

DECISION PROBLEMS AND DECIDABILITY

DECISION PROBLEM: A QUESTION (OVER MANY INSTANCES)
WITH A YES-NO ANSWER

- GIVEN GRAMMAR G , IS G AMBIGUOUS?
- GIVEN A TM M , DOES $L(M) = \Sigma^*$?
- GIVEN DFAs M_1, M_2 , IS $L(M_1) = L(M_2)$?
- GIVEN GRAPH G , IS G CONNECTED?

CAN BE REPRESENTED AS A SET (OR LANGUAGE)

$\{ \langle G \rangle : \text{"}\langle G \rangle\text{" IS A STRING ENCODING A GRAMMAR } G \text{ S.T. } G \text{ IS AMBIGUOUS} \}$

$\{ \langle M \rangle : \text{"}\langle M \rangle\text{" ENCODES A TM } M \text{ SUCH THAT } L(M) = \Sigma^* \}$

$\{ \langle M_1, M_2 \rangle : M_1, M_2 \text{ ARE DFAs, WITH } L(M_1) = L(M_2) \}$

$\{ \langle G \rangle : \text{"}\langle G \rangle\text{" IS A STRING ENCODING A CONNECTED GRAPH} \}$

WE SEEK AN ALGORITHM TO SOLVE THE PROBLEM
FOR ANY POSSIBLE INPUT. \rightarrow ALWAYS HALTS

A (DECISION) PROBLEM IS DECIDABLE

IF \exists ALGORITHM THAT \forall INSTANCES (INPUTS)
SAYS EITHER "YES" OR "NO" (AND IS CORRECT)

THUS A PROBLEM IS DECIDABLE

ASSOCIATED LANGUAGE IS RECURSIVE

NOTE

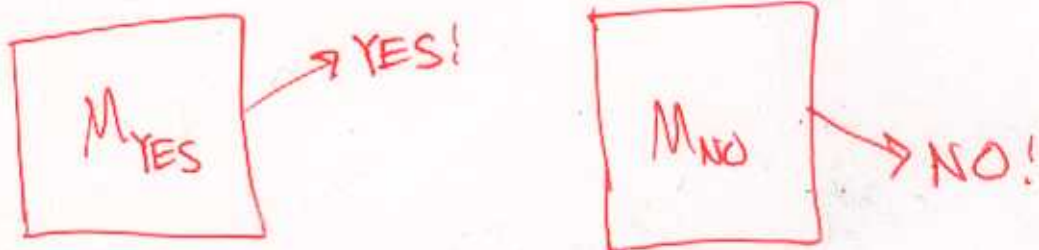
SINCE FINITE LANGUAGES ARE ALL RECURSIVE,
ANY DECISION PROBLEM WITH ONLY A FINITE
NUMBER OF INSTANCES IS DECIDABLE.

EXAMPLES, . . .

EXAMPLE 1

PROBLEM: \exists A SUBSTRING OF EXACTLY
83 CONSECUTIVE "7"'S IN π ?

"DECIDABLE"



ONE OF THESE INPUT-LESS ALGORITHMS
OUTPUTS THE CORRECT ANSWER.

DO BAD WE DON'T KNOW WHICH
ALGORITHM IS THE CORRECT ONE

MORAL: THIS IS NONSENSE. THE ORIGINAL PROBLEM
HAS NO "INSTANCES", IT ONLY ASKS A
SINGLE YES-NO QUESTION. WE ARE CONSIDERING
PROBLEMS THAT ARE QUESTIONS RANGING OVER
MANY POSSIBLE INPUTS.

NOT EVEN CLEAR WHAT LANGUAGE CORRESPONDS TO THE ABOVE "PROBLEM"

EXAMPLE 2

(LET'S TRY AGAIN)

PROBLEM: GIVEN n , \exists A SUBSTRING
OF EXACTLY n CONSECUTIVE "7"'S
IN DECIMAL EXPANSION OF π ?

CORRESPONDING LANGUAGE:

$\{n \in \mathbb{N} : \text{DECIMAL EXPANSION OF } \pi \text{ CONTAINS}$
THE SUBSTRING $a \underbrace{7777 \dots 7}_n b$

WHERE $a, b \neq 7$ }

IS THIS LANGUAGE RECURSIVE ???

\equiv CAN WE DECIDE THE ABOVE PROBLEM ???

EXAMPLE 3

Q: GIVEN n , \exists SUBSTRING OF $\geq n$
CONSECUTIVE "7"'S IN π ?

LANGUAGE:

$L = \{ n \in \mathbb{N} : \text{DECIMAL EXP OF } \pi \text{ CONTAINS}$
SUBSTRING $\underbrace{7777 \dots 7}_n \}$

Q: IS THIS DECIDABLE (RECURSIVE) ?

A: IT IS REGULAR !!

L IS EITHER ALL OF \mathbb{N} , OR ELSE $\exists n_0 \ L = \{1, 2, 3, \dots, n_0\}$

CODING OF TMs

- SHOW HOW TO ASSOCIATE EACH TM WITH A NUMBER IN A NICE WAY. THEN CAN TREAT PROGRAMS (TMs) AS DATA (NUMBERS)

LEMMA [HU THM 7.10]

IF $L \subseteq \{0,1\}^*$ IS ACCEPTED BY SOME TM
THEN \exists TM WITH $\Sigma = \{0,1\}$, $\Gamma = \{0,1,B\}$
THAT ACCEPTS L

NOW CODE ANY TM WITH $\Sigma = \{0,1\}$, $\Gamma = \{0,1,B\}$
AS BINARY STRING.

WLOG ASSUME

- STATES NUMBERED $1, 2, 3, \dots, k$ FOR SOME k
 - q_1 UNIQUE INITIAL STATE
 - q_2 UNIQUE HALT/ACCEPT STATE
 - q_3 UNIQUE HALT/REJECT STATE
- MAY NOT BE REACHED IF TM IN LOOP.

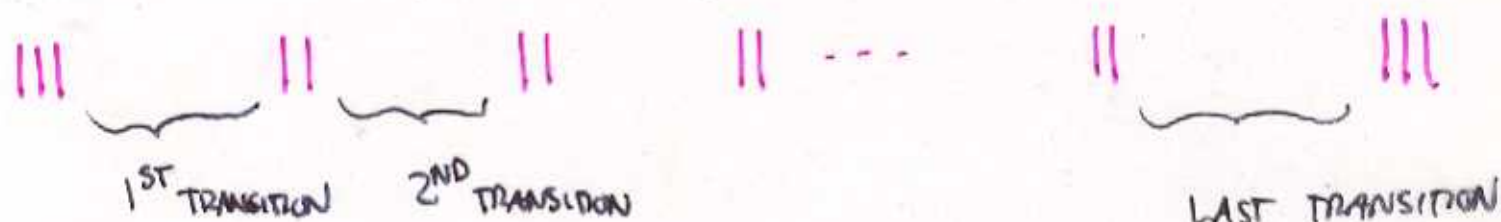
TO CODE A TM, WE LIST ITS TRANSITIONS
(ITS STATES ARE IMPLICITLY LISTED, AND WE NEED
NOT SPECIFY INITIAL STATE, HALT STATES, BY OUR CONVENTION)

TRANSITIONS ARE LISTED IN THE FOLLOWING ORDER,
OMITTING ANY THAT ARE UNDEFINED:

$$\delta(q_1, 0), \delta(q_1, 1), \delta(q_1, B), \delta(q_2, 0), \delta(q_2, 1), \delta(q_2, B) \dots$$

$$\dots \delta(q_k, 0), \delta(q_k, 1), \delta(q_k, B)$$

THE ENCODING:



WHERE EACH TRANSITION

$$\delta(q_i, \begin{Bmatrix} 0 \\ 1 \\ B \end{Bmatrix}) = (q_j, \begin{Bmatrix} 0 \\ 1 \\ B \end{Bmatrix}, \begin{Bmatrix} L \\ R \\ S \end{Bmatrix})$$

IS ENCODED BY

$$0^i \mid \begin{Bmatrix} 0 \\ 00 \\ 000 \end{Bmatrix} \mid 0^j \mid \begin{Bmatrix} 0 \\ 00 \\ 000 \end{Bmatrix} \mid \begin{Bmatrix} 0 \\ 00 \\ 000 \end{Bmatrix}$$

THUS $\delta(q_3, 1) = (q_2, 0, S)$ BECOMES...

$$000 \mid 00 \mid 00 \mid 0 \mid 000$$

• CLEARLY EVERY TM \rightarrow SOME $n \in \mathbb{N}$

• VIEW IT AS ONE-TO-ONE MAPPING:

IF $n \in \mathbb{N}$ DOESN'T SYNTACTICALLY
CORRESPOND TO ANY TM DESCRIPTION,

WE'LL LET n REPRESENT THE

"NULL TM" THAT ACCEPTS \emptyset .
BY LOOPING ON ALL INPUTS

• $\mathbb{N} =$ TURING-MACHINE DESCRIPTIONS

• LET $\langle M \rangle = n$ SUCH THAT n ENCODES M .

• WE WRITE M_i TO INDICATE i^{TH} TM,
THAT IS, M_i IS THE M SUCH THAT
 $\langle M \rangle = i$

A NON RECURSIVELY-ENUMERABLE LANGUAGE

$$L_d = \text{"DIAGONAL LANGUAGE"} \\ = \{i : i \notin L(M_i)\}$$

INPUTS \rightarrow (STRINGS CORR TO BINARY #'s)

TURING MACHINES \downarrow

	1	2	3	4	5	6	7	...
1	N	N	Y	Y	N	N	Y	
2	N	Y	Y	N	N	Y	N	$\leftarrow 7 \notin L(M_2)$
3	N	Y	N	Y	N	N	Y	
4	Y	N	N	Y	Y	Y	N	
5	Y	Y	Y	Y	N	Y	N	
6								
...								

L_d IS NOT REC. ENUM.

IF IT WERE, THEN $L_d = L(M_j)$ FOR SOME $j \in \mathbb{N}$

BUT $j \in L_d \Leftrightarrow j \notin L(M_j)$

BUT DOESN'T FOLLOWING ACCEPT L_d ?

M_d

ON INPUT i

GENERATE TM M_i DESCRIPTION

RUN M_i ON INPUT i

OUTPUT YES IF $M_i(i)$ OUTPUTS NO

OUTPUT NO IF $M_i(i)$ OUTPUTS YES

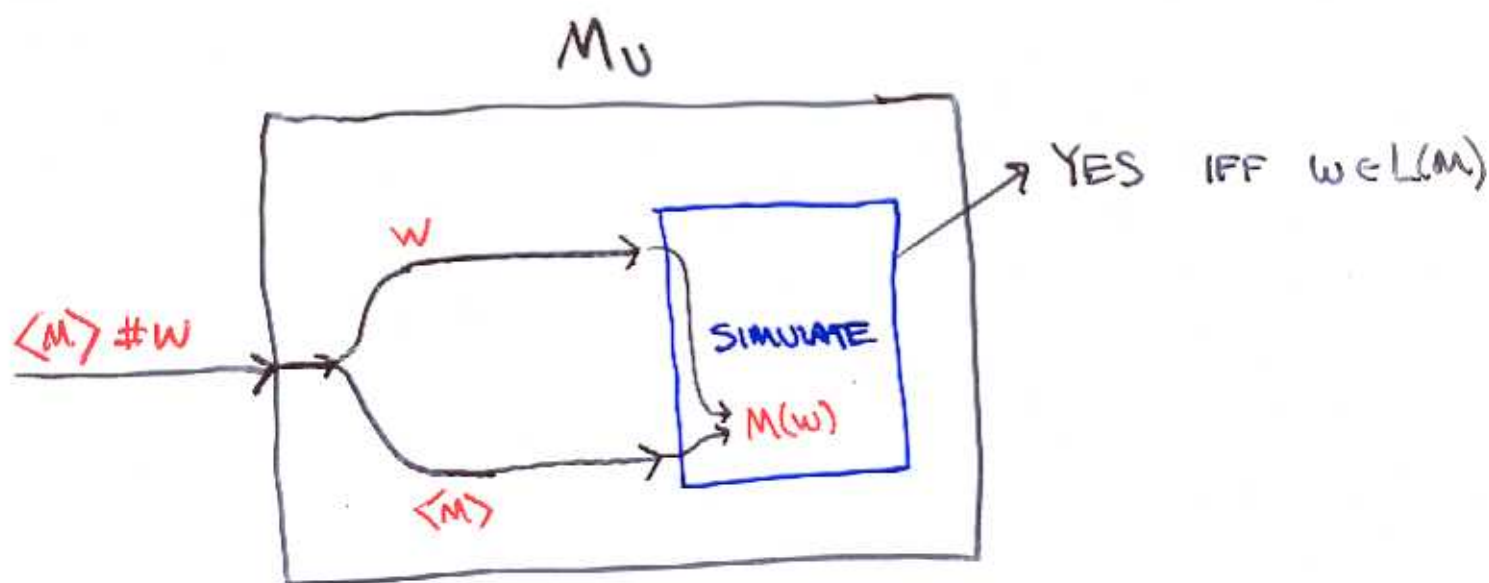
DOES M_d ACCEPT L_d ?

WHY NOT ?

UNIVERSAL LANGUAGE / UNIVERSAL TM

$L_U = \{ \langle M \rangle \# w : M \text{ ACCEPTS } w \}$ IS R.E.

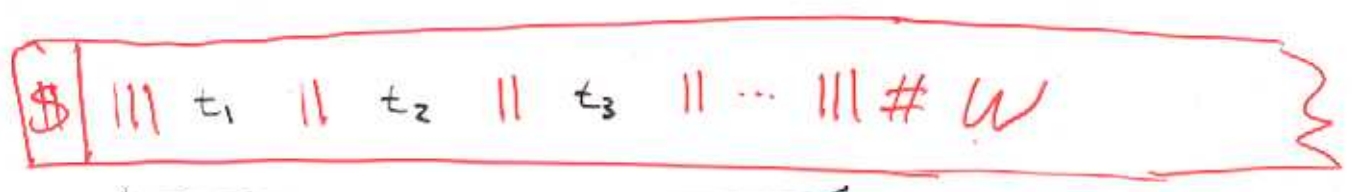
WE DESCRIBE M_U THAT ON INPUT $\langle M \rangle \# w$ SIMULATES $M(w)$ AND ACCEPTS IF M DOES



- SIMILAR TO SINGLE TM THAT SIMULATES AN ARBITRARY RAM PROGRAM
- \exists SINGLE TM M_U THAT FUNCTIONS AS STORED PGM COMPUTER. M_U IS THE EMBODIMENT OF "COMPUTER"

HOW M_U WORKS (3 TAPES)

TAPE 1



$\langle M \rangle =$ CODE FOR M
 $t_i =$ CODE FOR TRANSITION i

TAPE 1 NEVER CHANGES

① USING TAPE 3 AS WORKSPACE, M_U CHECKS THAT $\langle M \rangle$ IS VALID TM CODE.

THUS IT CHECKS, FOR EXAMPLE, THAT

- SUBSTRING $||0^i10^j|$ DOESN'T APPEAR TWICE (INDICATING NONDETERMINISM)
- NO FOUR 1's
- THREE 1's AT BEGINNING, END ONLY
- APPROPRIATE # OF 0's BETWEEN 1's IN TRANSITION CODES

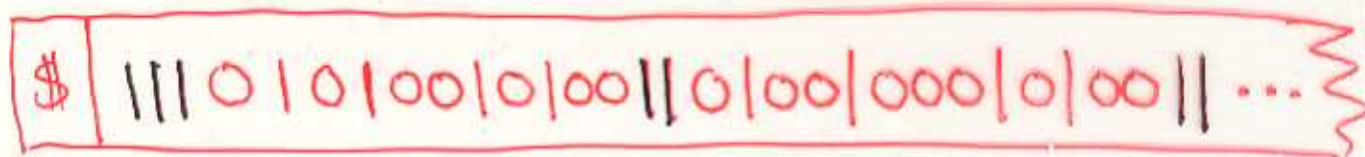
$||0000|0|00000|0000|...$

$\delta(q_4, 0) = (q_5,$ \uparrow NOT A VALID CHAR. TO WRITE

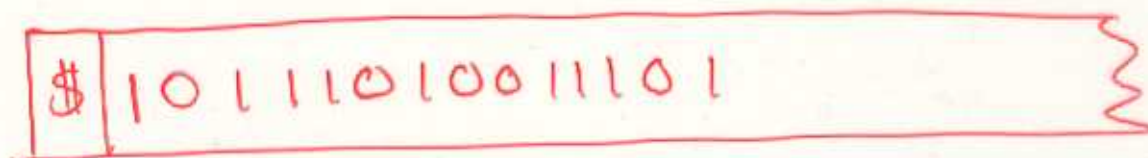
② ASSUMING $\langle M \rangle$ IS OK, M_U COPIES w TO TAPE 2

③ M_U WRITES "0" ON TAPE 3. THIS INDICATES M IS IN INITIAL STATE q_1 . (TAPE 3 HOLDS 0^i INDICATING STATE IS q_j) IF EVER TAPE 3 HOLDS $00 = q_2 = \text{HALT/ACCEPT}$ THEN M_U HALTS AND ACCEPTS

TAPE 1



TAPE 2



w

← CURRENT CONTENTS OF M 'S TAPE

TAPE 3

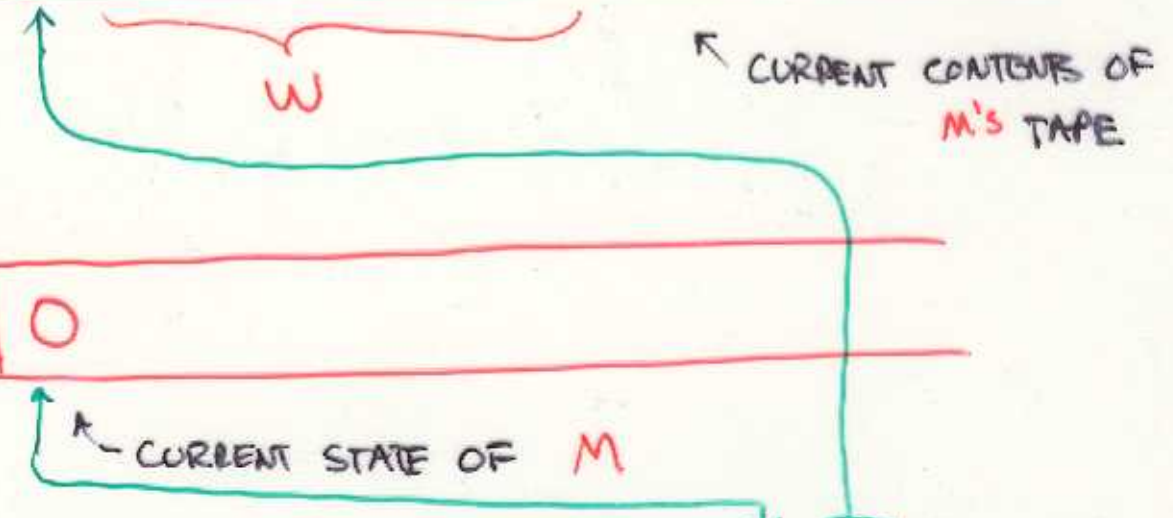
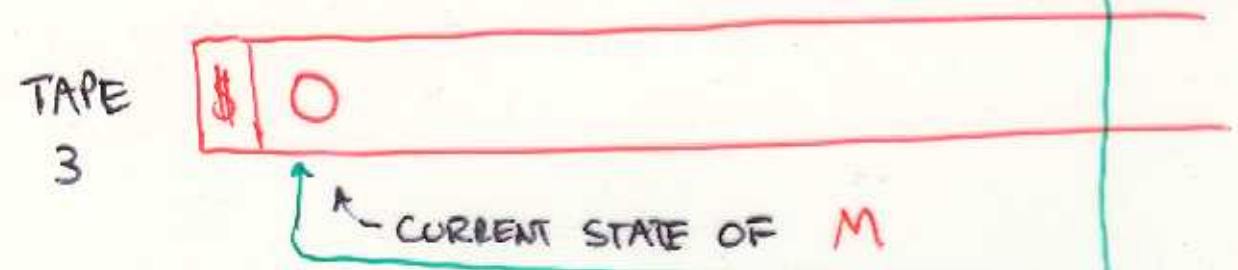
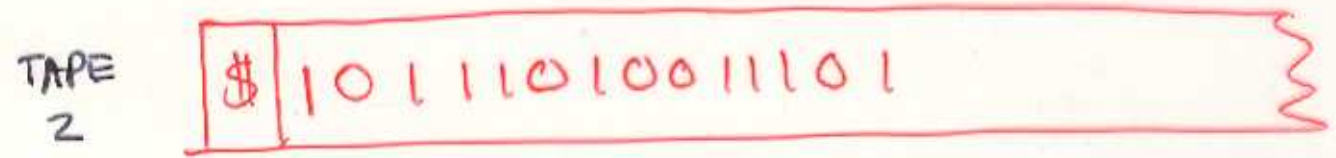


← CURRENT STATE OF M

④ IF TAPE 3 HOLDS 0^i AND HEAD ON TAPE 2 IS SCANNING SYMBOL "1" THEN M_U LOOKS FOR $10^i|00|...$ ON TAPE 1. IF NOT FOUND, THEN HALT/REJECT. ELSE M_U FINDS $10^i|00|0^j|0|00|$ (FOR EXAMPLE) SO M_U WRITES 0^j ON TAPE 3, REPLACES 1 WITH 0 ON TAPE 2, AND MOVES TAPE 2 HEAD ρ RIGHT

② ASSUMING $\langle M \rangle$ IS OK, M_U COPIES w TO TAPE 2

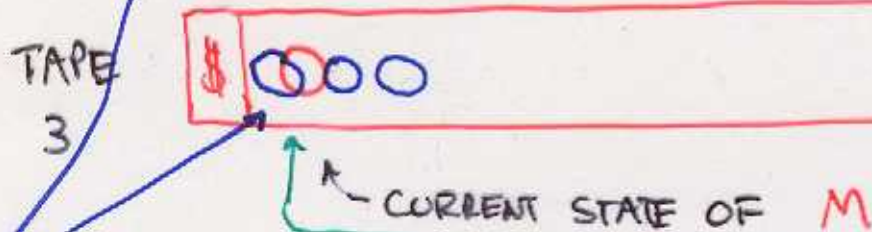
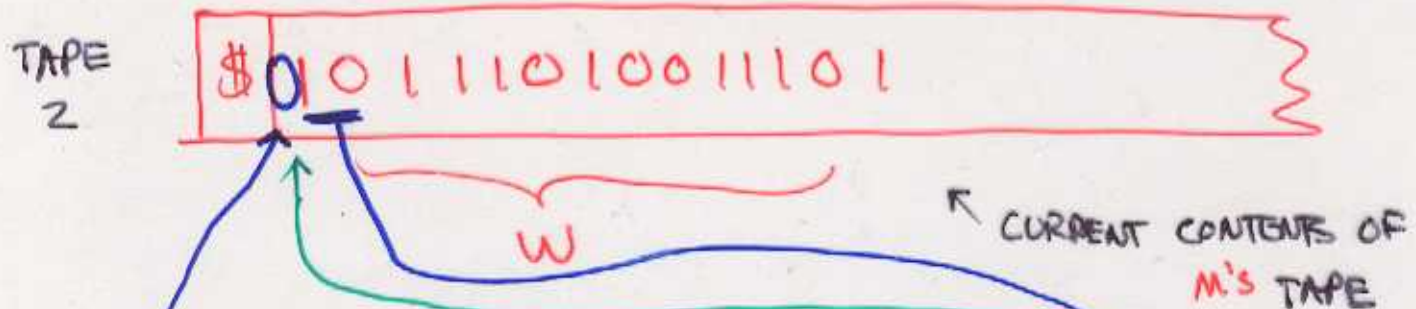
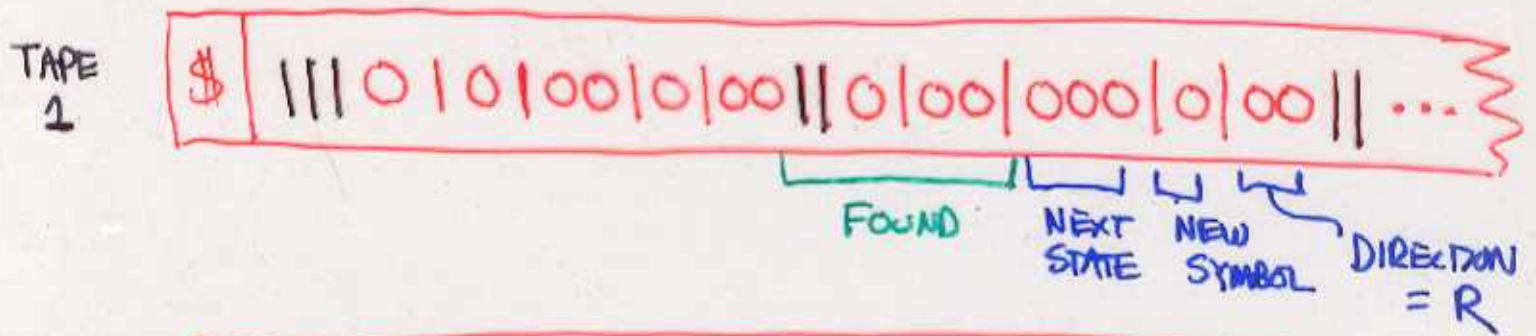
③ M_U WRITES "0" ON TAPE 3. THIS INDICATES M IS IN INITIAL STATE q_1 . (TAPE 3 HOLDS 0^j INDICATING STATE IS q_j) IF EVER TAPE 3 HOLDS $00 = q_2 = \text{HALT/ACCEPT}$ THEN M_U HALTS AND ACCEPTS



EXAMPLE: M_U LOOKS FOR $|| 0 | 00 |$ ON TAPE 1

② ASSUMING $\langle M \rangle$ IS OK, M_U COPIES w TO TAPE 2

③ M_U WRITES "0" ON TAPE 3. THIS INDICATES M IS IN INITIAL STATE q_1 . (TAPE 3 HOLDS q_j INDICATING STATE IS q_j) IF EVER TAPE 3 HOLDS $00 = q_2 = \text{HALT/ACCEPT}$ THEN M_U HALTS AND ACCEPTS



EXAMPLE: M_U LOOKS FOR $||0|00|$ ON TAPE 1

NOW COPY NEXT STATE TO TAPE 3

WRITE NEW SYMBOL ON TAPE 2

MOVE TAPE 2 HEAD TO RIGHT ONE CELL


⑤ MOVE TAPE 1, 3 HEADS TO $\$$. CHECK FOR ACCEPT. GO TO ④

THEOREM L_U IS NOT RECURSIVE

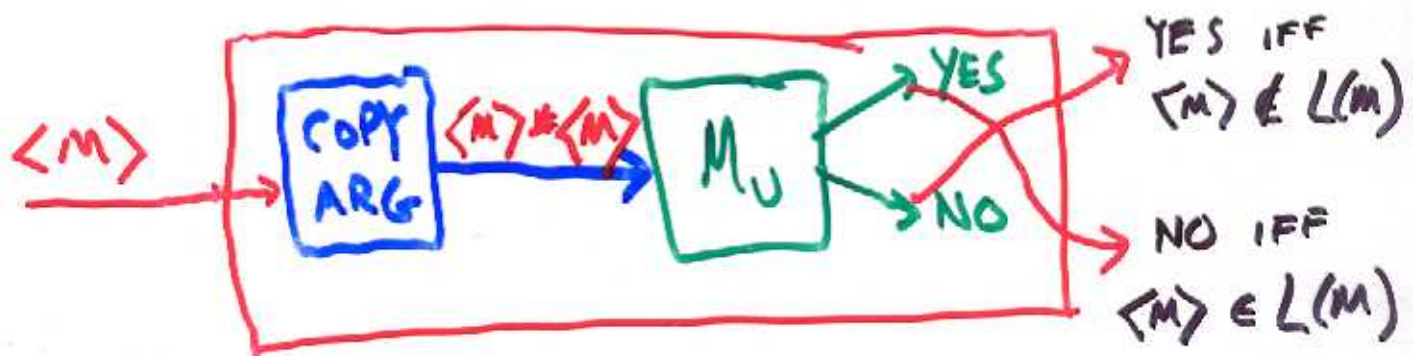
THUS IT IS UNDECIDABLE GIVEN A TM M AND INPUT w TO DETERMINE WHETHER OR NOT M ACCEPTS w

PROOF

ASSUME TO CONTRARY THAT M_U DECIDES L_U

THUS $\langle M \rangle \# w \rightarrow$  $\begin{cases} \text{YES IFF } w \in L(M) \\ \text{NO IFF } w \notin L(M) \end{cases}$

CONSIDER M'_U EXISTS, ALWAYS HALTS, BECAUSE M_U DOES



\rightarrow

"REDUCTIONS"

IF WE CAN USE AN ALGORITHM FOR DECIDING L_2 TO OBTAIN AN ALG. FOR DECIDING L_1

THEN WE SAY: L_1 REDUCES TO L_2

WRITE: $L_1 \leq L_2$

IF $L_1 \leq L_2$ AND L_2 IS RECURSIVE

THEN L_1 IS RECURSIVE.

(SO L_1 IS "NO HARDER" THAN L_2 , HENCE THE " \leq ")

IF $L_1 \leq L_2$ AND L_1 IS NOT RECURSIVE

THEN L_2 CANNOT BE RECURSIVE

" \leq " HAS A WELL DEFINED TECHNICAL MEANING. WE USE REDUCTIONS INFORMALLY HERE. LATER WE CAREFULLY DEFINE " \leq " IN CONTEXT OF NP-COMPLETENESS

ANOTHER NON RECURSIVE LANGUAGE:

THE HALTING PROBLEM IS UNDECIDABLE

PRBLM: GIVEN M, w , DECIDE IF M HALTS ON INPUT w

LANGUAGE: $L_{\text{HALT}} = \{ \langle M \rangle \# w : M \text{ HALTS ON INPUT } w \}$

ALTERNATIVE $K = \{ i : M_i(i) \text{ HALTS} \}$

K IS ALSO CALLED THE HALTING PROBLEM.

THEOREM: L_{HALT} IS NOT RECURSIVE

THUS \nexists ALGORITHM, GIVEN A PROGRAM (TM)

AND INPUT (w), THAT DECIDES WHETHER

OR NOT THE PROGRAM WILL GET INTO AN

∞ LOOP WHEN RUN ON INPUT w



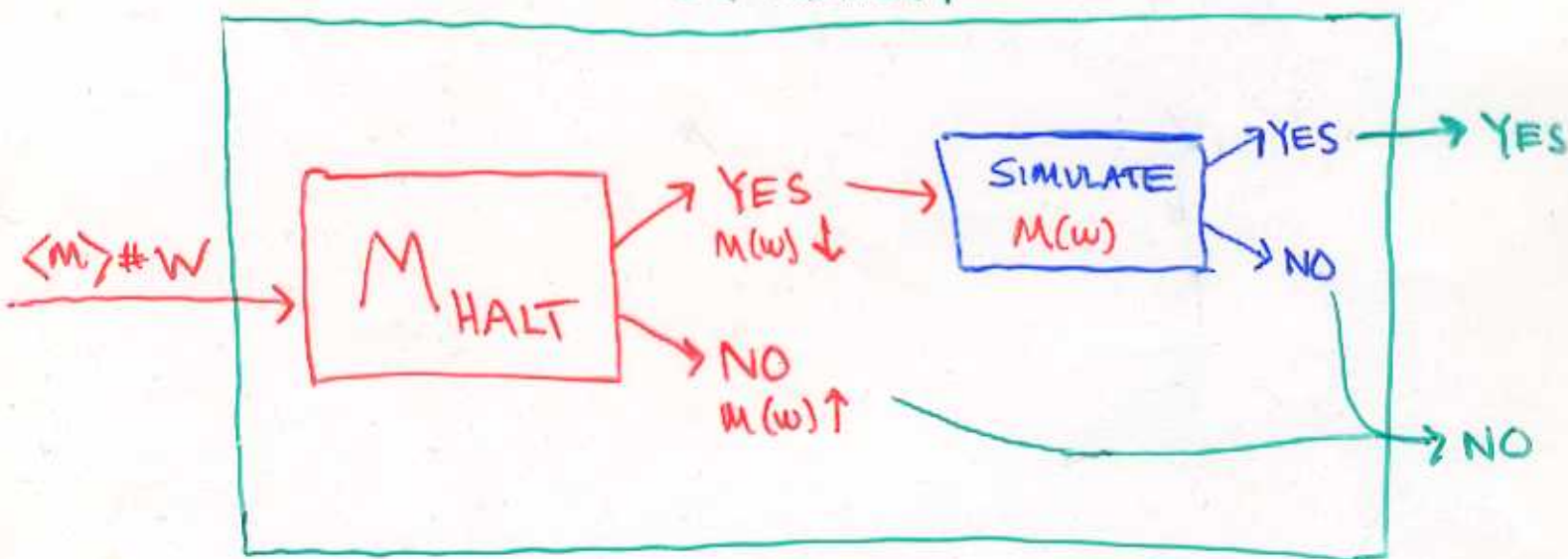
PROOF SHOW $L_U \cong L_{HALT}$

AND SINCE \nexists ALG FOR L_U , \nexists ALG FOR L_{HALT}

ASSUME L_{HALT} IS RECURSIVE, WITNESSED

BY

M_U (ALWAYS HALT)



M_U (ALWAYS HALT) DETERMINES $\forall \langle m \rangle, w$
WHETHER OR NOT M ACCEPTS w

ALWAYS HALTS BECAUSE IT ONLY SIMULATES
 $M(w)$ IF THE COMPUTATION IS GUARANTEED
TO HALT (BY ASSUMPTION THAT M_{HALT} IS CORRECT)

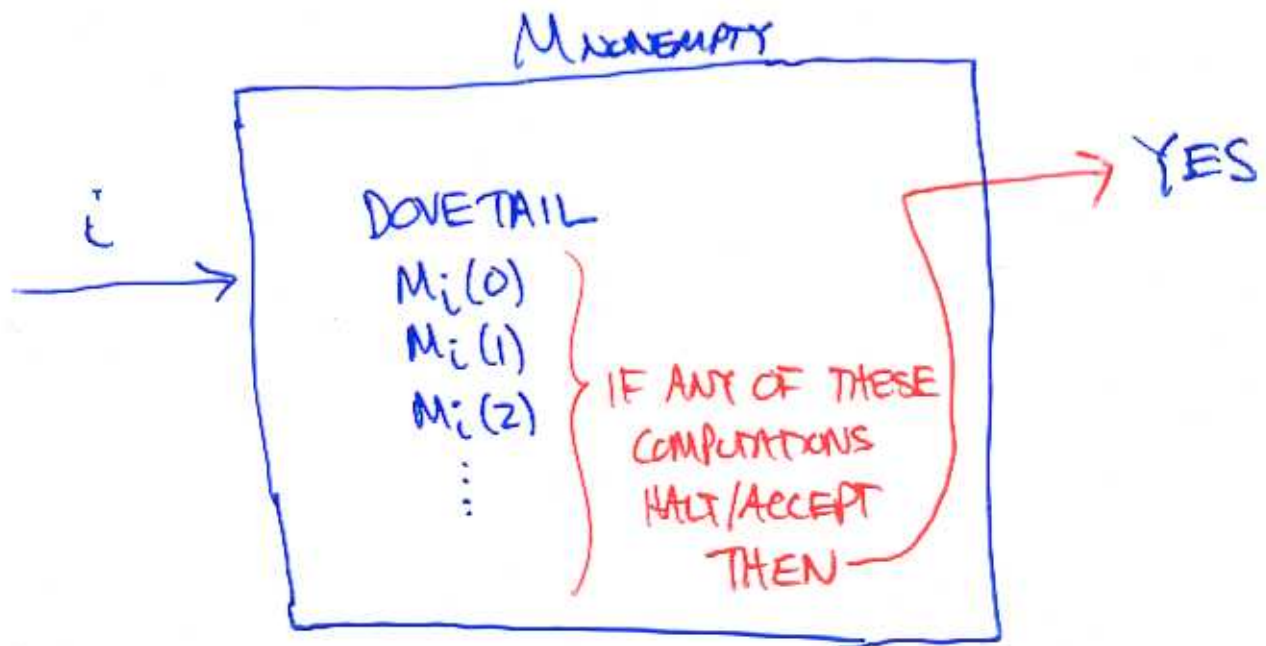
CONTRADICTS THAT L_U IS NOT RECURSIVE

ANOTHER EXAMPLE

$$L_{\text{NONEMPTY}} = \{ i : L(M_i) \neq \emptyset \}$$

IS R.E. BUT NOT RECURSIVE

L_{NONEMPTY} IS R.E.



M_{NONEMPTY} ACCEPTS i IFF $\exists w$ THAT M_i ACCEPTS
IFF $L(M_i) \neq \emptyset$

THEOREM

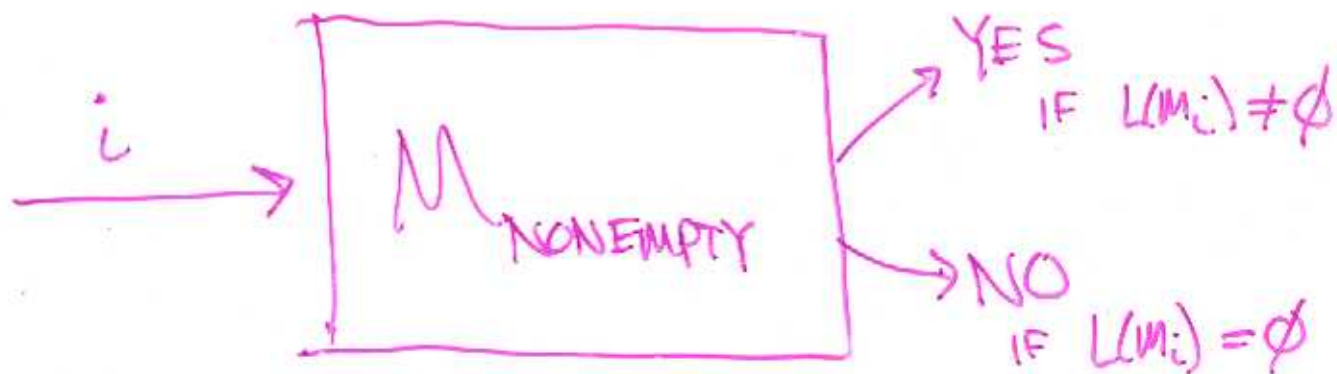
L_{NONEMPTY} IS NOT RECURSIVE

☹️ CAN'T DETERMINE, GIVEN A PROGRAM, WHETHER OR NOT THE PROGRAM ACCEPTS ANY INPUT

PROOF TECHNIQUE

SHOW THAT $L_U \leq L_{\text{NONEMPTY}}$, THUS IF WE HAD AN ALGORITHM TO DECIDE GIVEN i WHETHER OR NOT $L(M_i) = \emptyset$, THEN WE COULD USE IT TO OBTAIN AN ALGORITHM TO DECIDE GIVEN $\langle M \rangle \# w$ WHETHER OR NOT $w \in L(M)$.

ASSUME (TO THE CONTRARY) THAT THERE EXISTS



IDEA: CREATE A TRANSFORMATION (AN ALGORITHM) A
SUCH THAT



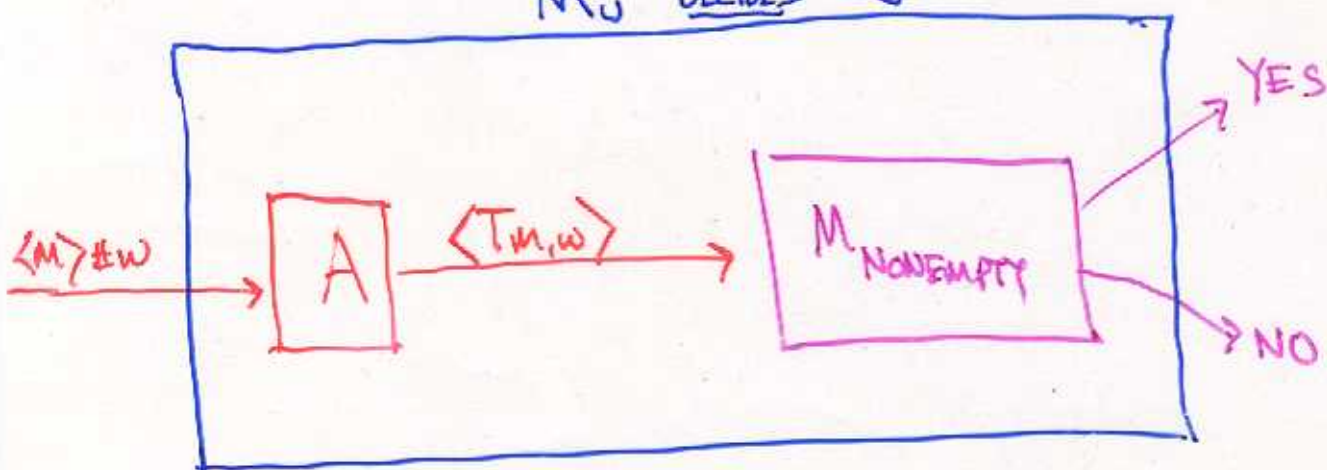
GIVEN $\langle M \rangle \# w$ A OUTPUTS CODING OF A

TM $T_{M,w}$ (THAT DEPENDS ON M, w)

SUCH THAT $L(T_{M,w}) \neq \emptyset \iff M$ ACCEPTS w

I.E., $\langle T_{M,w} \rangle \in L_{\text{NONEMPTY}} \iff \langle M \rangle \# w \in L_U$

M_U DECIDES L_U



M_U ACCEPTS $\langle M \rangle \# w$ IFF M_{NONEMPTY} ACCEPTS $\langle T_{M,w} \rangle$ IFF $\langle M \rangle \# w \in L_U$

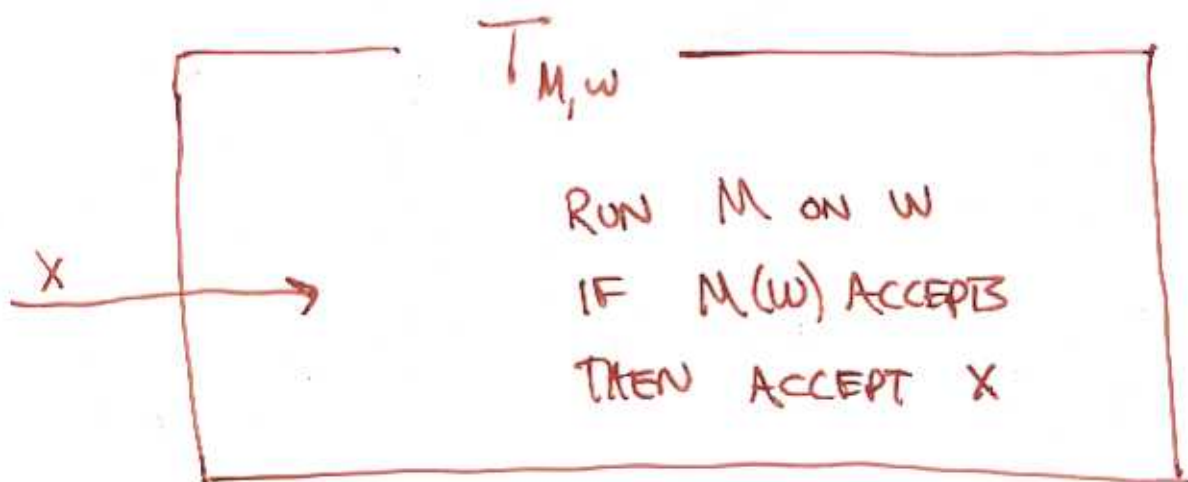
M_U ALWAYS HALTS, SINCE BY ASSUMPTION M_{NONEMPTY} DOES, AND BY CONSTR. A DOES.

DESCRIPTION OF TRANSFORMATION ALGORITHM A

INPUT TO A: $\langle M \rangle \# w$

OUTPUT OF A: $\langle T_{M,w} \rangle$

WHERE $T_{M,w}$ IS THE FOLLOWING TM:



$T_{M,w}$ IGNORES ITS INPUT AND ACCEPTS ONLY IF M ACCEPTS w

- IF M ACCEPTS w , $L(T_{M,w}) = \Sigma^*$
- IF M DOESN'T ACCEPT w , $L(T_{M,w}) = \emptyset$

THUS $L(T_{M,w}) \neq \emptyset \iff M$ ACCEPTS w AS DESIRED

... BUT WHAT DOES A LOOK LIKE ???

RECALL M_U (UNIVERSAL TM)

THAT RECEIVES $\langle M \rangle \# W$ AND SIMULATES M ON W

A OUTPUTS DESCRIPTION OF $T_{M,W}$ THAT RUNS M_U ON
INPUT $\langle M \rangle \# W$

THIS $T_{M,W}$ HAS SUBROUTINE TO WRITE $\langle M \rangle \# W$
ON TAPE (WRITING OVER INPUT)

CALLS M_U

GIVEN $\langle M \rangle \# W$, THE TM CODE NEEDED
TO WRITE $\langle M \rangle \# W$ IS OBVIOUS
SO A CAN CONSTRUCT THIS PORTION
OF CODE OF $T_{M,W}$ EASILY

THE TM CODE FOR M_U IS SOME NUMBER i
 A HAS i STORED AS PART OF ITS DATA.
(IN A BIG "PRINT" STATE)

THIS ON INPUT $\langle M \rangle \# W$, A PRINTS:

||| || |||

TM TRANSITIONS TO PRINT
OUT $\langle M \rangle \# W$ ON TAPE

$i = \langle M_U \rangle$

THUS PROBLEM OF DETERMINING, GIVEN M ,
WHETHER OR NOT $L(M) \neq \emptyset$ IS UNDECIDABLE.

COROLLARY 1 CANT DECIDE IF $L(M) = \emptyset$ EITHER,
THUS

$$L_{\text{EMPTY}} = \{i : L(M_i) = \emptyset\}$$

IS NOT RECURSIVE

COROLLARY 2 L_{EMPTY} IS NOT REC ENUM. EITHER.

PROOF: IF IT WERE, THEN BOTH

L_{EMPTY} , L_{NONEMPTY} WOULD BE R.F.

AND THUS BOTH WOULD BE RECURSIVE

SHOW L_{EMPTY} IS NOT REC. ENUM. DIRECTLY.

WE SHOW $L_d \leq L_{EMPTY}$, THUS A TM ACCEPTING L_{EMPTY} CAN BE USED TO OBTAIN A TM ACCEPTING L_d . (NEITHER TM NEED HALT ON REJECTING) BUT L_d IS KNOWN TO BE NON REC ENUM.

RECALL $L_d = \{i : i \notin L(M_i)\}$

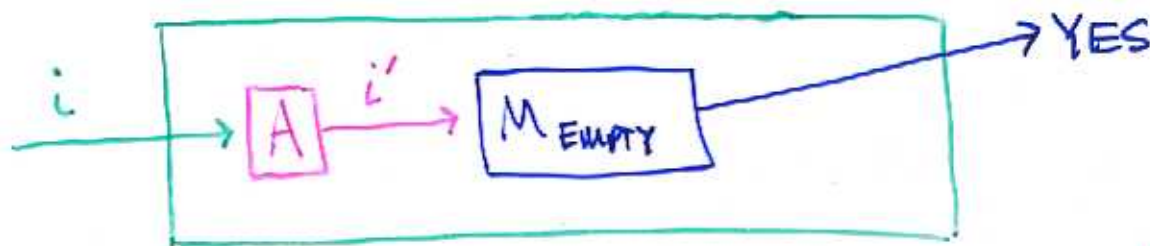
WE CONSTRUCT ALGORITHM A (ALWAYS HALTS) SUCH THAT



WHERE $i \notin L(M_i) \iff L(M_{i'}) = \emptyset$

I.E., $i \in L_d \iff i' \in L_{EMPTY}$

THUS IF M_{EMPTY} ACCEPTS L_{EMPTY} , WE HAVE



M_d ACCEPTS L_d , A CONTRADICTION

M_{EMPTY} CAN'T EXIST!

WE NEED:



$$i \notin L(M_i) \iff L(M_{i'}) = \emptyset$$

- RATHER THAN EXPLICITLY CONSTRUCTING A
WE JUST SHOW WHAT $M_{i'}$ IS IN TERMS OF M_i
- BECAUSE $M_{i'}$ WILL DEPEND ON M_i "IN THE SAME WAY"
REGARDLESS OF i , AND THE WAY IN WHICH IT DEPENDS
IS RATHER STRAIGHTFORWARD, IT IS POSSIBLE TO CONSTRUCT
 A THAT REALIZES THE TRANSFORMATION
- LEFT AS EXERCISE.

M_i'

"HARDCODED DATA": VALUE i

• RUN $M_i(i)$ FOR $|x|$ STEPS

• IF $M_i(i)$ HALTS AND ACCEPTS IN $|x|$ STEPS
THEN ACCEPT x
ELSE REJECT x

PROOF THAT $i \notin L(M_i) \iff L(M_i') = \emptyset$

• IF $i \notin L(M_i)$ THEN NO MATTER HOW MANY STEPS WE RUN $M_i(i)$, IT WILL NEVER ACCEPT.

SO EVERY x IS REJECTED AND $L(M_i') = \emptyset$

• IF $i \in L(M_i)$ THEN $\exists n$ $M_i(i)$ ACCEPTS IN n MOVES.

THUS $\forall |x| > n$ $M_i'(x)$ WILL RUN $M_i(i)$
FOR ENOUGH STEPS TO SEE THAT $M_i(i)$ ACCEPTS,

SO M_i' ACCEPTS x AND THUS $L(M_i') \neq \emptyset$

ONE MORE EXAMPLE REDUCTION

SHOW THAT THE PROBLEM OF DETERMINING WHETHER OR NOT, GIVEN M , $|L(M_i)| = 17$, IS UNDECIDABLE.

I.E. SHOW

$L = \{i : |L(M_i)| = 17\}$ IS NOT RECURSIVE.

METHOD: $L_U \leq L$

(SHOW HOW AN ALG FOR L GIVES AN ALG FOR L_U)

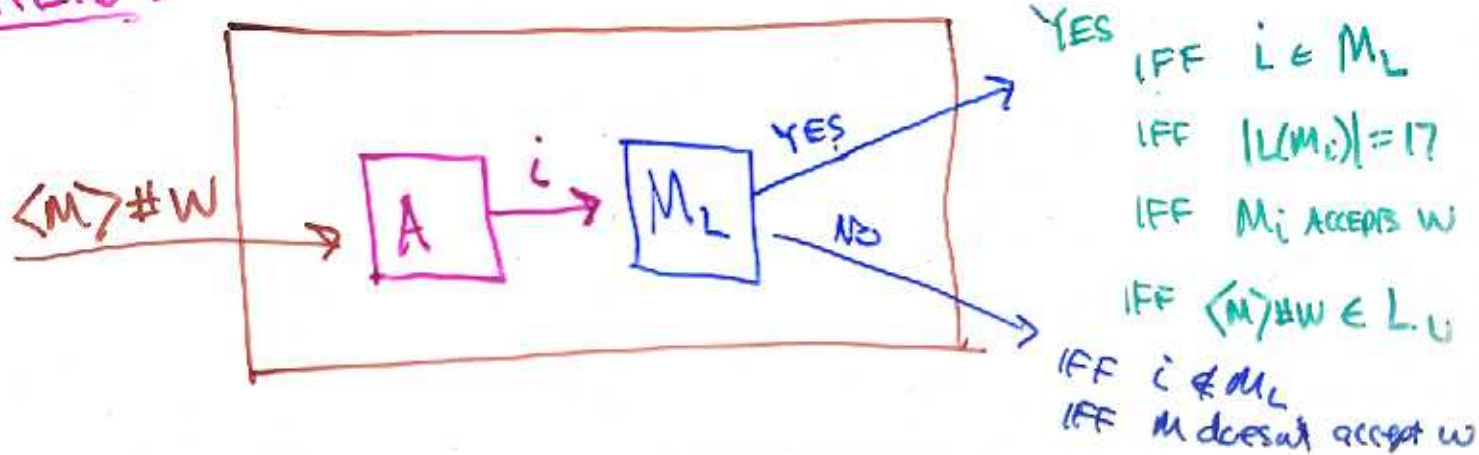
DEVISE TRANSFORMATION A SUCH THAT



$|L(M_i)| = 17$ IF M ACCEPTS W

$|L(M_i)| \neq 17$ IF M DOESN'T ACCEPT W .

THEN:



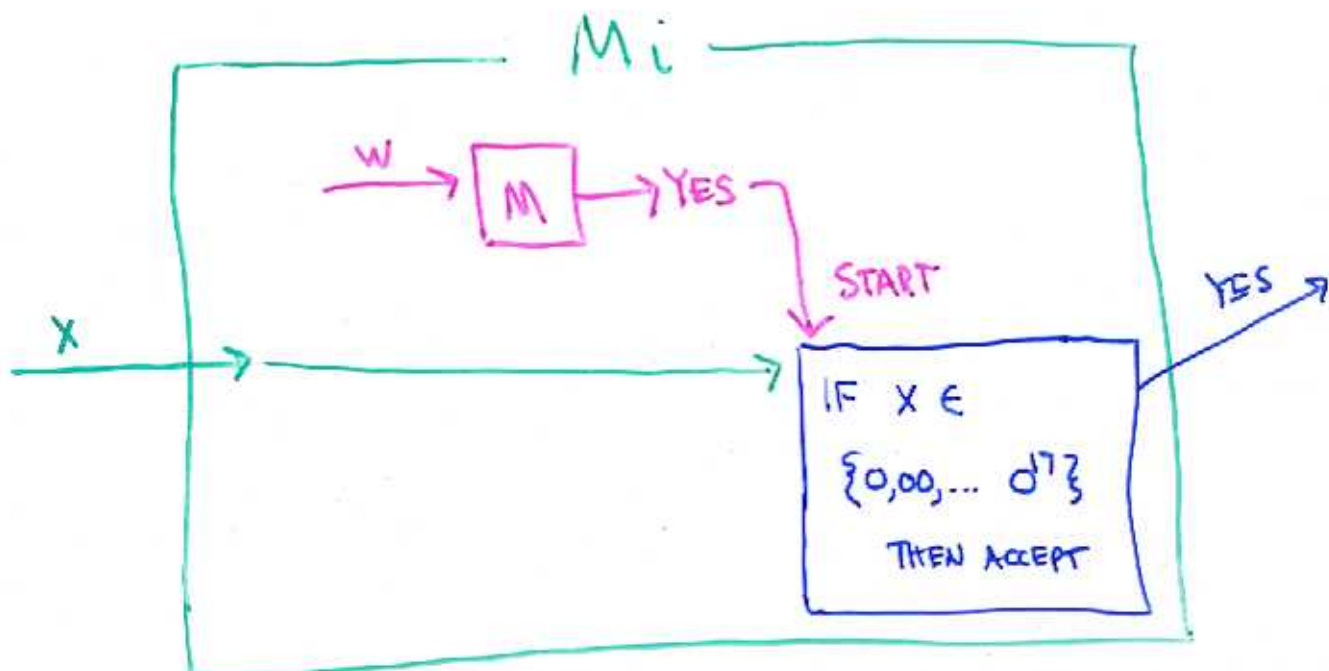


M ACCEPTS $w \iff |L(M_i)| = 17$

Q: WHAT SHOULD M_i LOOK LIKE W.R.T. M, w ?

A: IF $M(w)$ ACCEPTS $L(M_i)$ SHOULD BE ANY LANGUAGE OF 17 WORDS

IF $M(w)$ DOESN'T ACCEPT $L(M_i)$ SHOULD BE ANY LANG OF $\neq 17$ WORDS



IF M ACCEPTS $w \quad L(M_i) = \{0, 00, 000, \dots, 0^{17}\}$ CARDINALITY 17

IF M DOESN'T ACCEPT $w \quad L(M_i) = \emptyset$ CARDINALITY 0