$ denote this yes/no answer.
- e.g., Primality: given a word $w \in \{0,1\}^*$, is w the binary representation of a prime number?
  - P_{prime} denotes this function.
  - $P_{\text{prime}}(101) = \text{yes}$
  - $P_{\text{prime}}(10110) = \text{no}$

decision problem.

MAX

list, num $\longrightarrow$ is num the max in the list?

Undecidability

- We translate **problems** into **languages**: For every problem P, we associate it with

$$L_P = \{x \in \Sigma^* \ : \ P(x) = \text{ yes } \}.$$

- e.g., primality:

$$\begin{aligned} L_{\text{prime}} &= \{x \in \Sigma^* \ : \ x \text{ encodes a prime number } \}. \\ &= \{10, 11, 101, 111, \ldots, \} \end{aligned}$$

Undecidability

- From last time:
  - a language L is recursively enumerable (r.e.) if there exists a TM M which *accepts* L.
  - The language L is recursive if there exists a TM M which *recognizes* a language L (i.e., M **always halts**).
- A problem P is **undecidable** if L_P is **not** recursive.
- We also say that any language L which is not recursive is an undecidable language.

First Undecidable Language

- By **diagonalization** (Cantor).
- Need to define an ordering of TMs: $M_1, M_2, M_3, \ldots$.
- To do this, we will encode a TM as a word over $\{0, 1\}$.
- The *i*-th TM will be the *i*-th word over $\{0, 1\}$.

Encoding TMs

Let $\Sigma = \{0,1\}$. Let $M = (Q, \{0,1\}, \Gamma, \delta, q_1, B, F)$ be a TM. We can rename the states and tape alphabet so that:

- $Q = \{q_1, q_2, q_3, \dots, q_r\}$ for some $r \geq 1$. We can also assume that $F = \{q_r\}$.
- tape alphabet
  $\Gamma = \{\alpha_1, \alpha_2, \dots, \alpha_s\}$ for some $s \geq 3$. We assume $\alpha_1 = 0, \alpha_2 = 1$, and $\alpha_3 = B$.

Consider a transition $\delta(q_i, \alpha_j) = (q_k, \alpha_\ell, D)$. We encode this **single** transition as the word

$$0^i 1 0^j 1 0^k 1 0^\ell 1 0^{m(D)}$$

where $m(D)$ is $1, 2, 3$ if D is L, S, R, respectively.

(state from) 1 (symb. read) 1 (state to) 1 (symb. write) 1 (dir.)

Encoding TMs

We now encode the **entire** TM $M = (Q, \Sigma, \Gamma, \delta, q_1, B, F)$.
Let $C_1, C_2, \ldots, C_m$ be the encodings of the m transitions of the TM. Then we encode M as

$$e(M) = C_1 11 C_2 11 C_3 11 \cdots 11 C_m$$

- Can we decode the TM based on $e(M)$? ✓
- How can we guarantee that there are only finitely many transitions?

we have all finite components.

Ordering words and TMs

Let $\leq_\ell$ be the total lexicographical ordering of words over $\{0, 1\}$:

- if x is shorter than y, then $x \leq_\ell y$.
- if x, y have the same length, but x comes before y in lexicographical order, then $x \leq_\ell y$.

So:

- $00 \leq_\ell 111$
- $00 \leq_\ell 01$
- $00101 \leq_\ell 00110$.

Ordering words and TMs

$\leq_\ell$ imposes a total order on all words over $\{0,1\}$:

$\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, \ldots$

- Let w_i be the i-th word in this order. $w_1 = \varepsilon; w_2 = 0$, etc.
- We can design a TM that starts with i (in binary) on its tape and halts with w_i on its tape.
- Order TMs: M_i is the TM with $e(M_i) = w_i$. (Recall that $e(M) \in \{0,1\}^*$.)

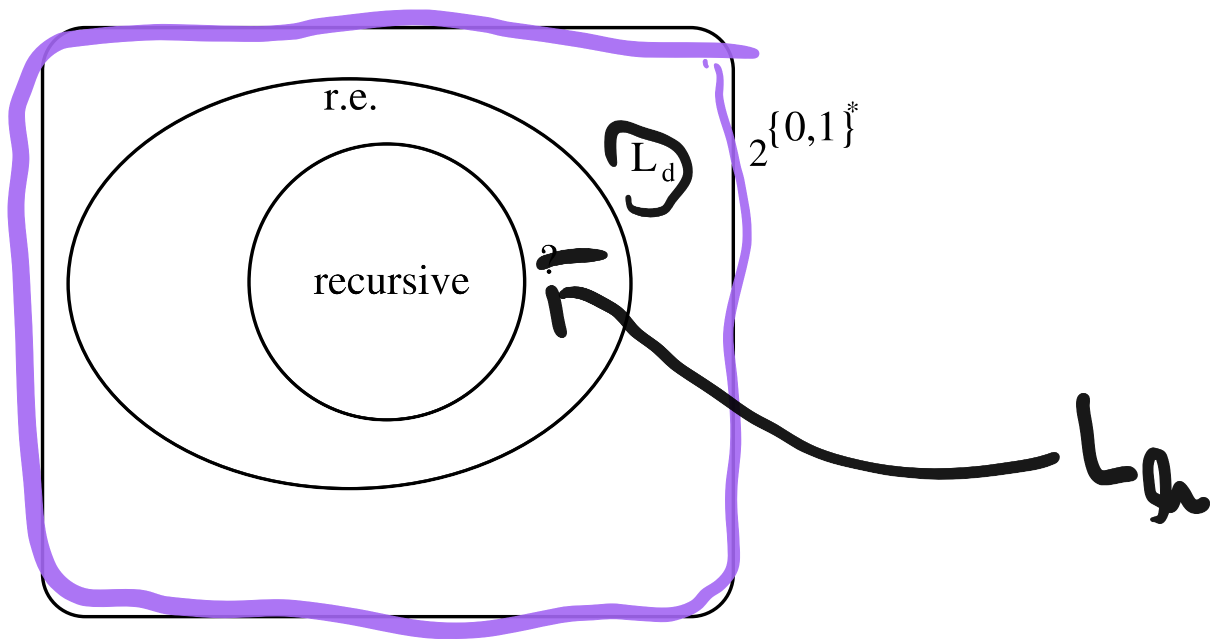
Diagonalization

$w_i \longleftrightarrow e(M_i)$

Define

$$L_d = \{w_i \in \{0,1\}^* \ : \ i \geq 0, w_i \notin L(M_i)\}.$$

Thm. L_d is not r.e.



Is there a language L which is r.e. but not recursive?

An undecidable problem that is r.e.

Consider the following problem (*the halting problem*):

Given a TM M, and a word w, does M accept w?

can't use for general purpose question. undecidable. => not recursive but it is r.e.

- The input to this problem is $(e(M), w)$.
- This problem is r.e.: we **can** give a TM which halts and accepts if the answer to the problem is yes (later...)
- This problem is undecidable: it is not recursive.

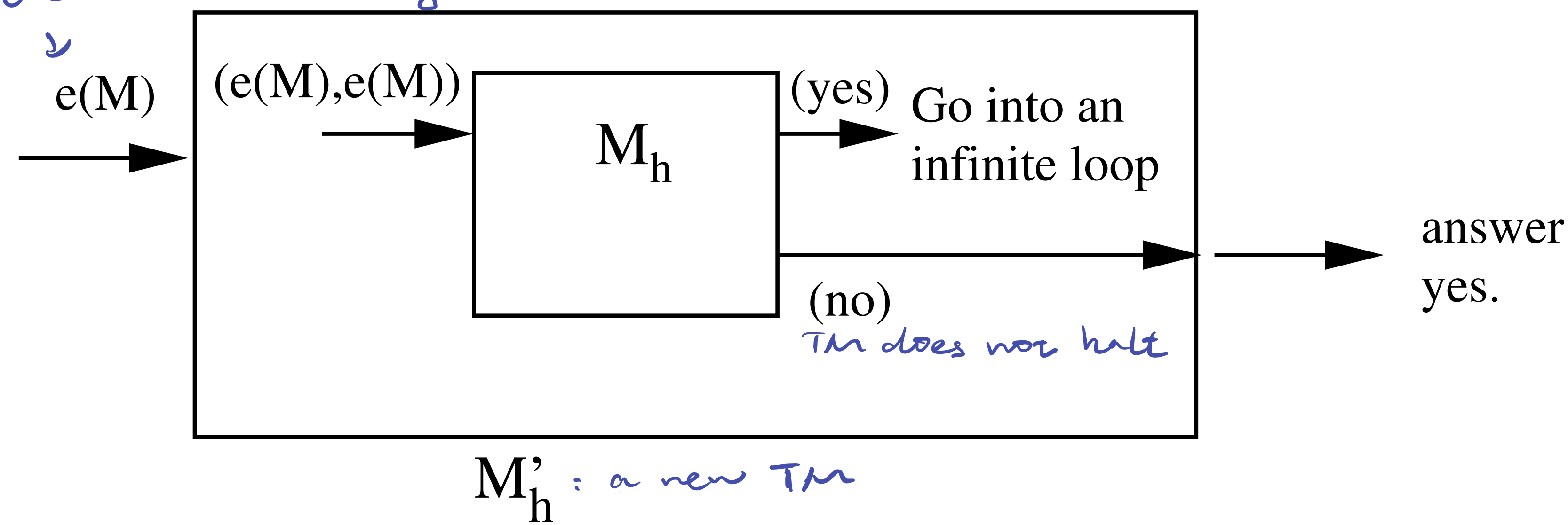
no algorithm.

Halting problem proof by contradiction.

Thm. The halting problem is undecidable.

Pf. Suppose there is a TM M_h which solves the halting problem $(e(M), w)$ and always halts with a yes/no answer.
Let M'_h be the following TM:

entire TM with encoding.



It does do anything.

What does M'_h do when it gets $e(M'_h)$ as input?

halting machine.

① M'_h ask M_h → do I stop when I get $e(M'_h)$ as input?

Ⓐ if M_h says yes.

- that means $e(M'_h) \in L(M'_h)$
- action of M'_h → goes into infinite loop.

Ⓑ if M_h says no

def of M'_h does.

- that means $e(M'_h) \notin L(M'_h)$
- action of M'_h → says yes.

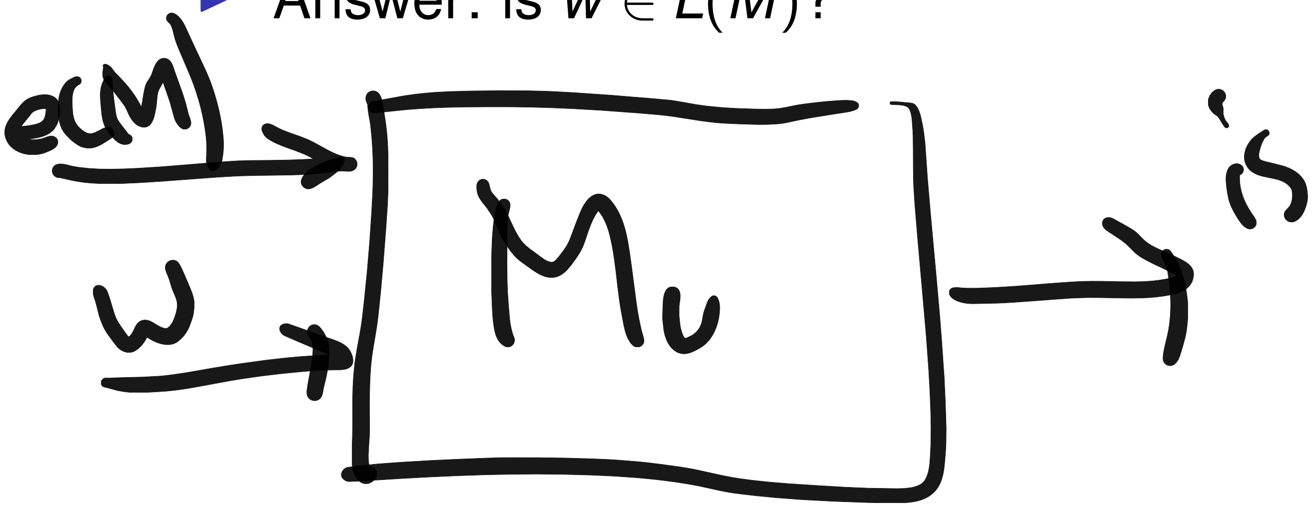
Both A and B reach contradiction.

Both A and B reach ...

Halting Problem

- We have shown that the halting problem “Does M halt on w?” is undecidable.
- Want to show that it is recursively enumerable: There exists a TM M_u such that if M halts on w, M_u halts and says yes.
- If M does not halt on w, M_u may not halt.
- We call this TM M_u a universal TM.
  - M_u gets $e(M), w$ as input.
  - Answer: is $w \in L(M)$?

universal ability to simulate any word.



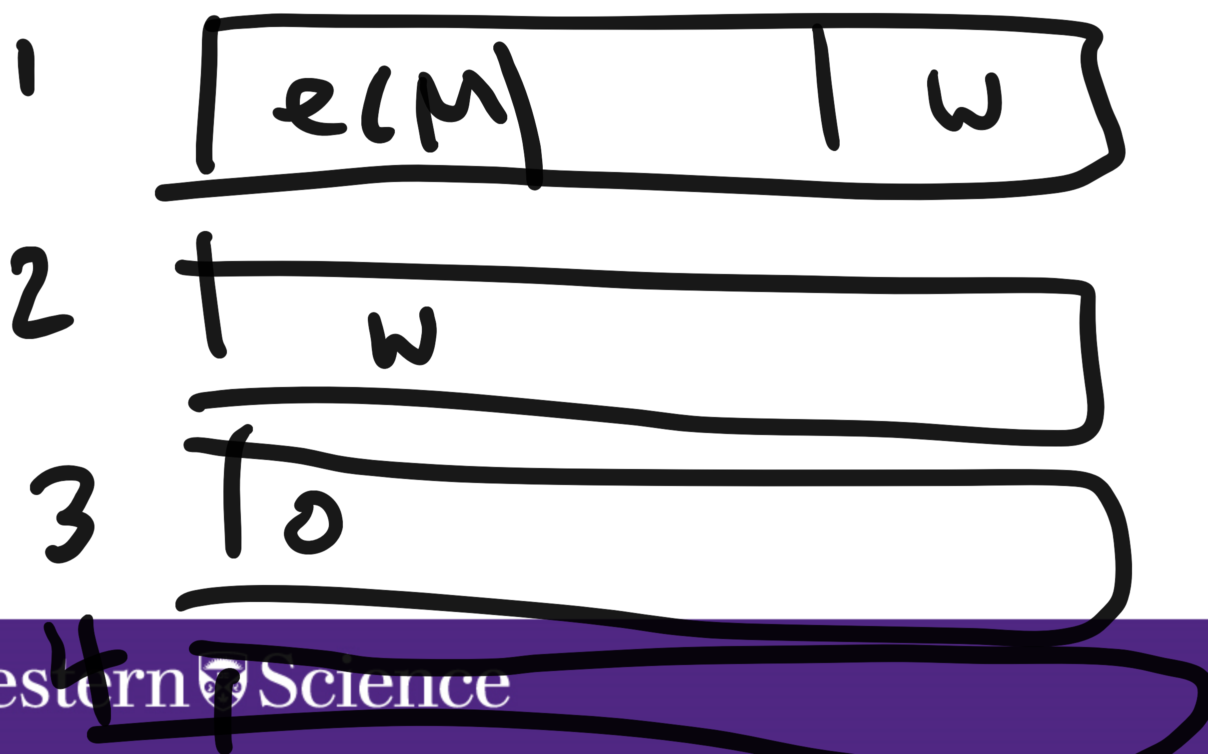
is $w \in L(M)$?

is w in the language accepted by TM?

Universal TM

M_u has at four tapes:

- On tape 1 is the input word $(e(M), w)$.
- Tape 2 will simulate the input tape of M; initially we copy w from tape 1 to tape 2.
- Tape 3 will contain the current state of M; initially we write the code for the start state of M on tape 3 (encoded as 0).
- Tape 4 will be a scratch tape. extra space to write stuff.



Universal TM

M_u works as follows:

- After initialization, M_u simulates steps of M.
- M_u searches through the transitions of M and simulates one step of M, changing the input to M on tape 2, updating the head position on tape 2, and the state of M on tape 3.
- M_u faithfully simulates M for acceptance and crashing.
- If M accepts w, then so does M_u.

M goes into inf loop,
=> Mu also goes into inf loop.
But we only need to consider the
case yes.

Universal TM

- The halting problem is r.e.:

$$\{(e(M), w) \ : \ w \in L(M)\}.$$

- However, it is **not** decidable. not recursive provably no TM .
- Universal TMs play a big part in problems involving TMs: it is often useful to simulate a TM on a given input.

UTM.

More Undecidable Problems

We now have two undecidable problems:

- The diagonalization language.
- The halting problem.

Are there more? How can we find them?

Tool: TM computing functions.

computable. str to str.

- A TM M **computes a function** $f : \Sigma^* \to \Sigma^*$ if, whenever the TM gets [redacted] x as input, it halts and outputs $f(x)$ on the tape.
- There is no concept of acceptance and rejectance. Whatever happens when M halts, we take the tape contents as the result of applying f to the input.
- M is deterministic.
- Examples:

get a TM for these =>

1. $f(a^n) = n$ (written in binary).
2. $f(a^n b^m) = c^{n^m}$ for all $n, m \geq 0$.
3. $f((e(M_1), e(M_2))) = e(M)$ where $L(M) = L(M_1) \cup L(M_2)$.

More Undecidable Problems

- Let P_0 be an undecidable problem.
- We can show that another problem P_1 is undecidable by **reduction**.
- A reduction is a function f which can be computed by a TM and satisfies the following properties:

  computable.

  - if $x \in P_0$ then $f(x) \in P_1$.
  - if $x \notin P_0$ then $f(x) \notin P_1$.

if the computable satisfies, then it is reduction.

Using Reductions

- A reduction of problem P_0 to P_1 means that P_0 is **at least as hard** as P_1.
- Suppose that P_1 is decidable. To solve P_0: Take input x, convert it to $f(x)$, and decide whether $f(x) \in P_1$.
- This is an algorithm for P_0. Therefore

IMPORTANT

$$\cancel{P_1 \text{ decidable} \Rightarrow P_0 \text{ decidable.}}$$

$$\boxed{P_0 \text{ undecidable} \Rightarrow P_1 \text{ undecidable.}}$$

contrapositive.
build reduction.

$x \in ? P_0$

$f(x) \in P_1$?

M_1

yes → $x \in P_0$

no → $x \notin P_0$

P_1 is decidable.

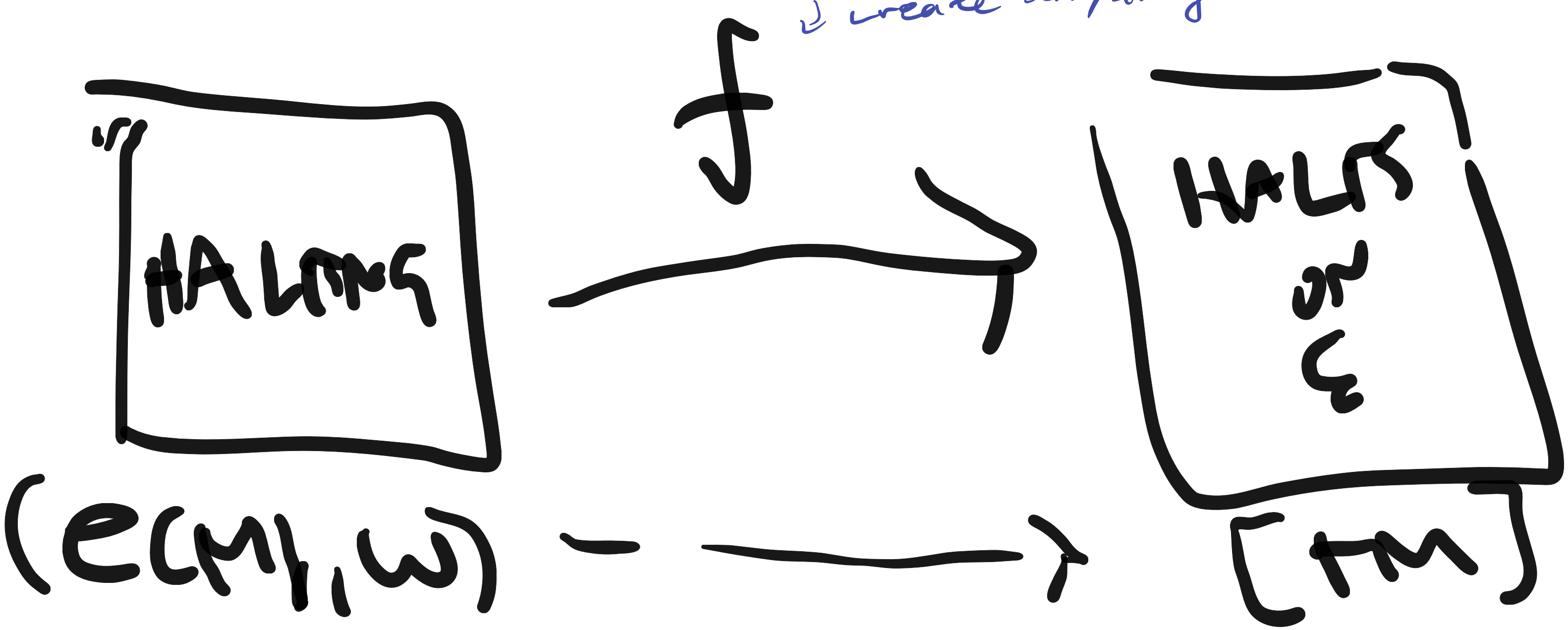
"Halts on ε"

Want to show that the following problem is undecidable (*halts on* ε):

> *Given a TM M, is $\varepsilon \in L(M)$?*

Use the halting problem show that "halts on ε" is undecidable.

we don't know,
create anything that is reduction.



"Halts on ε"

input

Suppose (M, w) is an instance of the halting problem. We map (M, w) to a new TM M_0. M_0 does the following on input x:

1. If $x \neq \varepsilon$, reject.
2. Otherwise, if $x = \varepsilon$, use M_u to simulate the action of M on w.
3. If M_u determines $w \in L(M)$, halt and accept ε.
4. If M_u determines $w \notin L(M)$, reject ε.

proving it is computable

$$(M, w) \rightarrow M_0 \textbf{ is our reduction!}$$

$\overline{\text{H.P.}}$

halting problem.

yes

no

$w \in L(M)$

$w \notin L(M)$

“Halts on ε”

What does M_0 do?

- If $w \in L(M)$, then $\varepsilon \in L(M_0)$.
- If $w \notin L(M)$, then $\varepsilon \notin L(M_0)$.

This is a reduction of the halting problem to “halts on ε”.
To solve the halting problem:

- Given (M, w), construct M_0.
- Use “halts on ε” to determine whether $\varepsilon \in L(M_0)$.
- This is an algorithm for the halting problem, therefore, an algorithm solving “halts on ε” must not exist.

therefore $\longrightarrow$ "halts on ε" is undecidable.

Post's Correspondence Problem

Post's Correspondence Problem (PCP) is a simple example of an undecidable problem:

Input: Two lists of words $(u_1, u_2, \ldots, u_n)$ and $(v_1, v_2, \ldots, v_n)$ of the same length $n \geq 1$.

Determine: does there exist indices $i_1, i_2, \ldots, i_m$ with $1 \leq i_j \leq n$ and $m \geq 1$ such that

$$u_{i_1} u_{i_2} \cdots u_{i_m} = v_{i_1} v_{i_2} \cdots v_{i_m}.$$

Thm. PCP is undecidable.

PCP: Examples

u $abc|bb$ X
v $\underbrace{ab}_{v_2}|c\,bc$

Example 1:

NO ←

i	1	2	3
u_i	ab	abc	bb
v_i	b	ab	cbc

Example 2:

$i_1 = 2$ $i_2 = 3$ $i_3 = 1$

i	1	2	3
u_i	ab	abc	bb
v_i	b	ab	$cbba$

→ YES

u $abc|bb|ab$
v $ab|cbba|b$

PCP is undecidable

Thm. PCP is undecidable.

- Given (M, w), we construct a PCP instance $(u_1, u_2, \ldots, u_n), (v_1, v_2, \ldots, v_n)$
- M accepts w iff the PCP instance has a solution.
- Idea: the solution to PCP will be a proof that M accepts w: a list of the IDs of M showing how w is accepted.
- $q_0 w = \alpha_0 \vdash_M \alpha_1 \vdash_M \cdots \vdash_M \alpha_n$.
- The solution will be of the form

$$\alpha_0 \# \alpha_1 \# \alpha_2 \# \cdots \# \alpha_n.$$

tape contents
+
state

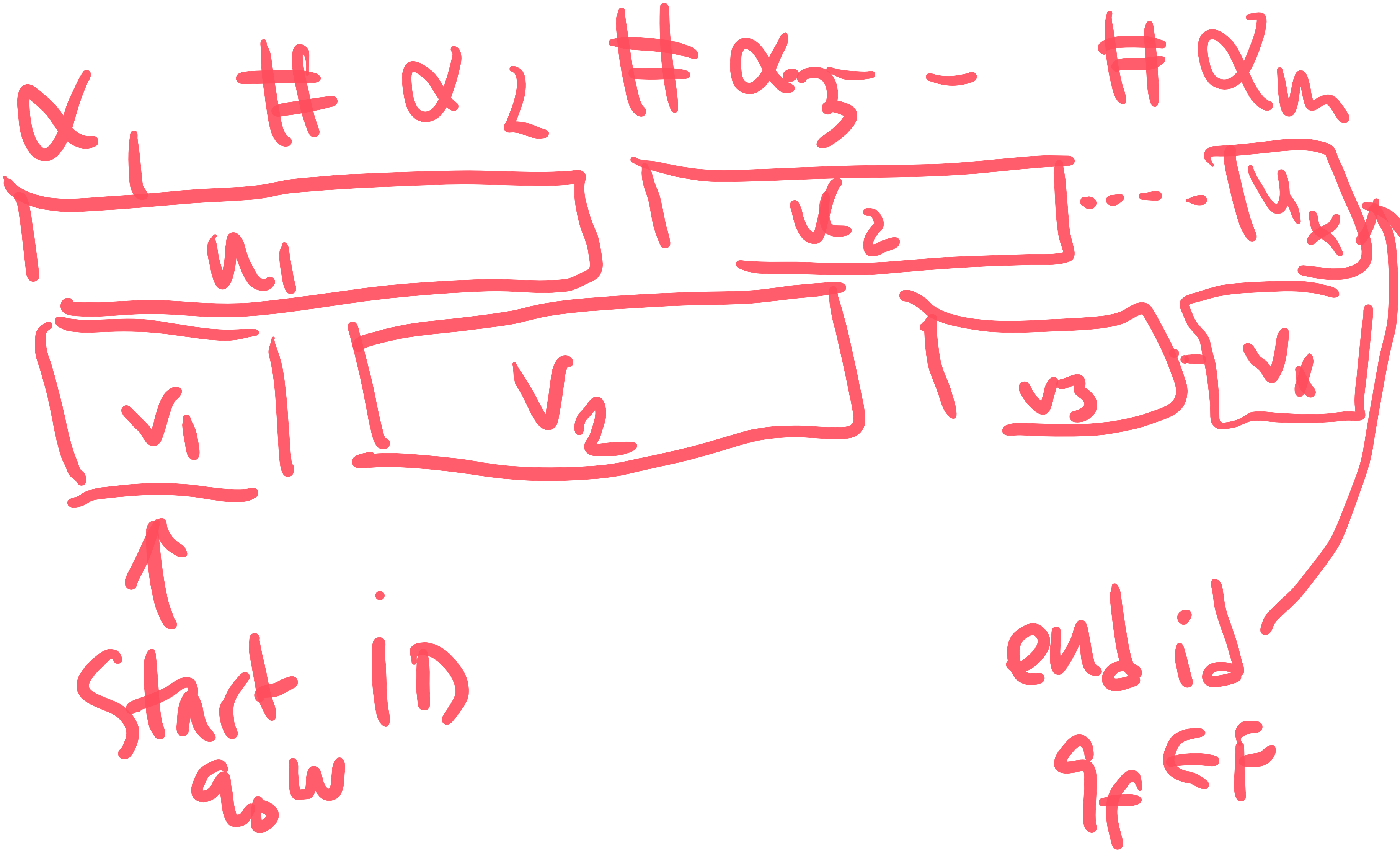
tape | q | tape

HALT $\xrightarrow{f}$ PCP

(M, w) $\rightarrow$ instance

$u_i \rightarrow$ simulate one step of TM

$v_i \rightarrow$ simulate one step of TM



Using PCP

Thm. Given CFGs G_1, G_2, it is undecidable whether $L(G_1) \cap L(G_2) = \emptyset$.

Thm. Given a CFG G, it is undecidable whether $L(G) = \Sigma^*$.

Thm. Given a CFG G, it is undecidable whether G is ambiguous.

PCP instance $\qquad\qquad$ CFG

$u_1, u_2, \ldots u_n \quad \longrightarrow \quad G_1$

$v_1, v_2, \ldots, v_n \qquad\qquad\quad G_2$.

G_1 $\{u_1, \dots, u_n\} \subseteq \Sigma$ $\{1, \dots, n\}$

$$S \to u_1 S 1$$
$$S \to u_2 S 2$$
$$\vdots$$
$$S \to u_n S n$$
$$S \to \#$$

$$L(G_1) = \{ u_{i_1} u_{i_2} \cdots u_{i_k} \# i_k i_{k-1} \cdots i_1 : 1 \le i_1, i_2 \cdots i_k \le n,\ k \ge 0 \}$$

Emil Post (1897–1954)