Computing Theory

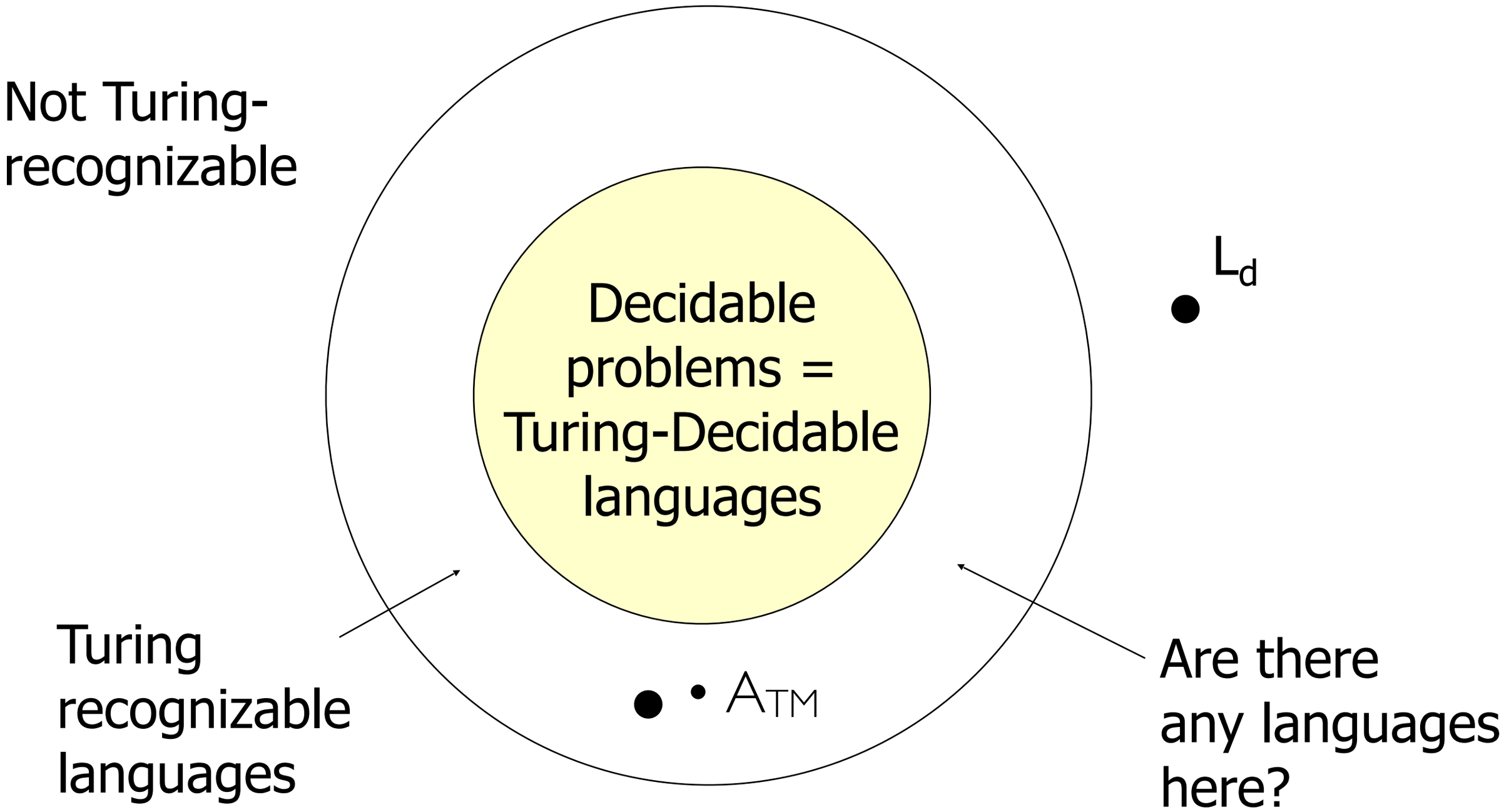
COMP 147 (4 units)

Chapter 5: Reducibility

Announcements

- Exam2
    - Thursday Nov. 17
- Review class on Tuesday
- Assignment5 due Monday Night
    - Will go over solutions during class on

Last Time: Bullseye Picture



Reducibility

- How can we prove that a problem is undecidable
- New Technique:
  Reduce this problem to another problem
- We use it to show that the following problems are undecidable
  - Is a given program guaranteed to halt
  - Are 2 TMs equivalent?
  - Will a given TM accept any string

Reducibility

- Problem A is reducible to problem B
  - solution to B can be used to solve A

- Implications
  - A cannot be a harder problem then B
  - If B is decidable, A is also decidable
  - If A is undecidable, B is also undecidable

- To prove problem B is undecidable,
  reduce some undecidable problem to B

Reducibility

- Normally used in the contrapositive.

- If we reduce L to L' and we know L is undecidable, then L' is undecidable.

$$A_{\mathsf{TM}} = \{\langle M, w\rangle |\ M \text{ is a } \mathsf{TM} \text{ and } M \text{ accepts } w\}$$

LOGIC

KNOWN FACT: A_{TM} IS UNDECIDABLE.

WHAT ABOUT SOME OTHER PROBLEM **P**...
IS **P** UNDECIDABLE?

THEOREM

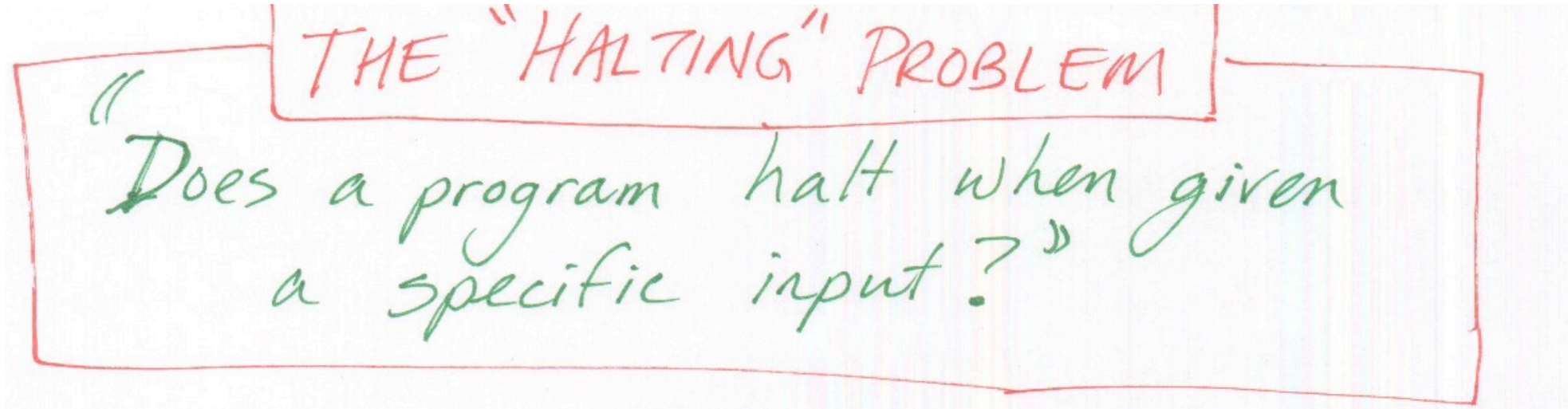
P IS UNDECIDABLE.

PROOF APPROACH

- Assume **P** IS DECIDABLE.
- Reduce A_{TM} (a "HARD" problem) into **P** (the "EASIER" problem)
- Use the solution of **P** to solve A_{TM}.
  Use the decidability of **P** to find an algorithm to decide A_{TM}.
  Build a TM to decide A_{TM} using the TM to decide **P** as a subroutine.
- But we know that a decider for A_{TM} cannot exist.

∴ Contradiction: **P** is not decidable!

The Halting Problem



- $HALT_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$
- $HALT_{\text{TM}}$ is undecidable. How can we prove it?

The Halting Problem

- $HALT_{\text{TM}}$ = { <M, w> | M is a TM and M halts on w }
- $HALT_{\text{TM}}$ is undecidable. How can we prove it?

- Proof: We reduce A_{TM} to $HALT_{\text{TM}}$
  - We show that if we could decide $HALT_{\text{TM}}$, the we decide A_{TM}
  - Proof by contradiction:
    1. Assume that $HALT_{\text{TM}}$ is decidable
    2. Show that (1) implies that A_{TM} is decidable (contradiction)
    3. Contradiction (2) means that $HALT_{\text{TM}}$ is not decidable.

$A_{\text{TM}} \leqslant HALT_{\text{TM}}$

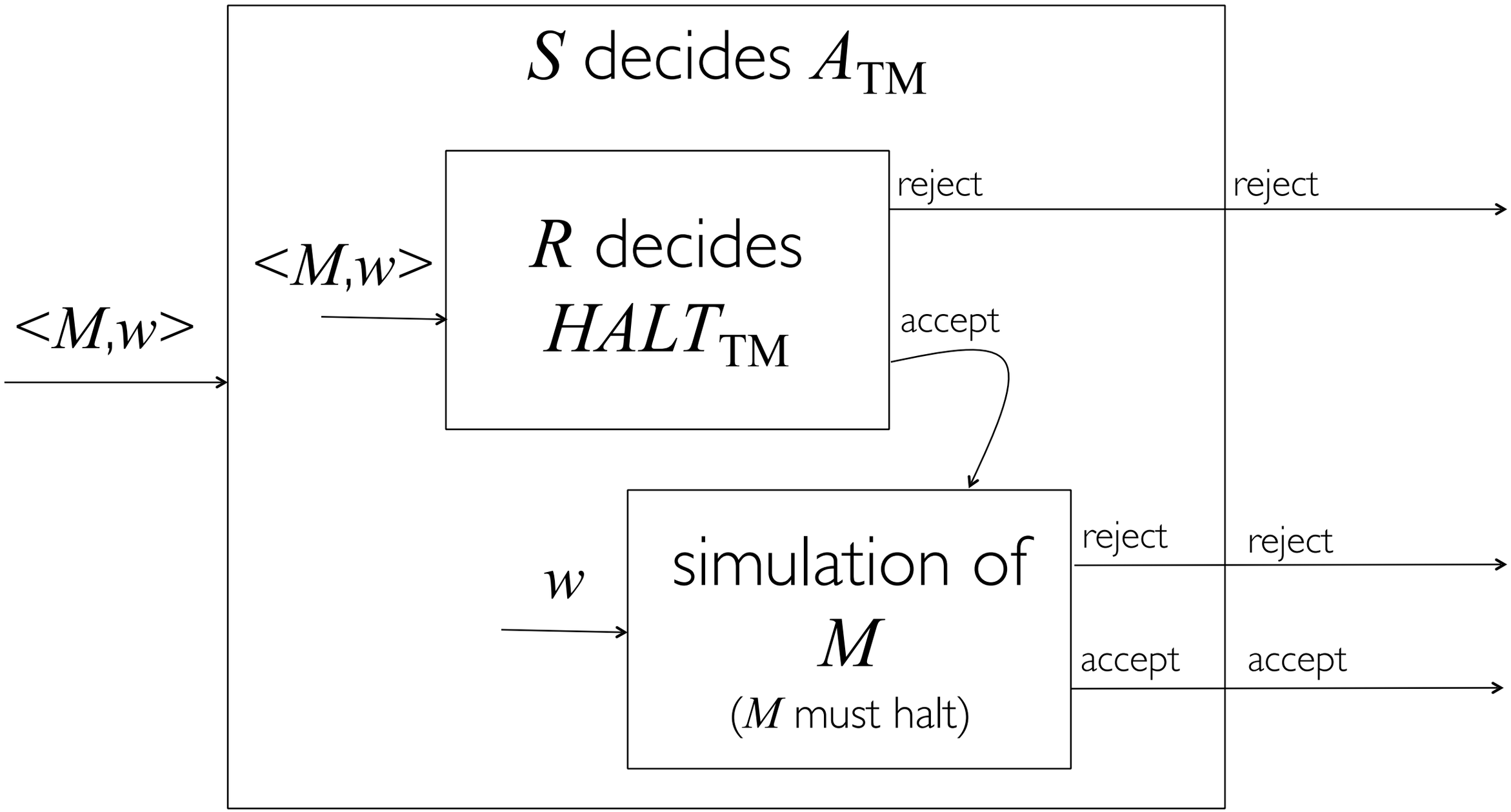
PROOF Let's assume for the purpose of obtaining a contradiction that TM R decides $HALT_{\text{TM}}$. We construct TM S to decide A_{TM}, with S operating as follows.

$S =$ "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

1. Run TM R on input $\langle M, w \rangle$.
2. If R rejects, *reject*.
3. If R accepts, simulate M on w until it halts.
4. If M has accepted, *accept*; if M has rejected, *reject*."

Clearly, if R decides $HALT_{\text{TM}}$, then S decides A_{TM}. Because A_{TM} is undecidable, $HALT_{\text{TM}}$ also must be undecidable.

$A_{\mathrm{TM}} \leqslant HALT_{\mathrm{TM}}$



E_{TM} is Undecidable

- $E_{\text{TM}} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \varnothing \}$

- We have a couple options:
  - Reduce A_{TM} to E_{TM}
  - Reduce $HALT_{\text{TM}}$ to E_{TM}

- We'll follow the book:
  - Assume R, a decider for E_{TM} exists
  - Construct S, a decider for A_{TM}, that use R

$A_{TM} \leqslant E_{TM}$ (insufficient idea)

- Assume R decides E_{TM}
- Construct S to decide A_{TM}
  - S receives $<M,w>$ and passes M to R
    - Case 1: R accepts M → L(M) is empty → M accepts no strings → M cannot accept w
    - Case 2: R rejects M → M accepts some strings,
      but S does not know if M accepts w

$A_{\mathrm{TM}} \leqslant E_{\mathrm{TM}}$

- S needs to modify M before passing it to R
- S adds preprocessing to M to get new TM M_1 that rejects any string that is not w
- S receives $\langle M, w \rangle$ and passes M_1 to R
  - Case 1: R accepts M_1 → L(M_1) is empty → M_1 accepts no strings → M cannot accept w
  - Case 2: R rejects M_1 → M_1 accepts some strings → since M_1 can only accept w, M_1 must accept w → M must accept w

$A_{\text{TM}} \leqslant E_{\text{TM}}$

M_1 = "On input x:

1. If $x \neq w$, *reject*.
2. If $x = w$, run M on input w and *accept* if M does."

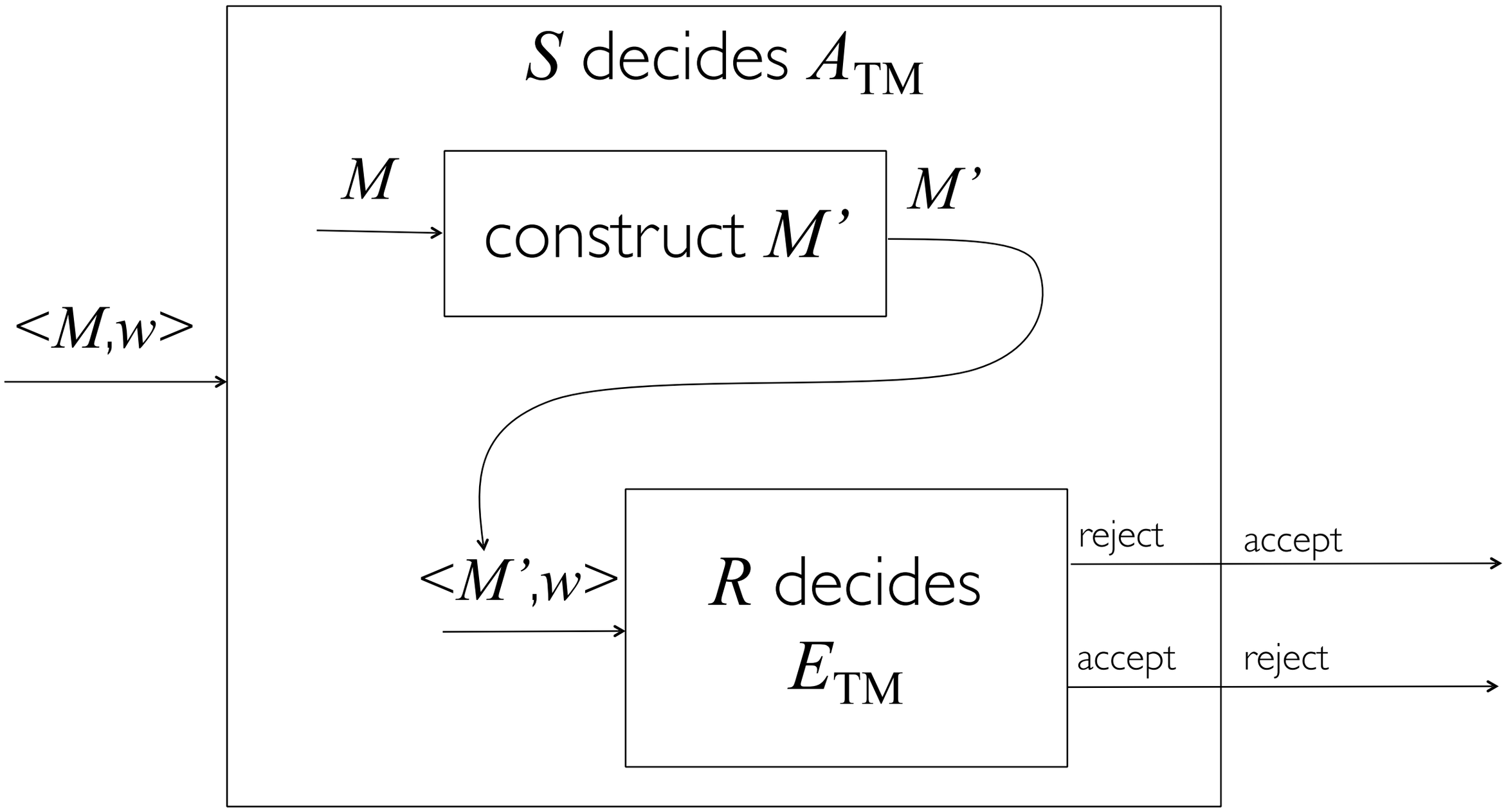
Putting all this together, we assume that TM R decides E_{TM} and construct TM S that decides A_{TM} as follows.

S = "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

1. Use the description of M and w to construct the TM M_1 just described.
2. Run R on input $\langle M_1 \rangle$.
3. If R accepts, *reject*; if R rejects, *accept*."

If R were a decider for E_{TM}, S would be a decider for A_{TM}. A decider for A_{TM} cannot exist, so we know that E_{TM} must be undecidable.

$A_{\mathrm{TM}} \leqslant E_{\mathrm{TM}}$



EQ_{TM} Is Undecidable

- $EQ_{\mathrm{TM}} = \{ \langle M_1, M_2 \rangle \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2) \}$

- Candidates for reduction:
  - A_{TM}: Does M accept w?
  - $HALT_{\mathrm{TM}}$: Does M halt on w?
  - E_{TM}: Is L(M) empty? (do in lab)

Examples of undecidable problems

Diophantine Equations

Instance: A multivariate polynomial equation such as

$$x^2 y + 3xy - y^2 - 17 = 0$$

Question: Does this equation have an integer-valued solution?

See "Hilbert's Tenth Problem is Unsolvable"
M.Davis, *American Mathematical Monthly* 80 (1973) pp.233-269

Examples of undecidable problems

Diophantine Equations

Instance: A multivariate polynomial equation such as

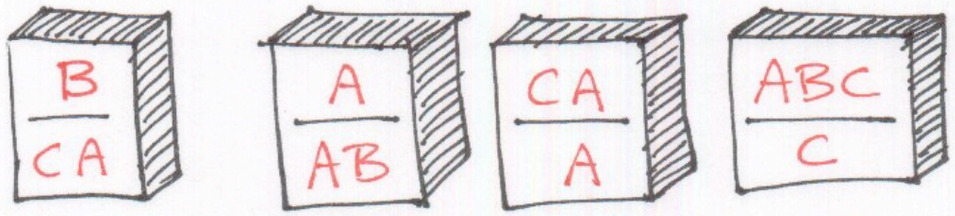
$$x^2 y + 3xy - y^2 - 17 = 0$$

Question: Does this equation have an integer-valued solution?

See "Hilbert's Tenth Problem is Unsolvable"
M.Davis, *American Mathematical Monthly* 80 (1973) pp.233-269

PCP: The Post Correspondence Problem

DOMINOS:



WE GET AS MANY OF EACH TYPE AS WE NEED.

GOAL: FIND A finite SEQUENCE OF DOMINOS SUCH THAT THE TOP AND BOTTOM STRINGS ARE THE SAME.

Example: PCP

- Let the alphabet be {0, 1}.
- Let the PCP instance consist of the two pairs (0, 01) and (100, 001).
- We claim there is no solution.
- You can’t start with (100, 001), because the first characters don’t match.

Example: PCP – (2)

Recall: pairs are (0, 01) and (100, 001)

0100100
01001001

But we can never make the first string as long as the second.

Must start with first pair

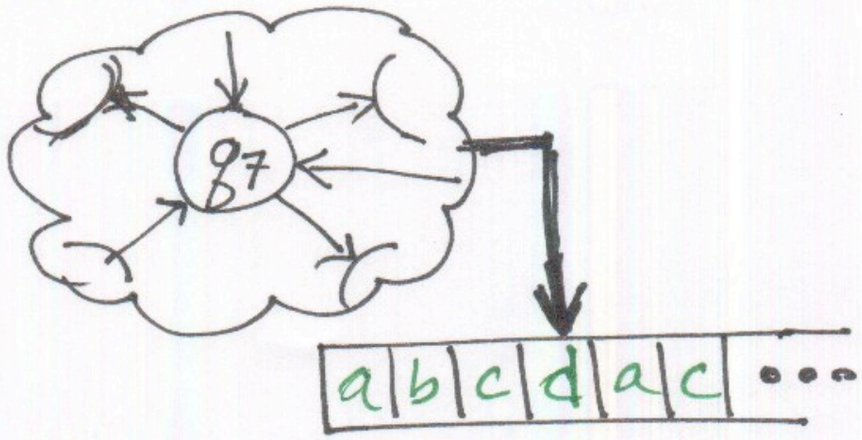
Can add the second pair for a match

As many times as we like

Example

- Is the answer to this instance of PCP “Yes”?
  (01110, 011) (101, 0101) (1110, 10111)

"CONFIGURATION"



C: a b c q_7 d a c

A SEQUENCE OF CONFIGURATIONS:

$\ldots \Rightarrow C_4 \Rightarrow C_5 \Rightarrow C_6 \Rightarrow C_7 \Rightarrow \ldots$

step in the computation.

AN "ACCEPTING COMPUTATION HISTORY"

$C_1 \Rightarrow C_2 \Rightarrow C_3 \Rightarrow C_4 \Rightarrow \ldots \Rightarrow C_L$

Initial Starting Configuration

Legal steps for this machine

An Accepting State

slides by Harry Porter PSU

Computation History

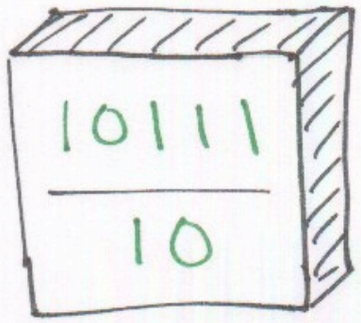
- If the machines does not halt?
    - no history
- A computation history is a finite sequence of configurations
- For a TM:
    - Accepting history
    - Rejecting history
    - no history if Machine Loops

PCP is undecidable

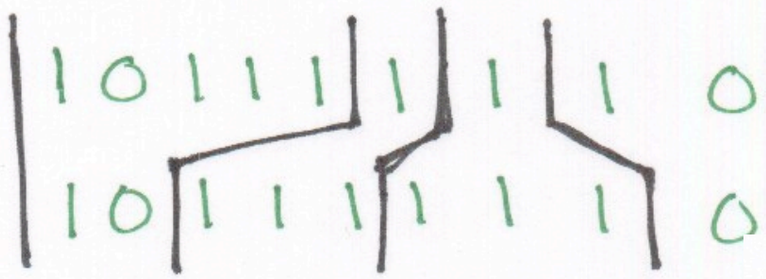
- *Proof Overview*
  - We reduce A_{TM} to PCP
  - We encode <M,w> into an instance of PCP
  - Assume that PCP is decidable
  - Show that this implies that A_{TM} is decidable (contradiction)
  - Contradiction! means that PCP is not decidable.

TILES:

	A	B
①	10111	10

$\equiv$  $\equiv$ $\left[\begin{array}{c} 10111 \\ \hline 10 \end{array}\right]$

PCP SOLUTIONS:



TURING MACHINE CONFIGURATIONS:

$10\,1\,q_4\,011$

COMPUTATION HISTORY:

$\# q_0 101 \# 1 q_4 01 \# \cdots \# 011 q_A 101 \#$

THIS IS AN ACCEPTING HISTORY

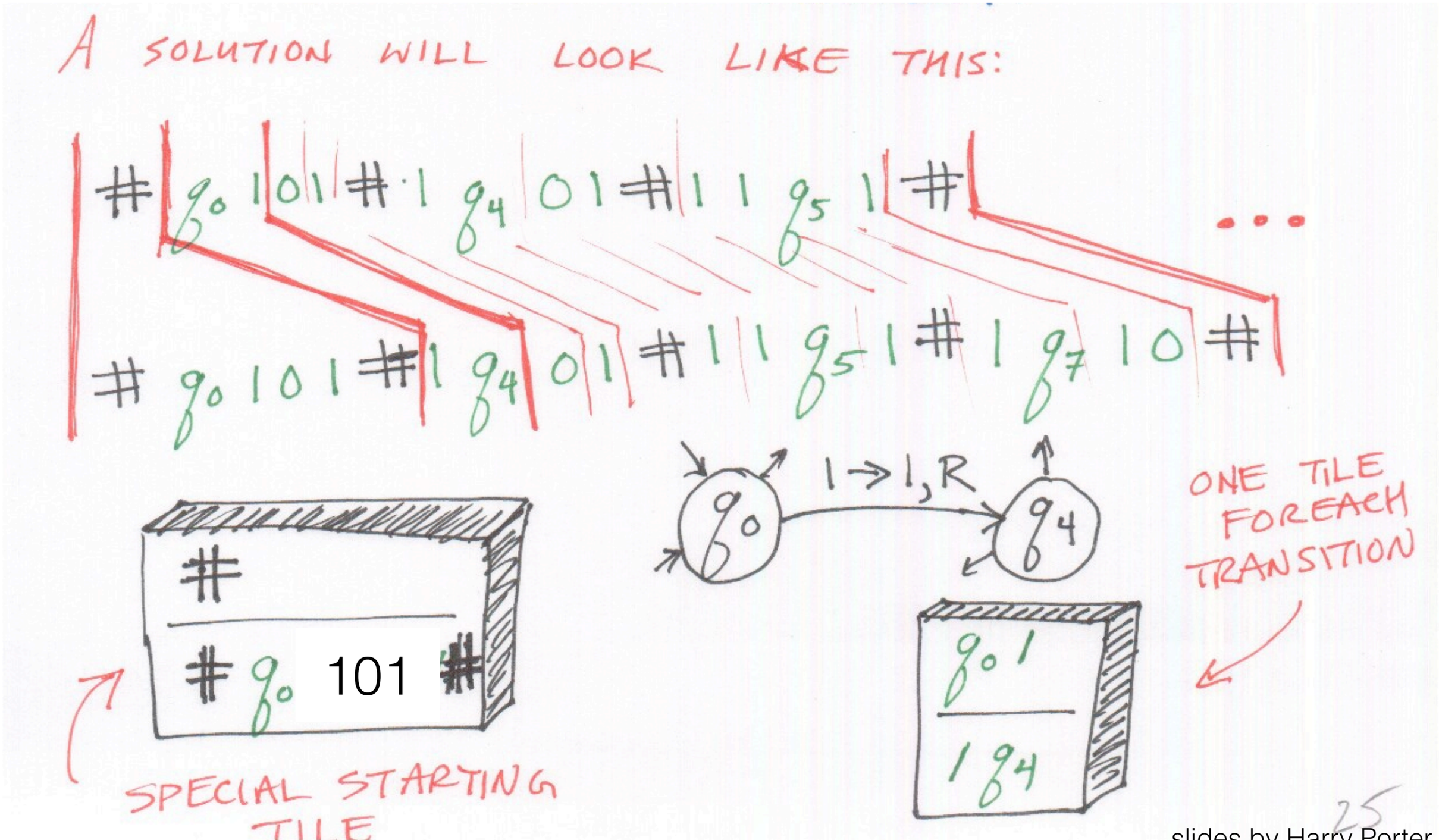
$q_A = q_{accept} = q_{acc}$

INITIAL STATE AT LEFT END OF TAPE

PCP is undecidable

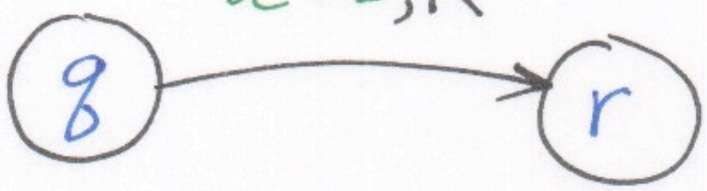
- Given input <M,w>
- Construct an input to PCP…
    - Create of collection of Tiles
- such that if you find a solution to PCP then you've found an accepting computation history

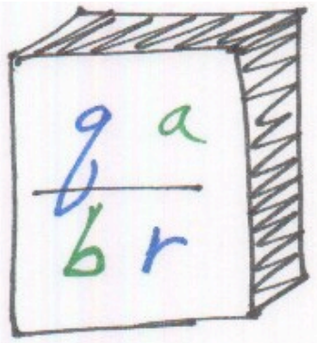
PCP is undecidable



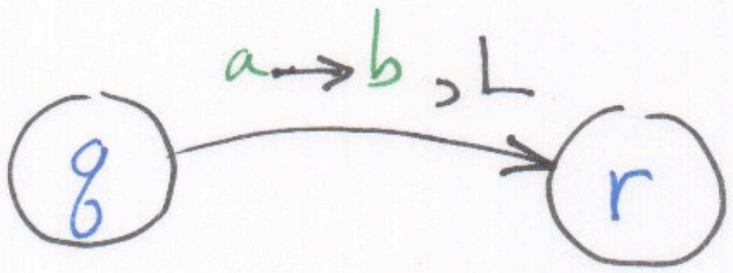
slides by Harry Porter PSU

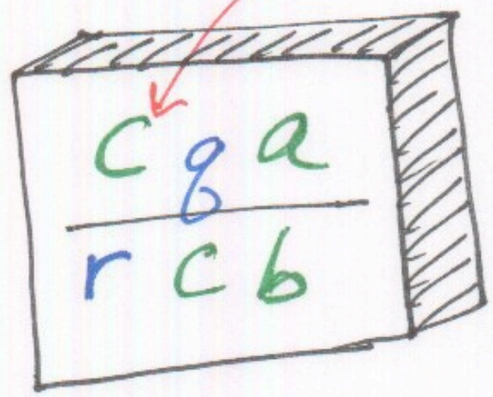
RIGHT MOVES:





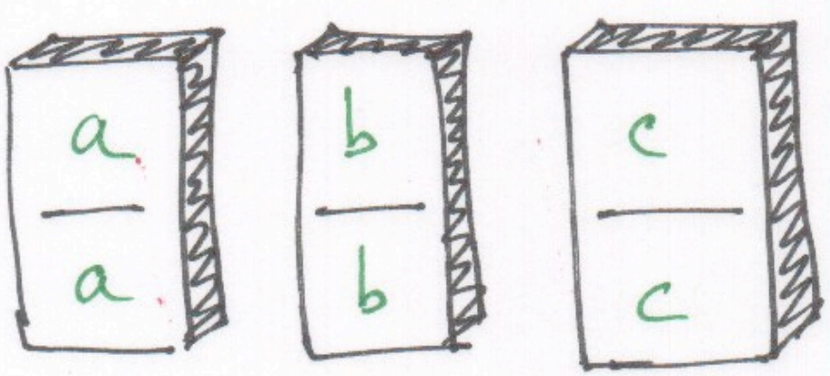
LEFT MOVES



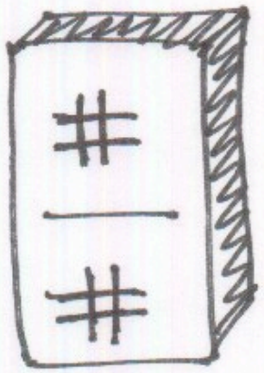


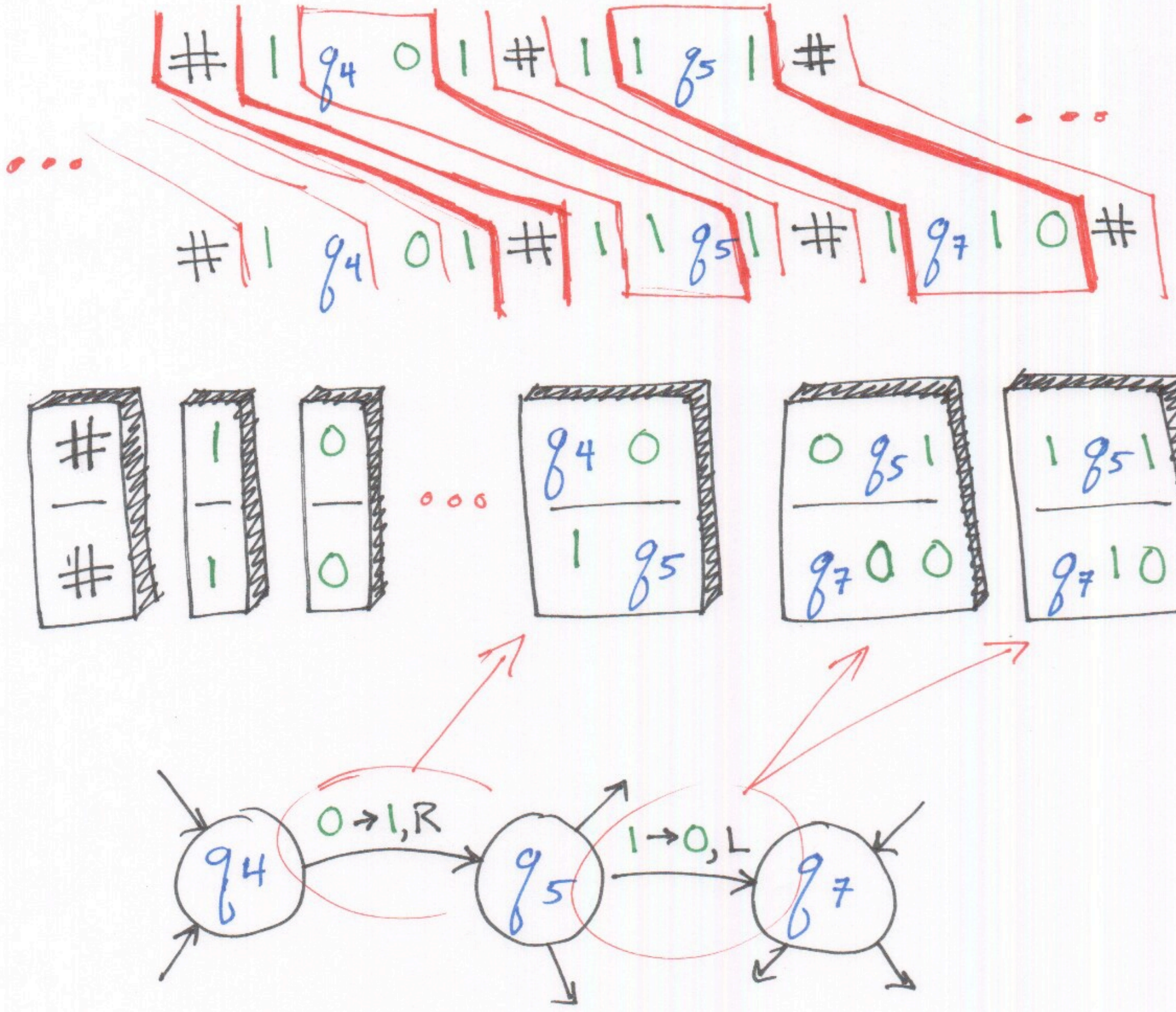
Add one tile like this for every $c \in \Gamma$

TILES TO "COPY" THE TAPE:



For every symbol in Γ

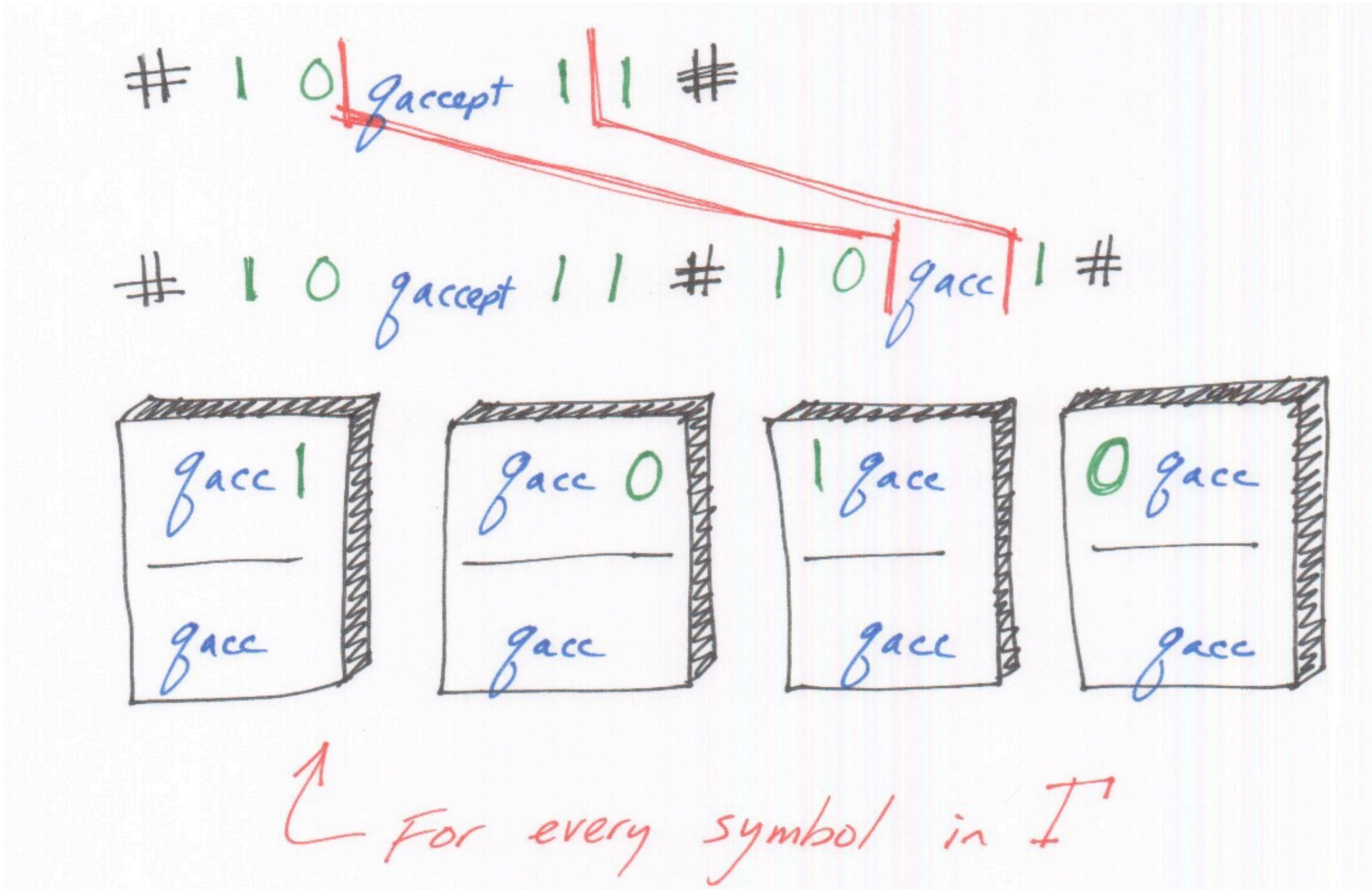




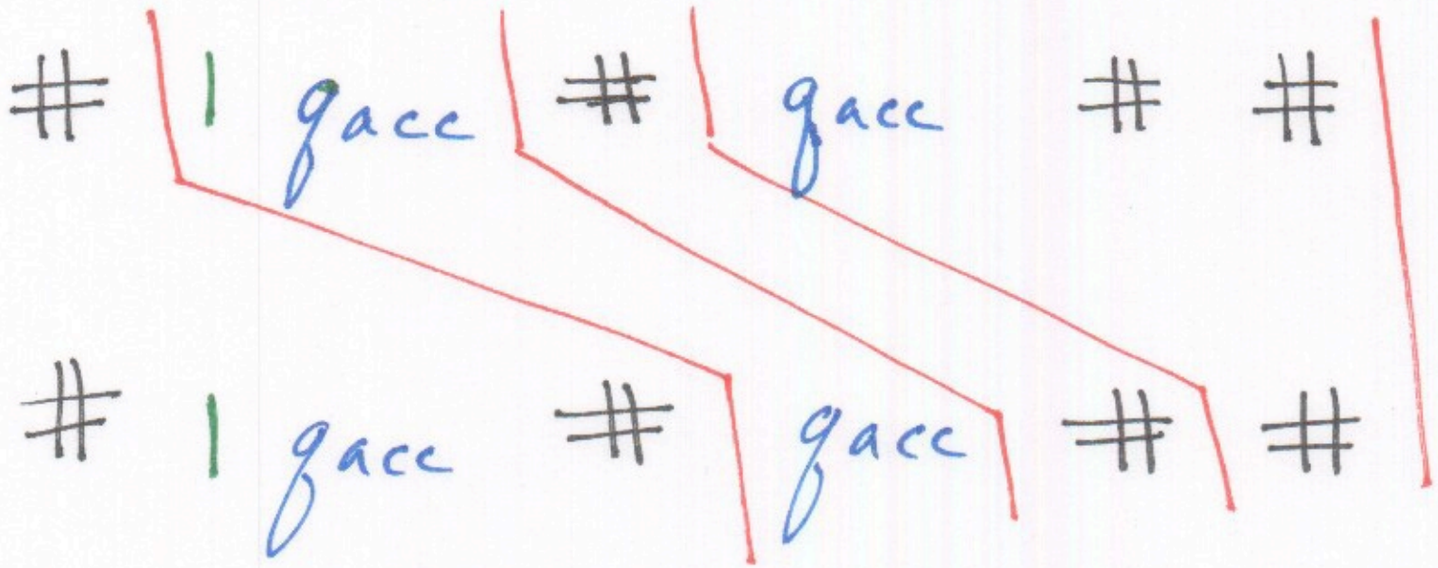
slides by Harry Porter PSU

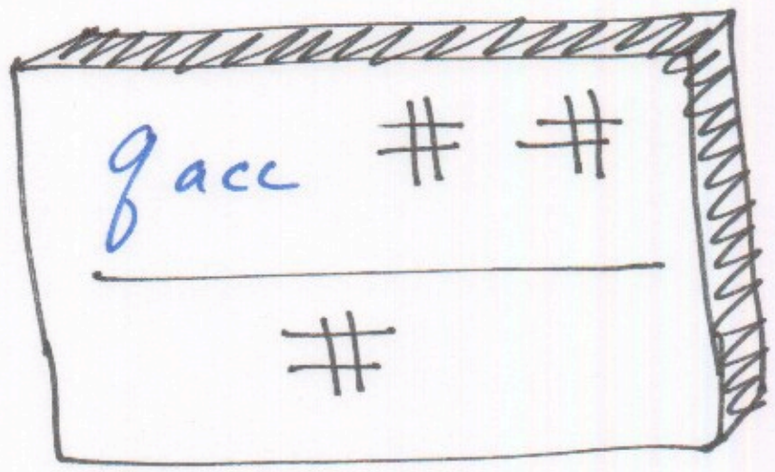
- How dow we accept?

- We will add special tiles to allow q_{accept} to "*eat*" the symbols on the tape



EVENTUALLY, NOTHING REMAINS
EXCEPT gacc.
ADD A TILE TO FINISH THE MATCH.





Review of Proof

- If you can find a solution to PCP then you have found an accepting computation history in which TM M accepts w

- If we can decide PCP then we can decide whether M accepts w

- We know A_{TM} is undecidable

- Therefore PCP is undecidable