# NON-DETERMINISTIC T.M. (NTM)

**RECALL** IN D.T.M.'S $\delta(q,a)$ IS EITHER UNIQUE OR UNDEFINED

**N.T.M.S** $\delta(q,a)$ IS A FINITE SET OF THE TYPE

$$\{(p_1, b_1, M_1), \ldots, (p_k, b_k, M_k)\}$$

AND ANY OF THESE OPTIONS CAN BE USED WHEN PERFORMING THE TRANSITION.

**AS BEFORE** STRING $w$ IS ACCEPTED IF N.T.M. HAS AT LEAST ONE EXECUTION LEADING TO A FINAL STATE.

---

**EXAMPLE** $\Sigma = \{0,1,2,3,\ldots,9\}$

$$L = \{ w \in \Sigma^* \mid \text{A ZERO APPEARS } i \text{ POSITIONS TO LEFT OF SOME } i \text{ IN } w, \text{ WITH } i>0 \}$$

$$= \{ w \mid \exists j>0,\ w_{j-w_j} = 0 \}$$

**THUS** $03156 \notin L$

$3720654432 \in L$

---

**NTM N** $Q = \{q_0, f, [p,0], [p,1], \ldots [p,9]\}$

$F = \{f\}$ $\Gamma = \{B, 0, 1, 2, \ldots, 9\}$

**IDEA** SCAN $w$ LEFT-TO-RIGHT, GUESS AT SOME $w_j = i$, STORE $i$ IN CPU REGISTER AND MOVE $i$ STEPS LEFT TO FIND 0.

## TRANSITIONS

- $\delta(q_0, 0) = \{(q_0, 0, R)\}$ (SINCE $\omega_j > 0$)
- $\forall i > 0$

  $\delta(q_0, i) = \{(q_0, i, R), ([p, i], i, L)\}$

  GUESSING CURRENT CELL IS $\omega_j$
- $\forall i > 1, \forall X \in \Gamma$

  $\delta([p, i], X) = \{([p, i-1], X, L)\}$
- ACCEPT

  $\delta([p, 1], 0) = \{(\theta, 0, R)\}$

---

## EXECUTION TRACE INPUT $\omega = 103332$

$q_0\,103332 \vdash 1q_0\,03332 \vdash 10\,q_3\,3332$

$\vdash 103\,q_3\,332 \vdash 10\,[p,3]\,3332$

$\vdash 1\,[p,2]\,03332 \vdash [p,1]\,103332$

(REJECT)

$q_0\,103332 \overset{*}{\vdash} 103\,q_0\,332 \vdash 1033\,q_0\,32$

$\vdash 103\,[p,3]\,332 \vdash 10\,[p,2]\,3332$

$\vdash 1\,[p,1]\,03332 \vdash 10\,\theta\,3332$

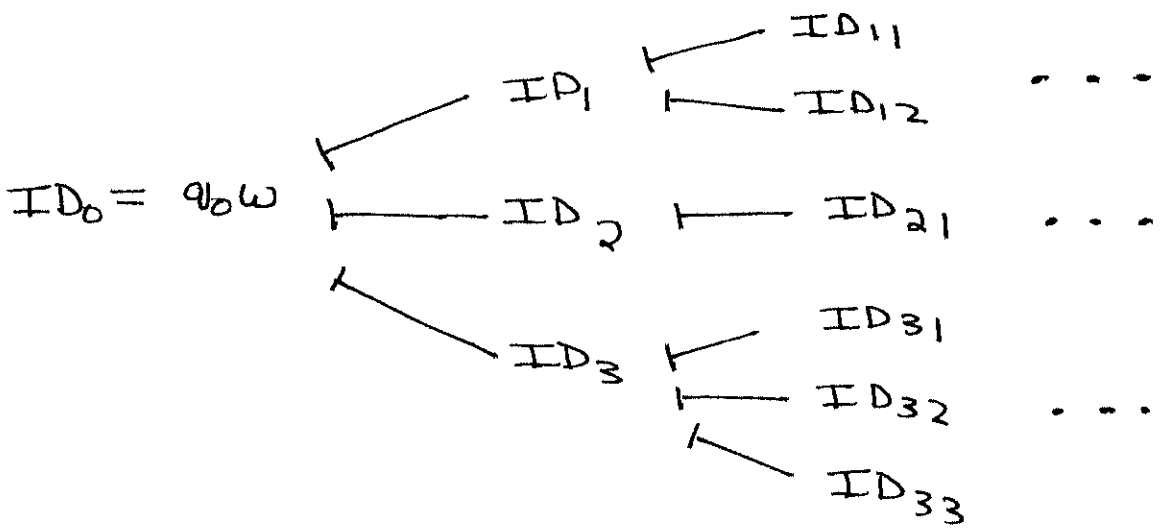
(ACCEPT)

**THEOREM** IF $N$ IS AN NTM
THEN $\exists$ DTM $D$ SUCH THAT $L(D) = L(N)$.

**PROOF** GIVEN $N$ AND INPUT $w$
WILL SHOW HOW A MULTI-TAPE DTM CAN SIMULATE $N$'S EXECUTION ON INPUT $w$.
CLEARLY CAN CONVERT $D$ TO SINGLE-TAPE LATER.

---

**SIMULATION IDEA** CONSIDER EXECUTION TREE OF $N$ ON $w$.




**DTM D** WILL PERFORM A B.F.S. OF THIS EXECUTION, SYSTEMATICALLY ENUMERATING THE ID'S UNTIL IT FINDS AN ACCEPTING I.D.

**TWO TAPES**

**TAPE 1** HOLDS A "QUEUE" OF ID'S OF $N$ IN BFS ORDER

**TAPE 2** SCRATCH TAPE.

## EVOLUTION OF TAPE 1'S CONTENTS.

INITIALLY $\hat{*}\ ID_0\ *$

$\Rightarrow$ $\hat{*}\ ID_0\ *\ ID_0\ *\ ID_0\ *\ ID_0\ *$

$\Rightarrow$ $\hat{*}\ ID_0\ *\ ID_1\ *\ ID_2\ *\ ID_3\ *$

$\Rightarrow$ $*\ ID_0\ \hat{*}\ ID_1\ *\ ID_2\ *\ ID_3\ *$

$\Rightarrow$ $*\ ID_0\ \hat{*}\ ID_1\ *\ ID_2\ *\ ID_3\ *\ ID_1\ *\ ID_1\ *$

$\Rightarrow$ $*\ ID_0\ \hat{*}\ ID_1\ *\ ID_2\ *\ ID_3\ *\ ID_{11}\ *\ ID_{12}\ *$

$\Rightarrow$ $*\ ID_0\ \hat{*}\ ID_1\ \hat{*}\ ID_2\ *\ ID_3\ *\ ID_{11}\ *\ ID_{12}\ *$

$\Rightarrow$ $*\ ID_0\ *\ ID_1\ \hat{*}\ ID_2\ *\ ID_3\ *\ ID_{11}\ *\ ID_{12}\ *\ ID_2\ *$

$\Rightarrow$ $*\ ID_0\ *\ ID_1\ \hat{*}\ ID_2\ *\ ID_3\ *\ ID_{11}\ *\ ID_{12}\ *\ ID_2$

$\Rightarrow$ $*\ ID_0\ *\ ID_1\ *\ ID_2\ \hat{*}\ ID_3\ *\ ID_{11}\ *\ ID_{12}\ *\ ID_2$

$\vdots$

---

## ALGORITHM FOR D.

- TAPE 1 CONTAINS SEQUENCE OF ID'S IN BFS ORDE SEPARATED BY *'S
- $\hat{*}$ MARKS NEXT ID TO BE EXPLORED — ALL TO LEF OF THIS HAVE BEEN EXPLORED ALREADY, AND ALL TO RIGHT HAVE YET TO BE EXPLORED.
- INITIALLY ONLY $ID_0$ IS ON THE TAPE
- USE MULTIPLE TRACKS TO STORE $\hat{*}$ AND FOR OTHE PURPOSES IN FOLLOWING ALGORITHM.

ALGORITHM:

STEP 0 EXAMINE CURRENT $ID_c$ (JUST AFTER $\hat{*}$) AND READ OFF $q, a$ IN IT.

IF $q \in F$ THEN ACCEPT & HALT.

STEP 1 LET $\delta(q,a)$ HAVE $k$ POSSIBLE TRANSITIONS

COPY $ID_c$ ONTO TAPE 1 & MAKE $k$ NEW COPIES OF $ID_c$ AT END OF TAPE 2.

STEP 2 MODIFY $k$ COPIES OF $ID_c$ ON TAPE 2 TO THE $k$ POSSIBLE OUTCOMES OF $\delta(q,a)$ BEING APPLIED

STEP 3 MOVE $\hat{*}$ DOWN PAST $ID_c$

CLEAN UP TAPE 2

RETURN TO STEP 0.

---

VERIFY

- ABOVE CAN BE IMPLEMENTED IN A DTM
- CORRECT IN THAT D ACCEPTS $w$ IF AND ONLY IF SOME EXECUTION PATH OF N ON $w$ WILL ACCEPT $w$.

---

DONE

---

REGALL T.M. M (DTM OR NTM) IS SAID TO HAVE RUNNING TIME IF ON INPUT $w$ WITH $|w| = n$ ALL POSSIBLE EXECUTION PATHS LEAD TO HALTING IN $\leq T(n)$ TRANSITIONS.

## SIMULATION TIME

- SUPPOSE NTM N HAS RUNNING TIME $T(n)$.
- QUESTION WHAT IS D'S RUNNING TIME?

---

LET $m$ BE THE MAXIMUM SIZE OF $\delta(q, a)$ IN N
— MAXIMUM NUMBER OF NON-DETER. CHOICES

---

FOR N ON $w$ LET $t = T(|w|)$

CLEARLY EXECUTION TREE OF N ON $w$ HAS AT MOST $t$ LEVELS.

TREE SIZE $S \leq 1 + m + m^2 + \cdots + m^t$

$$= \frac{m^{t+1} - 1}{m - 1} = O(m^t).$$

THUS D HAS AT MOST $O(m^t)$ ITERATIONS.

---

OBSERVE WHILE DPDAS COULD NOT SIMULATE NPDAS, DTMS CAN SIMULATE NTMS EVEN THOUGH NTMS ARE MORE POWERFUL THAN DTMS.

WHY? BECAUSE THE VERY FACT THAT TMS ARE MORE POWERFUL GIVES THEM THE ABILITY TO SIMULATE NON-DETERMINISM — ALBEIT AT A TERRIBLE INCREASE IN RUNNING TIME.

# ALGORITHMS VS PROCEDURES

**ALGORITHMS** HALTING T.M. — ALWAYS HALTS IN FINITE TIME ON ALL INPUTS AND EXECUTION PATHS, REGARDLESS OF ACCEPTANCE OR REJECTIONS.

**RECURSIVE** CLASS OF LANGUAGES ACCEPTED BY ALGORITHMS (ALSO CALLED DECIDABLE)

**UNDECIDABLE** LANGUAGES WHICH ARE NON-RECURSIVE OR DON'T HAVE ALGORITHMS.

---

**PROCEDURES** ARBITRARY T.M. (MAY NOT HALT ON $w \notin L$)

**RECURSIVELY ENUMBERABLE (R.E.)** LANGUAGE CLASS DEFINED BY T.M. OR PROCEDURES.

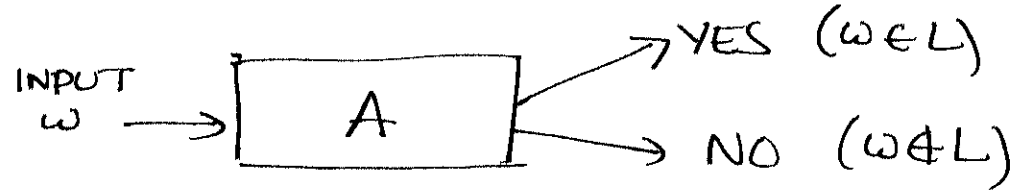
---

**REMARK** ASSUME DTMs IN ABOVE DEFINITIONS, BUT DOESN'T REALLY MATTER AS NTM'S CAN BE SIMULATED

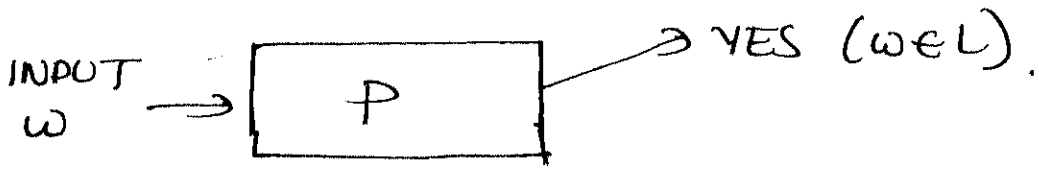
---

**PICTURES**

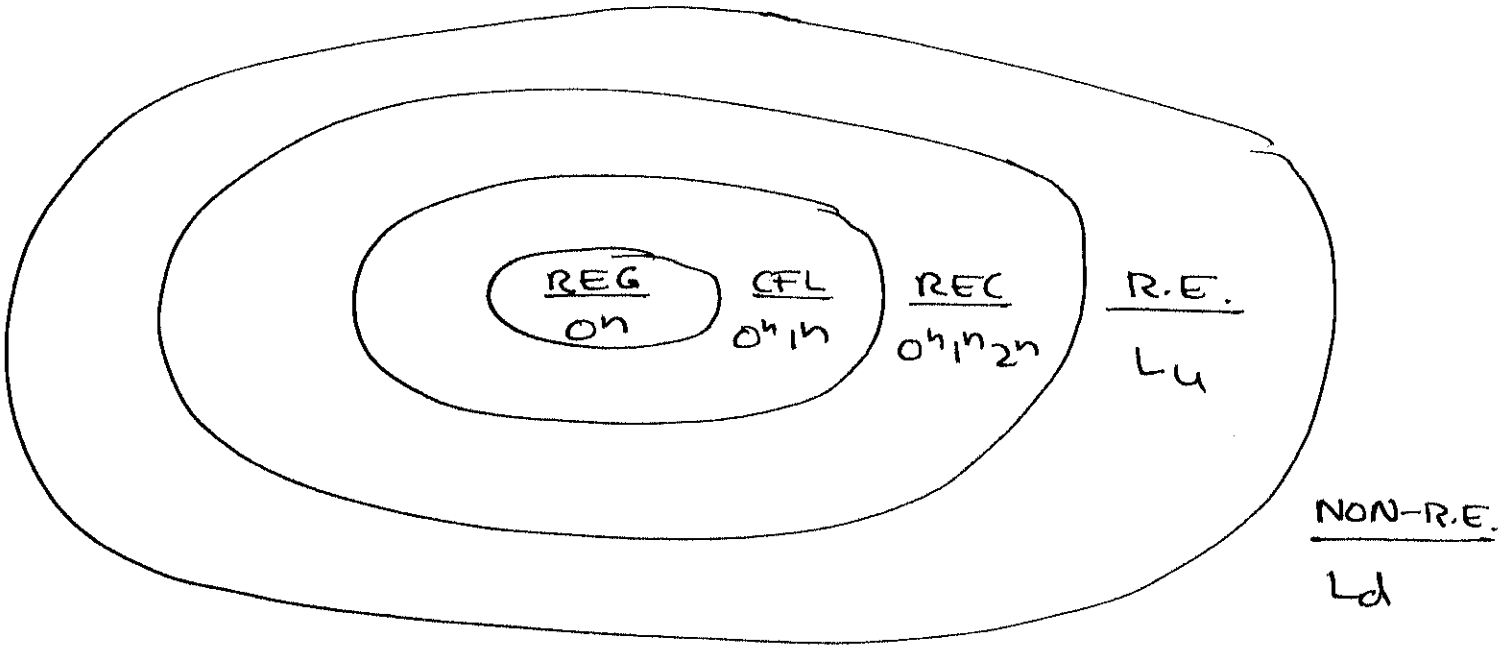
ALGORITHM




PROCEDURE



## WORLD OF LANGUAGES




$L_u / L_d$ WILL BE DEFINED SHORTLY.

---

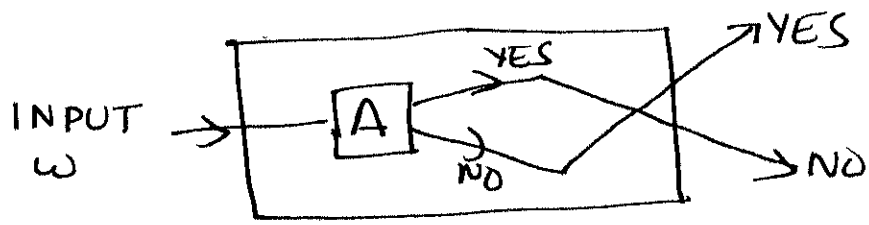
## CLOSURE PROPERTIES

**THEOREM** $L$ RECURSIVE $\Longrightarrow$ $\overline{L}$ RECURSIVE

**PROOF** GIVEN ALGO $A$ FOR $L$

CONSTRUCT ALGO. $\overline{A}$ FOR $\overline{L}$

**ALGO $\overline{A}$**




---

MEANING (OF PICTURE)

YES $\longrightarrow$ NO $\forall q \in F$, $\left\{ \begin{array}{l} \text{REPLACE } \delta(q,X) = (q,-,-) \\ \text{BY } \delta'(q,X) \text{ — UNDEFINE} \end{array} \right.$

NO $\longrightarrow$ YES IF $\delta(q,X)$ UNDEFINED, THEN SET

$$\delta'(q,X) = (q,X,R)$$

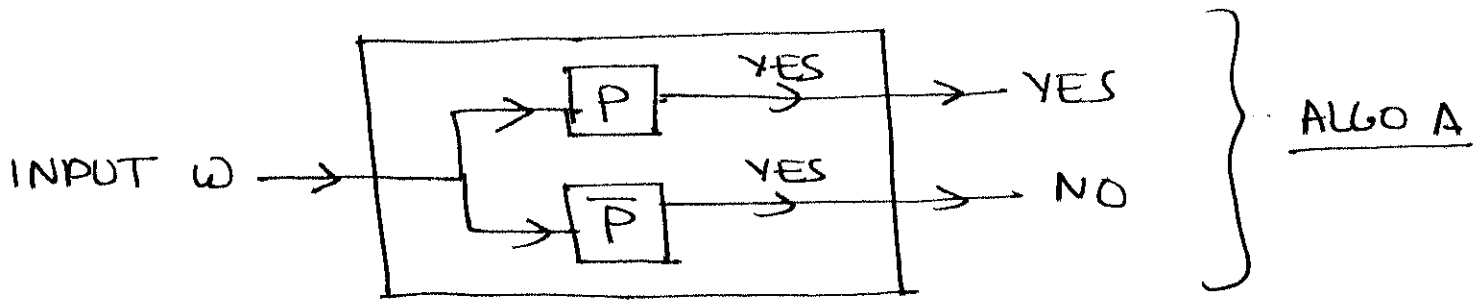
NOTE: A SIMILAR PROCESS CAN BE USED

**THEOREM** BOTH $L, \overline{L}$ ARE R.E. $\Longrightarrow$ BOTH $L, \overline{L}$ ARE REC.

---

**PROOF** LET $P, \overline{P}$ BE PROCEDURES FOR $L, \overline{L}$

RUN IN PARALLEL TO GET ALGO. A




FOR ANY ω ONE OF $P, \overline{P}$ WILL HALT AND GIVE THE RIGHT ANSWER

---

IMPLEMENTATION?

IN A USE 2 TAPES, ONE EACH FOR $P$ AND $\overline{P}$

STATES $p$ IN $P$, $q$ IN $\overline{P}$ $\Longrightarrow$ $\langle p, q \rangle$ IN A.

TRANSITIONS
$$\begin{cases} \delta(p, x) = (p', x', M_1) & \text{IN } P \\ \delta(q, y) = (q', y', M_2) & \text{IN } \overline{P} \end{cases}$$

$$\Longrightarrow \delta(\langle p, q \rangle, x, y) = (\langle p', q' \rangle, (x', M_1), (y', M_2))$$

FINAL? ANY $\langle p, q \rangle$ IN A WHERE $p$ IS FINAL IN F

NOTE IF $\overline{P}$ ACCEPTS, $q$ IS FINAL SO $p$ CANNO BE FINAL, SO A WILL REJECT.

DONE

THUS ANY LANGUAGE L IS SUCH THAT:

EITHER BOTH L, $\bar{L}$ ARE REC.

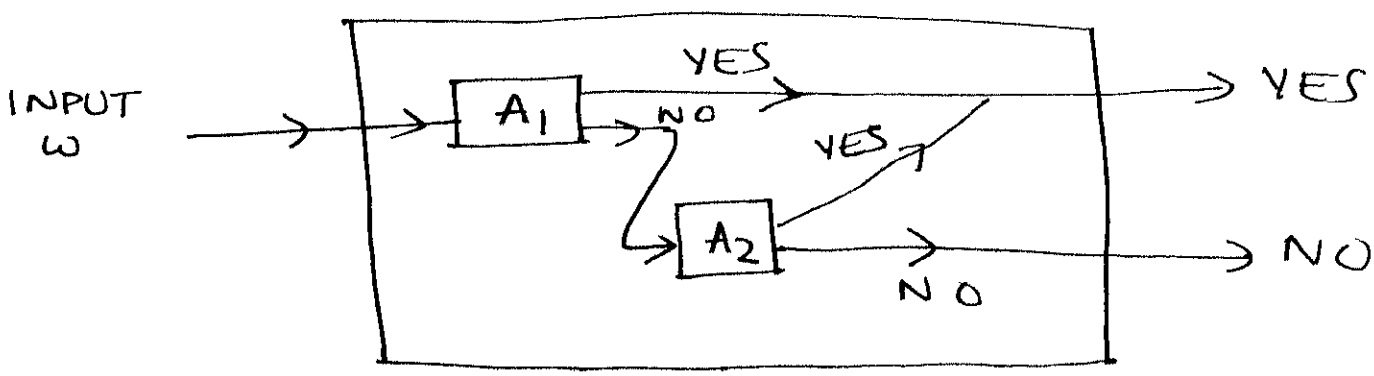
OR AT LEAST ONE OF L, $\bar{L}$ IS NON-R.E.

POSSIBILITIES

| | $\bar{L}$ REC | $\bar{L}$ R.E. BUT NOT REC. | $\bar{L}$ NON-R.E |
|---|---|---|---|
| L REC | ✓ | ✗ | ✗ |
| L R.E. BUT NOT REC | ✗ | ✗ | ✓ |
| L NON-R.E. | ✗ | ✓ | ✓ |

THEOREM $L_1, L_2$ REC. $\Longrightarrow L_1 \cup L_2$ REC.

PROOF ASSUME ALGORITHMS $A_1$ FOR $L_1$ AND $A_2$ FOR $L_2$




COMBINING $A_1$ AND $A_2$ IN SERIES?

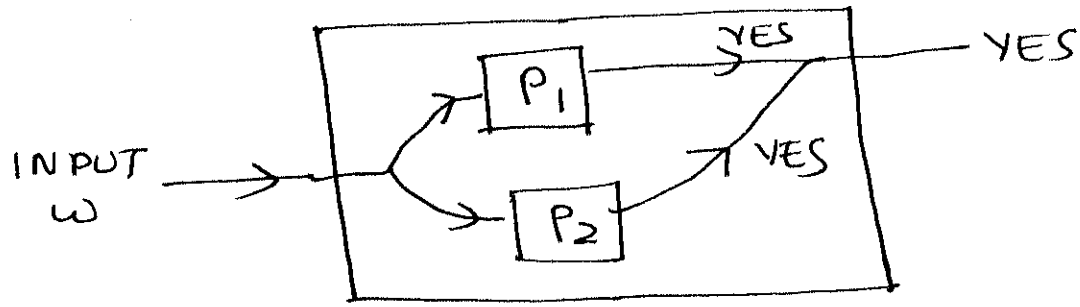
IF $A_1$ HALTS IN NON-FINAL STATE ($\omega \notin L_1$)

THEN HAVE TRANSITION TO INITIAL STATE

OF $A_2$ (HAVING SAVED ω ON SECOND TAPE).

**THEOREM** $L_1, L_2$ ARE R.E. $\Longrightarrow$ $L_1 \cup L_2$ ARE R.E.

**PROOF** ASSUME PROCEDURES $P_1, P_2$ FOR $L_1, L_2$




OBSERVE IF $\omega \in L_1 \cup L_2$, ONE OF $P_1$ AND $P_2$ WILL HALT AND SAY "YES".

---

**EXERCISE** CHECK CLOSURE UNDER $\cap$, REVERSAL.

---

**QUESTION** HOW DO WE SHOW THAT A LANGUAGE IS UNDECIDABLE OR NON-R.E.?

**KEY CONCEPT** UNIVERSAL T.M. (UTM)

UTM
- INPUT { T.M. M; M'S INPUT ω }
- OUTPUT ACCEPTS $\langle M, \omega \rangle \Longleftrightarrow$ M ACCEPTS ω.

**INTUITION**
- UTM $\equiv$ COMPUTER
- TM M $\equiv$ PROGRAM.

**THUS** NEED "PROGRAMMING LANGUAGE" TO BE ABLE TO SPECIFY T.M. M TO UTM.

# ENCODING T.M. AS INTEGERS

**CONSIDER** ANY T.M. $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

**ASSUME**

- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, B\}$
- $Q = \{q_1, q_2, \ldots, q_r\}$
  - — INITIAL STATE $q_1$
  - — FINAL STATE $q_2$

---

**SET**

- $X_1 = 0, \; X_2 = 1, \; X_3 = B$
- $D_1 = \text{LEFT}, \; D_2 = \text{RIGHT}$

---

**CONSIDER** TRANSITION $\delta(q_i, X_j) = (q_k, X_\ell, D_m)$

WITH $i, k \in \{1, \ldots r\}$

$j, \ell \in \{1, 2, 3\}$

$m \in \{1, 2\}$

**ENCODE AS** BIT STRING $0^i 1 0^j 1 0^k 1 0^\ell 1 0^m$.

---

**ENCODING T.M.**

$$\langle M \rangle = C_1 11 C_2 11 C_3 11 \cdots 11 C_n$$

WHERE $C_1, C_2, \ldots C_n$ ARE THE ENCODINGS OF THE $n$ TRANSITIONS IN $M$.

---

**NOTE** WE USE $\langle M \rangle$ TO DENOTE THE STRING ENCODIN T.M. M AND TO AVOID CONFUSION WITH M.

<u>NOTE</u> TO ENCODE T.M. M ALONG WITH ITS INPUT $\omega$

$$\langle M, \omega \rangle = \langle M \rangle 111\, \omega$$

---

<u>OBSERVE</u> ANY BIT-STRING WHICH IS A VALID ENCODING WILL ENCODE A UNIQUE T.M. — BUT IF IT IS NOT A VALID ENCODING, WE WILL ASSUME THAT IT ENCODES THE PARTICULAR MACHINE $M_0$ WITH 1 STATE AND NO TRANSITIONS

$$(L(M_0) = \emptyset).$$

---

<u>THUS</u> EACH T.M. M CAN BE ENCODED AS BINARY STRING $\langle M \rangle$ SUCH THAT:

a) EACH T.M. HAS $\geq 1$ ENCODING

b) EACH NUMBER ENCODES A <u>UNIQUE</u> T.M.

---

<u>ENUMERATING BINARY STRINGS.</u>

WE DEFINE A SPECIFIC ORDERING OF ALL BINARY STRINGS — IN INCREASING ORDER OF LENGTH, AND WITHIN ALL STRINGS OF SAME LENGTH WE ORDER THEM LEXICOGRAPHICALLY

$$\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots$$

<u>LET</u> $\omega_i$ BE THE $i^{TH}$ STRING IN THIS ORDER $(\omega_1 = \epsilon)$.

---

<u>DEFINE</u> $M_i$ AS THE T.M. ENCODED BY $\omega_i$

<u>THUS</u> WE GET AN ORDERING OF T.M.'S IN WHICH EACH T.M. APPEARS AT LEAST ONCE, BUT POSSIBLY MANY TIMES

# THE DIAGONALIZATION LANGUAGE

**CONSIDER** INFINITE TABLE $T$ SUCH THAT $\forall i, j \in N$

$$T(i,j) = \begin{cases} 1 & w_j \in L(M_i) \\ 0 & w_j \notin L(M_i) \end{cases}$$

---

| | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $\cdots$ |
|---|---|---|---|---|---|
| $M_1$ | 0 | 1 | 1 | 0 | |
| $M_2$ | 1 | 1 | 0 | 0 | |
| $M_3$ | 0 | 0 | 0 | 0 | |
| $M_4$ | 1 | 1 | 0 | 1 | |
| $\vdots$ | | | | | |

**THUS** EACH ROW IS A CHARACTERISTIC VECTOR OF $L(M_i)$ ENCODING WHICH STRINGS BELONG TO IT

**NOTE** WE CAN POSTULATE THIS TABLE ONLY BECAUSE WE SHOWED ORDERING/ENUMERATION OF $w_i$/$M_j$

---

**DEFN** DIAGONAL LANGUAGE

$$L_d = \{ w_i \mid T(i,i) = 0 \}$$
$$= \{ w_i \mid w_i \notin L(M_i) \}$$

---

**THUS** $L_d$ IS DEFINED BY TAKING DIAGONAL OF $T$, COMPLEMENTING EACH BIT TO OBTAIN A LANGUAGE'S CHARACTERISTIC VECTOR.

**QUESTION** Is $L_d$ R.E.? Is $L_d$ REC?

**THEOREM** $L_d$ IS NON—R.E.

**PROOF** ASSUME $L_d$ IS R.E. AND HAS A T.M.

THEN $\exists\, k \in N$ SUCH THAT $L(M_k) = L_d$

QUESTION IS $w_k \in L_d$?

---

CASE 1 $[w_k \in L_d]$

$\Rightarrow w_k \in L(M_k)$

$\Rightarrow T(k,k) = 1$

$\Rightarrow w_k \notin L_d$ CONTRADICTION

---

CASE 2 $[w_k \notin L_d]$

$\Rightarrow w_k \notin L(M_k)$

$\Rightarrow T(k,k) = 0$

$\Rightarrow w_k \in L_d$ CONTRADICTION

---

**INTUITION** $L_d$ WAS DEFINED SO THAT IT DISAGREED WITH EACH $M_i$ ON AT LEAST STRING $(w_i)$ — THUS NONE OF THE $M_i$'S COULD HAVE $L_d$ AS ITS LANGUAGE

**BUT** ALL T.M. M ARE $M_i$ FOR SOME $i$, SO NO T.M. CAN ACCEPT $L_d$

**PROBLEM** $L_d$ IS NON—CONSTRUCTIVE!

# UNIVERSAL T.M.

**UTM**
- **INPUT** $\langle M, \omega \rangle$ — ENCODING OF SOME T.M. M AND ITS INPUT $\omega$
- **ACTION** SIMULATE M ON $\omega$, AND ACCEPT $\langle M, \omega \rangle$ IF AND ONLY IF M ACCEPTS $\omega$.

---

**QUESTION** WHAT IS ITS LANGUAGE?

**DEFN** UNIVERSAL LANGUAGE $L_u$

$$L_u = \{ \langle M, \omega \rangle \mid \omega \in L(M) \}.$$

**THEOREM** $L_u$ IS R.E.

**WHY?** BECAUSE WE CAN CONSTRUCT A UTM U FOR $L_u$

---

**UTM U** HAS 4 TAPES

- **TAPE 1** CONTAINS $\langle M, \omega \rangle$ (READ-ONLY)
- **TAPE 2** SIMULATES M'S TAPE
- **TAPE 3** CURRENT STATE $q_i$ OF M
- **TAPE 4** SCRATCH TAPE.

**EXECUTION?** STEP-BY-STEP SIMULATION OF M ON $\omega$

**INITIALLY** COPY $\omega$ TO TAPE 2, $q_i$ TO TAPE 3

**EACH STEP** USE TAPES 2/3 TO DETERMINE WHICH TRANSITION, AND MODIFY TAPES 2/3 APPROPRIATELY — ACCEPTING IF FINAL STATE

# UNDECIDABILITY OF $L_U$.

OBSERVE $L_U$ IS R.E. SINCE $L_U = L(U)$, BUT NOW WE SHOW THAT IT ISN'T RECURSIVE.

TO DO SO WE WILL EMPLOY NOTION OF A "REDUCTION" WHICH WE HAVE SEEN EARLIER INFORMALLY.

---

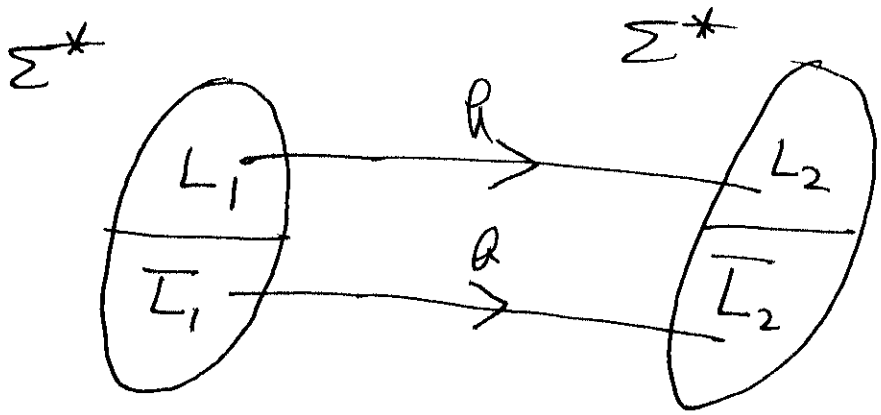
DEFN $L_1$ REDUCES $L_2$ (DENOTED $L_1 < L_2$) IF THERE EXISTS A FUNCTION $f$ (CALLED THE REDUCTION) SUCH THAT:

(a) SOME T.M. $M_f$ COMPUTES $f$ BY TAKING AS INPUT A STRING $w$ AND HALTING WITH STRING $f(w)$ ON ITS TAPE

(b) $f$ IS SUCH THAT

$$w \in L_1 \iff f(w) \in L_2$$

---



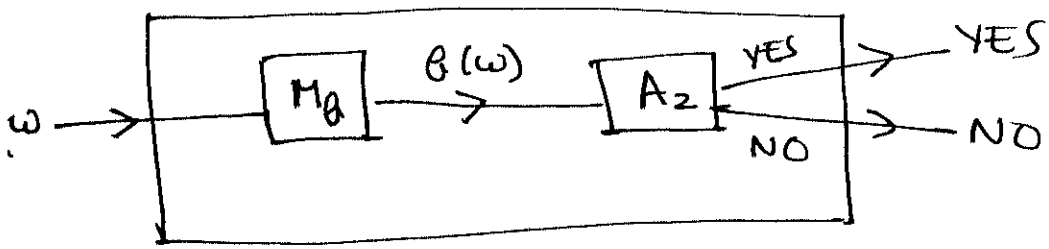

THAT IS $f$ MAPS ALL STRINGS IN $L_1$ TO A SUBSET OF STRINGS IN $L_2$, AND ALL STRINGS IN $\overline{L_1}$ TO A SUBSET OF STRINGS IN $\overline{L_2}$

**THEOREM** $L_1 < L_2$ AND $L_2$ IS RECORSIVE | DECIDABLE
$\Longrightarrow$ $L_1$ IS RECURSIVE | DECIDABLE

**PROOF** GIVEN $\begin{cases} \text{ALGORITHM } A_2 \text{ FOR } L_2 \\ \text{ALGORITHM } M_f \text{ FOR } f \end{cases}$

CONSTRUCT ALGORITHM $A_1$ FOR $L_1$

ALGO $A_1$




$A_1$ IS CORRECT SINCE $\omega \in L_1 \Longleftrightarrow f(\omega) \in L_2$

DONE

---

**COROLLARY:** $L_1 < L_2$ AND $L_1$ IS UNDECIDABLE | NON-REC
$\Longrightarrow$ $L_2$ IS UNDECIDABLE | NON-REC

**OBSERVE**
- SWITCH IN DIRECTION OF STATEMENT
- CAN BE USED TO SHOW LANGUAGES ARE UNDECIDABLE

---

**REMARK** ABOVE RESULTS APPLY TO R.E. ALSO

**THUS** IF $L_1 < L_2$ AND $L_2$ R.E.
THEN $L_1$ R.E.

**AND** IF $L_1 < L_2$ AND $L_1$ NON-R.E.
THEN $L_2$ NON-R.E.

**THEOREM** $L_u$ IS NON-RECURSIVE

**PROOF** SINCE $L_d$ IS NON-R.E.
FOLLOWS THAT $\overline{L_d}$ IS NON-REC.
WILL SHOW $\overline{L_d}$ REDUCES TO $L_u$ $(\overline{L_d} < L_u)$
THUS $L_u$ MUST BE NON-REC

---

**RECALL** $L_d = \{ w_i \mid w_i \notin L(M_i) \}$

$$\Longrightarrow \overline{L_d} = \{ w_i \mid w_i \in L(M_i) \}$$

**NOTE** BY DEFN, $w_i = \langle M_i \rangle$

**THUS** $\overline{L_d}$ IS SET OF STRINGS ACCEPTED BY T.M. M WHICH THEY ENCODE.

---

**REDUCTION $f$** GIVEN STRING $w$ WHICH IS INPUT FOR $\overline{L_d}$
PRODUCE STRING $\langle M, w \rangle$ — INPUT FOR $L_u$.

DEFINE $f(w) = \langle w, w \rangle$
$= w \,|||\, w$

---

**CLEARLY** $\exists$ HALTING T.M. $M_f$ TO CONVERT $w$ INTO $\langle w, w \rangle$

**NEED TO SHOW** $w \in \overline{L_d} \iff f(w) \in L_u$.

**OBSERVE** $w_i \in \overline{L_d} \iff w_i \in L(M_i)$
$\iff \langle M_i, w_i \rangle \in L_u$
$\iff \langle w_i, w_i \rangle \in L_u$
$\iff f(w_i) \in L_u$

**DONE**

THUS

- $L_d$ IS NON-R.E. DIAGONALIZATION PROOF
- $\overline{L_d}$ IS R.E. BUT NON-REC. SIMPLE EXERCISE
- $L_u$ IS R.E. BUT NON-REC BY REDUCTION FROM $\overline{L_d}$
- $\overline{L_u}$ IS NON-R.E. BY INFERENCE FROM ABOVE.

---

OBSERVE $L_d$ WAS SHOWN NON-R.E. VIA A DIRECT PROOF MUCH LIKE THE "HELLO-WORLD" PROOF —— SINCE $w_i \in L_d$ WHEN $M_i$ DOESN'T ACCEPT ITS OWN ENCODING

BUT FROM NOW ON WE CAN ESTABLISH NEW PROBLEMS ARE UNDECIDABLE VIA REDUCTIONS, AS WE DID FOR $L_u$.

IDEA A TO SHOW L IS NON-R.E., GIVE A REDUCTION FROM KNOWN NON-R.E. LANGUAGE SUCH AS $L_d$ OR $\overline{L_u}$

B TO SHOW L IS NON-REC., GIVE A REDUCTION FROM KNOWN NON-REC LANGUAGE SUCH AS $\overline{L_d}$ OR $L_u$.

NOTE WHILE REDUCTIONS FROM $L_d$ OR $\overline{L_u}$ WILL SHOW THAT LANGUAGE L IS NON-REC TOO, SINCE IT ALSO SHOWS L IS NON-R.E. IT CANNOT WORK WHEN L IS NON-REC BUT R.E.

CONSIDER FOLLOWING LANGUAGES

$$L_e = \{ \langle M \rangle \mid L(M) = \emptyset \}$$

$$L_{ne} = \{ \langle M \rangle \mid L(M) \neq \emptyset \}$$

HERE $L_e$ IS SET OF ALL STRINGS $w$ SUCH THAT T.M. $M$ ENCODED BY $w$ HAS AN EMPTY LANGUAGE, AND $L_{ne}$ IS THE COMPLEMENT LANGUAGE.

---

CLAIM 1 $L_{ne}$ IS R.E.

IDEA CONSTRUCT NTM $N$ FOR $L_{ne}$

GIVEN INPUT $\langle M \rangle$ $N$ BEHAVES AS FOLLOWS:

1) GUESS A STRING $w \in \Sigma^*$
2) SIMULATE $M$ ON $w$ (LIKE A UTM)
3) ACCEPT $\langle M \rangle$ IF $M$ ACCEPTS ON $w$

CLEARLY $\langle M \rangle \in L(N) \iff \exists w,\ w \in L(M)$

$$\iff \langle M \rangle \in L_{ne}$$

---

CLAIM 2 $L_{ne}$ IS NON-REC.

PROOF GIVE A REDUCTION FROM $L_u$ TO $L_{ne}$

REDUCTION $f$ — INPUT $\langle M, w \rangle$ — INSTANCE OF $L_u$

— OUTPUT $\langle M' \rangle$ — INSTANCE OF $L_{ne}$

— SUCH THAT

$$\langle M, w \rangle \in L_u \iff \langle M' \rangle \in L_{ne}$$

AND $f$ COMPUTABLE BY HALTING T.M.

DESCRIPTION OF $M'$

SUPPOSE $M'$ HAS INPUT $z$

THEN $M'$ IGNORES $z$ COMPLETELY AND INSTEAD SIMULATES $M$ ON $\omega$ (USING UTM)

IF $M$ ACCEPTS $\omega$ THEN $M'$ HALTS & ACCEPTS $z$

OTHERWISE IF $M$ NEVER HALTS ON $\omega$ OR REJECTS $\omega$ $M'$ ALSO NEVER HALTS OR REJECTS $z$.

---

OBSERVE

IF $\omega \in L(M)$ $L(M') = \Sigma^*$

IF $\omega \notin L(M)$ $L(M') = \phi$

THUS $\langle M, \omega \rangle \in L_u \iff \langle M' \rangle \in L_{ne}$.

---

ALSO NOTE IT IS POSSIBLE TO DEFINE HALTING T.M. $M_\theta$ WHICH GIVEN $M, \omega$ CAN CONSTRUCT $M'$ WHICH BEHAVES AS ABOVE.

---

THUS $L_u < L_{ne}$, AND SINCE $L_u$ IS NON-REC IT FOLLOWS THAT $L_{ne}$ IS NON-REC.

DONE

---

$L_{ne}$ R.E. BUT NOT-REC.

$L_e$ MUST THEN BE NON-R.E.

<u>OBSERVE</u> WE SHOWED THAT A SPECIFIC PROPERTY OF T.M. LANGUAGES (NON-EMPTINESS) IS UNDECIDABLE.

<u>TURNS OUT</u> SIMILAR PROOF SHOWS THE UNDECIDABILITY OF <u>ALL</u> NON-TRIVIAL PROPERTIES OF R.E. LANGUAGES.

---

<u>PROPERTY P</u> CAN BE REPRESENTED BY A LANGUAGE

$$L_P = \{ \langle M \rangle \mid L(M) \text{ HAS PROPERTY } P \}$$

<u>OBSERVE</u> P IS NOT A ~~LANGUAGE~~ PROPERTY OF T.M.'S (E.G., HAS $\leq 100$ STATES), BUT OF THEIR LANGUAGES — THUS ALL T.M.'S WITH THE SAME LANGUAGE ARE EITHER ALL IN $L_P$ OR ALL NOT IN $L_P$

---

<u>NON-TRIVIAL PROPERTY</u> AT LEAST ONE R.E. LANGUAGE HAS PROPERTY P, AND AT LEAST ONE DOESN'T.

---

<u>RICE'S THEOREM</u> EVERY NON-TRIVIAL PROPERTY OF R.E. LANGUAGES IS UNDECIDABLE

<u>PROOF</u> <u>FIX</u> ANY NON-TRIVIAL PROPERTY P

<u>ASSUME</u> $\emptyset$ DOESN'T HAVE PROPERTY P

(<u>OTHERWISE</u> WORK WITH $\overline{P}$ — SINCE P IS DECIDABLE IF AND ONLY IF $\overline{P}$ IS DECIDABLE)

<u>WILL SHOW</u> $L_U < L_P$.

SINCE P IS NON-TRIVIAL, CAN ASSUME THERE IS SOME $M_1$ SUCH THAT $L(M_1)$ HAS PROPERTY P ($M_1 \in L_P$) AND WE ALREADY ASSUMED $\emptyset$ DOESN'T HAVE P.

IDEA REDUCTION $f$ TAKES AS INPUT $\langle M, w \rangle$ AN INSTANCE OF $L_u$ AND PRODUCES $M_2$

---

$M_2$'S BEHAVIOR

- IGNORES ITS OWN INPUT $z$ AT FIRST
- SIMULATES $M$ ON $w$
- IF $M$ ACCEPTS $w$ $M_2$ SIMULATES $M_1$ ON $z$, AND ACCEPTS IF $M_1$ ACCEPTS $z$
- IF $M$ REJECTS $w$ (OR DOESN'T HALT) $M_2$ DOES THE SAME

---

THUS

- $w \in L(M) \implies L(M_2) = L(M_1)$ [HAS P]
- $w \notin L(M) \implies L(M_2) = \emptyset$ [HASN'T P]

OR $\langle M, w \rangle \in L_u \iff M_2 \in L_P$

---

FOLLOWS $L_P$ IS UNDECIDABLE

DONE

---

NOTE $L_{no}$ PROOF WAS SAME, WITH $L(M_1) = \Sigma^*$.