CLOSURE PROPERTIES OF CFLs.

THEOREM CFLs ARE CLOSED UNDER UNION ($\cup$), CONCATENATION ($\cdot$) AND CLOSURE ($*$).

PROOF

- **IDEA** GRAMMAR—BASED CONSTRUCTION.
- **GOAL** GIVEN CFLs L_1 AND L_2

  SHOW THAT THE FOLLOWING ARE CFLs

  — $L_1 \cup L_2$
  — $L_1 \cdot L_2$
  — L_1^*

LET CFLs L_1 & L_2 HAVE CFGs G_1 & G_2

ASSUME ALL VARIABLES IN G_1 AND G_2 ARE DISTINCT WITH START SYMBOLS S_1 AND S_2.

NEW GRAMMARS

(a) $L_1 \cup L_2$ COMBINE G_1 AND G_2 VIA THE PRODUCTION

$$S \to S_1 \mid S_2$$

(b) $L_1 \cdot L_2$ USE $S \to S_1 . S_2$

(c) L_1^* ADD $S \to S_1 . S \mid \epsilon$ TO G_1

CORRECTNESS SIMPLE EXERCISE

TEXTBOOK PROVES CLOSURE UNDER "SUBSTITUTION" — MORE GENERAL AND INCLUDES ABOVE AS SPECIAL CASES, BUT AMOUNTS TO SAME PROOF

EXAMPLE

$$L_1 = \{0^i 1^j 2^k \mid i = j\}$$
$$L_2 = \{0^i 1^j 2^k \mid j = k\}$$

G1

$S_1 \to AB$

$A \to 0A1 \mid \epsilon$

$B \to 2B \mid \epsilon$

G2

$S_2 \to CD$

$C \to 0C \mid \epsilon$

$D \to 1D2 \mid \epsilon$

CONSIDER $L = L_1 \cup L_2 = \{0^i 1^j 2^k \mid i = j \text{ OR } j = k\}$

GRAMMAR G

- **USE** ALL OF G_1, G_2
- **ADD** $S \to S_1 \mid S_2$.

THEOREM CFLs **NOT** CLOSED UNDER INTERSECTION

PROOF **CONSIDER** L_1, L_2 IN ABOVE EXAMPLE

DEFINE $L = L_1 \cap L_2$

$$\Rightarrow L = \{0^i 1^j 2^k \mid i = j \text{ AND } j = k\}$$
$$= \{0^n 1^n 2^n \mid n \geq 0\}$$

BUT PUMPING LEMMA SHOWS L IS **NOT** CFL.

THEOREM CFLs **NOT** CLOSED UNDER COMPLEMENT.

PROOF OTHERWISE WOULD GET CLOSURE UNDER $\cap$

SINCE $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

AND CLOSED UNDER UNION.

THEOREM CFLs CLOSED UNDER REVERSAL

IDEA REVERSE R.H.S. OF EACH PRODUCTION

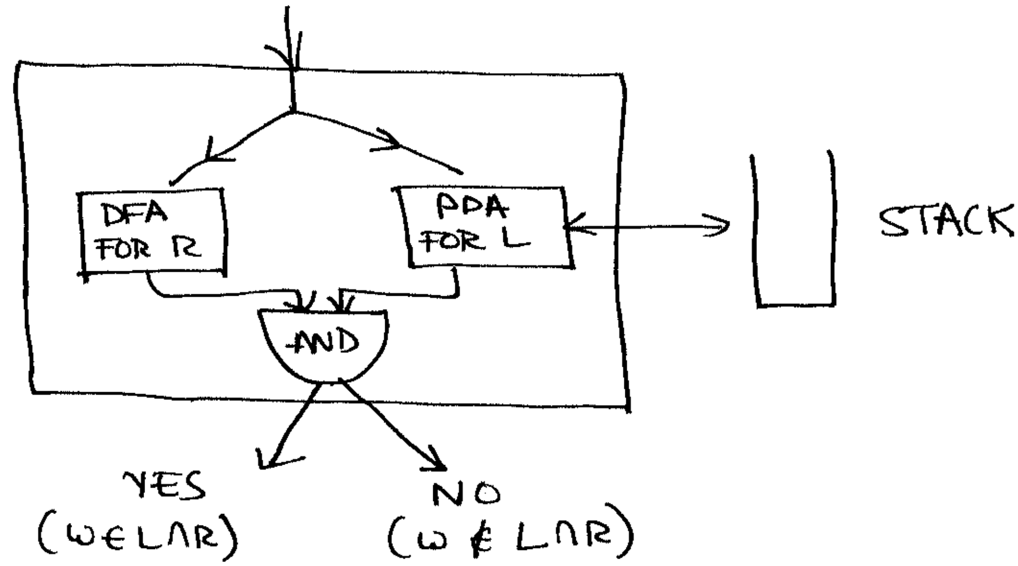
E.G. $A \to BCD$ BECOMES $A \to DCB$

THEOREM CFLs CLOSED UNDER INTERSECTION WITH REGULAR LANGUAG[ES]

OR

L IS CFL AND R IS REGULAR $\Longrightarrow$ $L \cap R$ IS CFL.

IDEA



GIVEN PDA P FOR L, DFA D FOR R.

MAKE PDA M FOR $L \cap R$.

STATES OF M $[q, p]$ — D's STATE (q), P's STATE (p)

TRANSITIONS **SUPPOSE**
$$\begin{cases} \text{IN D} & \delta(q, a) \longrightarrow q' \\ \text{IN P} & \delta(p, a, X) \longrightarrow (p', \alpha) \end{cases}$$

THEN $\Big\{ \text{IN M} \quad \delta([q,p], a, X) \longrightarrow ([q', p'], \alpha)$

IF $a = \epsilon$ THEN $q' = q$.

ACCEPT? **ASSUME** L IS FINAL STATE LANGUAGE OF P

FINAL STATES OF M $[q, p]$ WITH q FINAL IN D AND p FINAL IN P

CFL DECISION PROBLEMS.

EMPTINESS GIVEN CFL L (AS PDA/CFG), IS $L = \emptyset$?

IDEA $L = \emptyset \iff$ IN CFG, START S IS USELESS

THUS USE ALGORITHM FOR "USELESS" ELIMINATION.

MEMBERSHIP GIVEN w AND CFL L, IS $w \in L$?

IDEA ASSUME GIVEN CFG G FOR L

STEP 1 IF $w = \epsilon$, CHECK IF S IS NULLABLE

STEP 2 CONSTRUCT CNF GRAMMAR G' FOR $L - \{\epsilon\}$

STEP 3 TRY ALL POSSIBLE DERIVATIONS IN G' OF LENGTH $\leq 2n-1$, WHERE $n = |w|$

AT MOST $|P|^{2n-1}$ POSSIBILITIES

WHY $2n-1$? FOR $|w| = n$ AND CNF GRAMMAR G'

$$S \overset{*}{\Longrightarrow} w$$

IN EXACTLY $2n-1$ STEPS — EXERCISE

FINITENESS SIMILAR TO REGULAR — HOMEWORK PROBLEM.

EQUALITY? GIVEN CFL'S L_1 & L_2, IS $L_1 = L_2$?

AMBIGUITY? GIVEN CFG G, IS G AMBIGUOUS?

CLAIM SUCH PROBLEMS CANNOT BE SOLVED — NEED TURING MACHINES TO PROVE THAT!

RECALL EQUALITY-TESTING FOR REG. LANG. USED CLOSURE UNDER COMPLEMENT/INTERSECTION

$$\left(L_1 = L_2 \iff L = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset \right)$$

DECIDABILITY THEORY

QUESTION HOW DO WE SHOW THAT THERE IS NO ALGORITHM TO DECIDE EQUALITY / AMBIGUITY OF CFLs; OR ANY OTHER PROBLEM WHICH IS NOT COMPUTABLE

PROBLEM WE DON'T HAVE A FORMAL, MATHEMATICAL DEFINITION OF ALGORITHMS.

SIMPLER GOAL? LET US TRY TO PROVE THAT THERE IS NO C PROGRAM TO SOLVE SUCH PROBLEMS.

HELLO-WORLD PROBLEM.

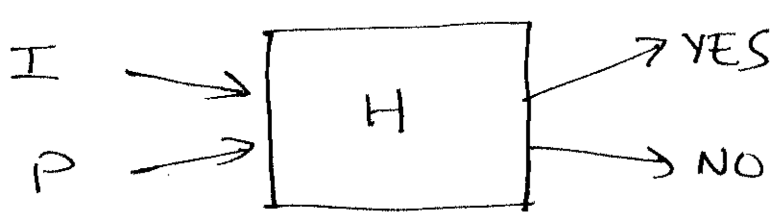
PROGRAM P_0

```
MAIN ( )
{   PRINTF ("HELLO, WORLD \n");
}
```

CLEARLY THE FIRST 12 CHARACTERS OUTPUT BY P_0 IS "HELLO, WORL

HELLO-WORLD PROBLEM. GIVEN AN ARBITRARY C PROGRAM P AND AN INPUT I FOR P, DOES P(I) PRINT "HELLO, WORL AS ITS FIRST 12 CHARACTERS?

CONSIDER A SOLUTION H (ITSELF A C PROGRAM)



QUESTION DOES THERE EXIST SUCH C PROGRAM H?

OF COURSE YOU MIGHT TRY TO WRITE H BY HAVING IT SCAN P'S PRINTF STATEMENTS, BUT WE NEED TO KNOW IF THOSE WILL BE EXECUTED ON INPUT I

FERMATS LAST THEOREM. THE EQUATION $x^n + y^n = z^n$ HAS NO INTEGER SOLUTION FOR $n \geq 3$.

OBSERVE FOR $n=2$, ONE SOLUTION IS $(x=3, y=4, z=5)$

$$3^2 + 4^2 = 5^2.$$

BUT FOR $n \geq 3$ FOR 300 YEARS NO PROOF WAS KNOWN UNTIL RECENTLY, WHEN WYLES GAVE A PROOF

BUT HIS FIRST PROOF WAS WRONG, AND SECOND PROOF IS LONG & COMPLEX AND NOT YET FULLY VERIFIED.

CONSIDER C PROGRAM P_1 (GIVEN FULLY IN BOOK).

P_1

1) TAKE INPUT n
2) **FOR** ALL POSSIBLE (x, y, z) **DO**

   IF $x^n + y^n = z^n$

   THEN PRINTF ("HELLO, WORLD \n")

SUPPOSE INPUT = 3, THEN P_1 PRINTS "HELLO, WORLD" ONLY IF **F.L.T.** IS FALSE, ELSE IT LOOPS FOREVER.

QUESTION HOW CAN YOU (OR, C PROGRAM H) DECIDE IF P_1 PRINTS "HELLO, WORLD" ?

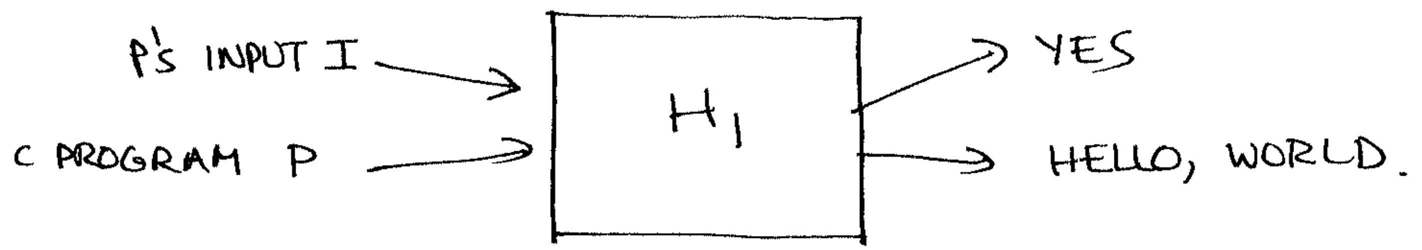
THEOREM THERE CANNOT EXIST A C-PROGRAM H WHICH IS A HELLO-WORLD TESTER.

PROOF **ASSUME** H EXISTS

SHOW CONTRADICTION.

RECALL ON INPUT (P, I), PROGRAM H OUTPUTS "YES" IF P(I) PRINTS "HELLO, WORLD", ELSE H PRINTS "N

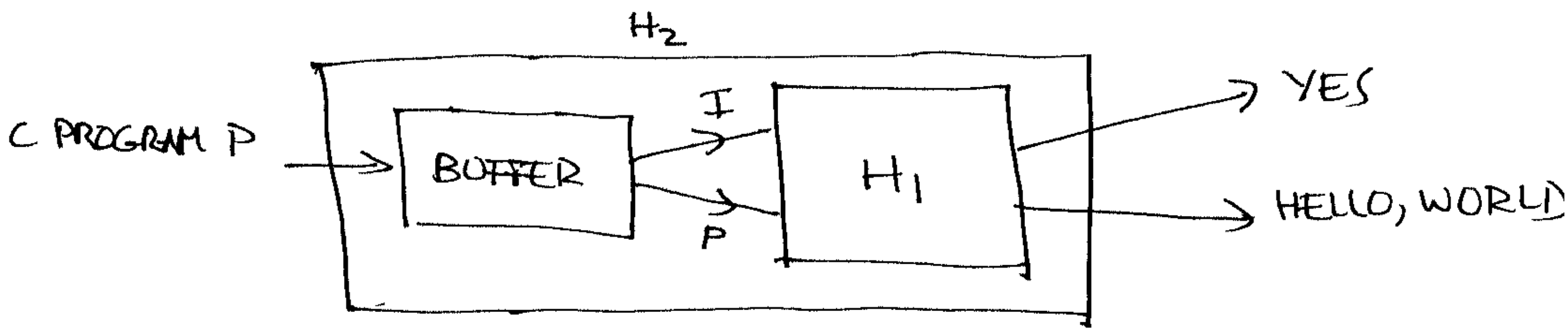
MODIFY H TO C PROGRAM H_1 WHICH BEHAVES EXACTLY LIKE H BUT PRINTS "HELLO, WORLD" INSTEAD OF "N



NOTE OBTAINING H_1 REQUIRES MODIFYING H'S PRINTF("NO")

MODIFY H_1 TO C PROGRAM H_2, WHICH IS SAME AS H_1 BUT TAKES ONLY INPUT P AND MAKES I SAME AS P.

IDEA WRITE A "BUFFER" FUNCTION WHICH READS IN "P", BUFFERS IT, AND THEN FEEDS P/I TO H_1



QUESTION WHAT IS THE OUTPUT OF $H_2(P)$ WHEN $P = H_2$?

SUPPOSE $H_2(H_2) = \text{YES} \implies P(P) = \text{HELLO, WORLD}$

SUPPOSE $H_2(H_2) = \text{HELLO, WORLD} \implies P(P) \neq \text{HELLO, WORLD}$

BUT $P = H_2$

GET CONTRADICTION $\implies H, H_1, H_2$ CANNOT EXIST!

REDUCTIONS

SO FAR WE SHOWED HELLO-WORLD PROBLEM IS UNDECIDABLE

NEXT WE SHOW NEW PROBLEMS CAN BE SHOWN UNDECIDABLE VIA REDUCTION TO HELLO-WORLD, WITHOUT HAVING TO REPEAT ENTIRE PROOF STRUCTURE USED FOR HELLO-WOR

FOO PROBLEM GIVEN PROGRAM R AND ITS INPUT z, DOES R EVER CALL A FUNCTION NAMED FOO WHILE EXECUTING ON INPUT z.

IDEA WE SHOW THAT IF IT'S POSSIBLE TO SOLVE THE FOO PROBLEM ON (R, z), THEN WE CAN SOLVE THE HELLO-WORLD PROBLEM ON (Q, y) — FOR ANY PROGRAM Q WITH INPUT y.

SINCE LATTER IS UNDECIDABLE, SO IS THE FORMER.

SUPPOSE SOME PROGRAM F CAN TAKE AS INPUT (R, z) AND DECIDE THE FOO PROBLEM.

WE SHOW F CAN BE USED TO CONSTRUCT H FOR HELLO-WORLD DECISION ON (Q, y).

IDEA APPLY SERIES OF MODIFICATIONS ON Q:

1) GET Q_1 IF Q CONTAINS A FUNCTION NAMED "FOO" THEN CHANGE ITS NAME THROUGHOUT Q

2) GET Q_2 ADD A DUMMY FUNCTION "FOO" TO Q_1

3) GET Q_3 MODIFY Q_2 TO STORE EVERY CHARACTER IT PRINTS IN SOME ARRAY A.

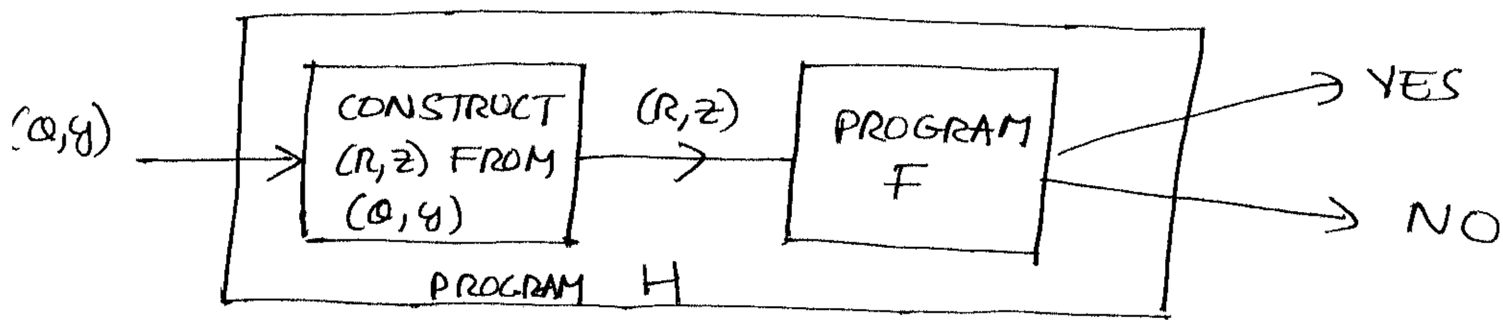
4) <u>GET Q_4</u> MODIFY Q_3 SO THAT AFTER EVERY "PRINT" STATEMENT IT CHECKS ARRAY A TO SEE IF "HELLO, WORLD" HAS BEEN PRINTED YET — IF SO, CALL FUNCTION FOO.

<u>NOW</u> LET $R = Q_4$ AND $z = y$

<u>CLEARLY</u> (Q, y) PRINTS "HELLO, WORLD" $\Longleftrightarrow$ (R, z) CALLS FUNCTION FOO.

<u>THUS</u> WE CAN USE PROGRAM F TO SOLVE FOO-PROBLEM ON (R, z), THEREBY GETTING SOLUTION TO HELLO-WORLD PROBLEM ON (Q, y)

<u>IN EFFECT</u> GIVEN F, WE CAN CONSTRUCT H AS FOLLOWS:



<u>BUT</u> SINCE H CANNOT EXIST, F CANNOT EXIST!

<u>DONE</u>

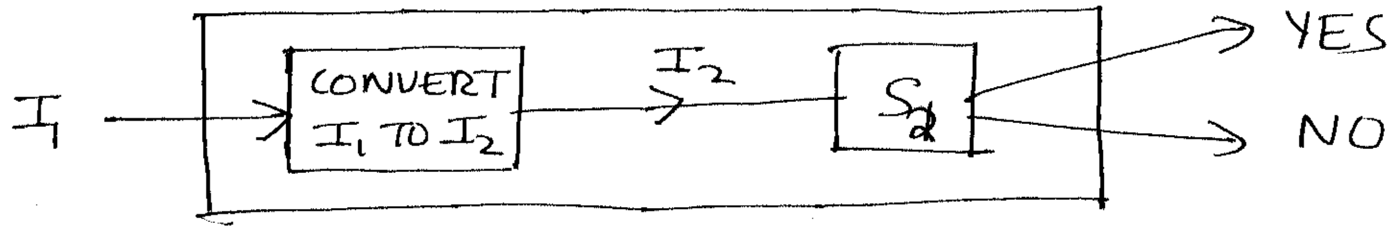
<u>REDUCTION</u>

<u>PROBLEM P_1</u> TAKES INPUT I_1 — <u>KNOWN</u> TO BE UNDECIDABLE

<u>PROBLEM P_2</u> TAKES INPUT I_2 — <u>WE WOULD LIKE TO SHOW</u> IS UNDECIDABLE

<u>REDUCTION IDEA</u> CONVERT I_1 TO I_2 SUCH THAT ANSWER TO P_1 ON INPUT I_1 IS "YES" IF AND ONLY IF ANSWER TO P_2 ON INPUT I_2 IS "YES".

<u>NOW</u> GIVEN SOLUTION PROGRAM S_2 FOR P_2, CAN CONSTRUC SOLUTION PROGRAM S_1 FOR P_1



<u>OF COURSE</u> SINCE S_1 CANNOT EXIST, WE OBTAIN THAT S_2 CANNOT EXIST AND P_2 IS UNDECIDABLE

<u>TRICKY PART</u> CONVERTING I_1 TO I_2 — BOTH ARE PROGRAM

TURING MACHINES.

<u>PROBLEM</u> DEVELOPING COMPUTATION THEORY IN TERMS OF PROGRAMS WRITTEN IN C IS NOT ELEGANT OR EVEN POSSIBLE IN GENERAL:

<u>A</u> HAVE TO WORRY ABOUT RUN-TIME ENVIRONMENT AND RUN-TIME ERRORS

<u>B</u> LANGUAGE CONSTRUCTS ARE TOO COMPLEX

<u>C</u> NEED TO WORRY ABOUT FINITE MEMORY

<u>D</u> "STATE" IS TOO COMPLICATED.

<u>E</u> YOU MIGHT COMPLAIN THAT RESULTS APPLY ONLY TO C — WHAT IF THERE IS A MORE GENERAL

TURING MACHINES

A. SIMPLE AND UNIVERSAL PROGRAMMING LANGUAGE

B. EASY TO DESCRIBE STATE AND RUNTIME CONFIGURATION

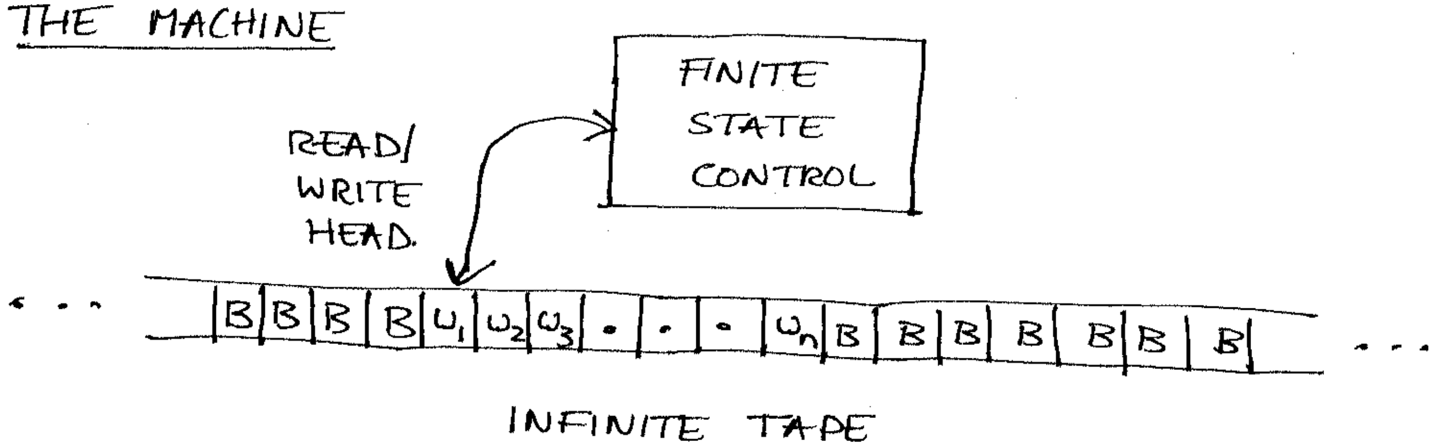
C. CAN SIMULATE ANY KNOWN COMPUTER OR LANGUAGE

D. ALL ATTEMPTS TO DESCRIBE POWERFUL MODELS OF COMPUTATION END UP AS SPECIAL CASES OF TURING M/C.

CHURCH-TURING HYPOTHESIS THERE IS NO MORE POWERFUL MODEL OF COMPUTATION THAN TURING MACHINES

THUS IT MODELS ANYTHING WE CAN COMPUTE.

THE MACHINE



PROGRAMMING LANGUAGE? — SPECIFY TRANSITIONS

MOVE DEPENDS ON
- CURRENT STATE
- SYMBOL READ BY TAPE HEAD

EFFECT
- NEW STATE
- OVERWRITE TAPE CELL
- MOVE HEAD — LEFT/RIGHT.

OBSERVE RELATIONSHIP TO REAL COMPUTERS

— CPU $\equiv$ FINITE STATE CONTROL

— MEMORY $\equiv$ TAPE

CAVEATS ABSTRACTION LOSES SOME ASPECTS

1) MEMORY IS NO LONGER RANDOM ACCESS
2) INSTRUCTION SET IS RATHER LIMITED
3) VARIOUS OTHER FEATURES ARE OMITTED.

HOWEVER T.M. CAPTURE ALL ESSENTIAL ASPECTS

IN FACT CHAPTER 8-6 SHOWS HOW T.M. CAN SIMULATE A REAL COMPUTER — AT SOME LOSS OF EFFICIENCY

PAYOFF SIMPLER ABSTRACTION ALLOWS US TO REASON BETTE

FORMALLY $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Q — FINITE STATES

Σ — INPUT ALPHABET

$F \subseteq Q$ — FINAL STATES

$q_0 \in Q$ — INITIAL STATE

Γ — TAPE ALPHABET

$B \in \Gamma$ — BLANK SYMBOL.

OBSERVE

- Q, Σ, Γ — FINITE ~~STATE~~ SETS
- $\Sigma \subseteq \Gamma$ — SINCE INPUT ω IS PROVIDED ON TAPE
- $B \in \Gamma - \Sigma$ — REST OF TAPE IS INITIALLY BLAN

INITIALLY — STATE q_0

— TAPE CONTAINS ω SURROUNDED BY BLANK