NON-REGULAR LANGUAGES.

CONSIDER
- $L_{EQ} = \{ w \mid w \text{ HAS EQUAL \# OF 0'S AND 1'S} \}$
- $L_{ALT} = \{ w \mid w \text{ HAS ALTERNATING 0'S AND 1'S} \}$

CLAIM L_{ALT} IS REGULAR

PROOF REG. EXP $R = (\epsilon + 0)(10)^*(\epsilon + 1)$

CLAIM L_{EQ} IS NON-REGULAR

QUESTION HOW DO WE PROVE THIS?

INTUITION
- DFAs CAN COUNT UPTO $n = |Q|$ ONLY
- L_{EQ} REQUIRES UNBOUNDED COUNTING.

PUMPING LEMMA LET $L \subseteq \Sigma^*$ BE REGULAR

THEN $\exists$ CONSTANT n (WHICH DEPENDS ON L)

SUCH THAT

FOR ALL $w \in L$ WITH $|w| \geq n$

CAN DECOMPOSE $w = xyz$

SUCH THAT

a) $|y| \geq 1$ $(y \neq \epsilon)$

b) $|xy| \leq n$

c) $\forall k \geq 0$, $xy^kz \in L$.

OBSERVE THIS IS A PROPERTY OF REGULAR LANGUAGES WHICH WE CAN USE TO SHOW THAT A GIVEN LANGUAGE IS NOT REGULAR

IDEA PICK $w \in L$ SUCH THAT WE CAN EASILY ARGUE THAT $xy^kz \notin L$ FOR SOME CHOICE OF k.

PROBLEM MUST DO SO REGARDLESS OF n AND x, y, z CHOICES

APPLICATION PROVING L IS NON-REGULAR.

IDEA PLAY FOLLOWING GAME

a) YOU ASSUME L IS REGULAR

b) ADVERSARY THINKS OF VALUE $n > 0$
(BUT DOESN'T TELL YOU THE VALUE OF n)

c) YOU PICK STRING $w \in L$ WITH $|w| \geq n$
(IN TERMS OF UNKNOWN PARAMETER n)

d) ADVERSARY THINKS OF DECOMPOSITION $w = xyz$
WITH $y \neq \epsilon$, AND $|xy| \leq n$
(BUT DOESN'T TELL YOU x, y, OR z)

e) YOU FIND $k \geq 0$ SUCH THAT $xy^kz \notin L$
(WITHOUT KNOWING x, y, OR z).

NOTE ORDER OF STEPS IS VERY IMPORTANT.

NOTE LACK OF INFORMATION ABOUT n, x, y, OR z MAKES THIS QUITE DIFFICULT — BUT, SURPRISINGLY, WE CAN STILL PLAY THE GAME SUCCESSFULLY.

EXAMPLE L_{EQ} IS NON-REGULAR.

PROOF ASSUME L IS REGULAR AND APPLY P.L.

BY P.L. THERE EXISTS SOME $n \geq 0$

WE CHOOSE $w = 0^n 1^n$

CLEARLY $w \in L$ AND $|w| \geq n$

NOTE CLEVER CHOICE OF w

BY P.L. $w = xyz$ SUCH THAT $\begin{cases} \cdot\ y \neq \epsilon \\ \cdot\ |xy| \leq n. \end{cases}$

BUT SINCE $|xy| \leq n$, IT MUST BE THE CASE THAT x, y HAVE ONLY 0'S.

LET $|x| = a$, $|y| = b$, AND $|xy| = a+b \leq n$, $b > 0$

THEN $w = \underbrace{0^a}_{x} \underbrace{0^b}_{y} \underbrace{0^{n-a-b} 1^n}_{z}$

WE CHOOSE $k = 0$

$\Rightarrow \quad xy^k z \in L_{EQ}$

$\Rightarrow \quad xz \in L_{EQ}$

$\Rightarrow \quad 0^{n-b} 1^n \in L_{EQ}$

BUT FOR $b > 0$, THIS IS NOT TRUE.

CONTRADICTION

THUS ASSUMING L_{EQ} IS REGULAR GIVES A CONTRADICTION

$\Rightarrow L_{EQ}$ IS NON-REGULAR.

NOTE CHOOSING ANY $k \neq 1$ WILL WORK!

EXAMPLE 2 $L_p = \{ w \mid$ LENGTH OF w IS A PRIME NUMBER $\}$

CLAIM L_p IS NOT REGULAR.

PROOF ASSUME L_p IS INDEED REGULAR AND APPLY P.L

BY P.L. THERE EXISTS AN $n > 0$.

WE CHOOSE PRIME $p \geq n+2$

(SINCE THERE AN INFINITE NUMBER OF PRIMES, WE CAN ALWAYS CLAIM $p \geq n+2$ EXISTS, WITHOUT KNOWING n).

CHOOSE $w = 1^p$

CLEARLY $w \in L$ AND $|w| \geq n$

BY P.L. $w = xyz$ SUCH THAT $\begin{cases} \bullet\ y \neq \epsilon \\ \bullet\ |xy| \leq n \end{cases}$

SUPPOSE $|x| = a$, $|y| = b$ WITH $b > 0$, $|a+b| \leq n$

WE CHOOSE $k = p - b$

GET $x y^{p-b} z \in L_p$

BUT
$$\begin{aligned} |x y^{p-b} z| &= a + (p-b)b + (p-a-b) \\ &= (p-b)\,b + (p-b) \\ &= (p-b)\,(b+1) \end{aligned}$$

NOTE $b+1 \geq 2$ SINCE $b \geq 1$

NOTE $p - b \geq 2$ SINCE $\begin{cases} \bullet\ p \geq n+2 \\ \bullet\ b \leq n \end{cases}$

THUS $|x y^{p-b} z|$ HAS 2 NON-TRIVIAL FACTORS AND CANNOT BE A PRIME.

CONTRADICTION

PROOF OF PUMPING LEMMA.

GIVEN REGULAR LANGUAGE L

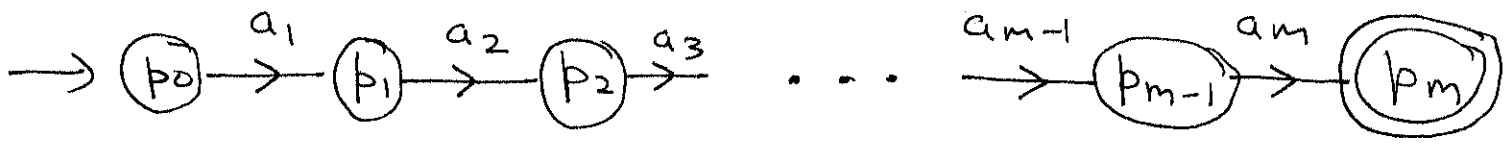
LET M BE SMALLEST DFA FOR L, WITH $|Q| = n$.

CONSIDER ANY $w \in L$ WITH $|w| = m \geq n$

SINCE $w \in L(M)$ CONSIDER ACCEPTING PATH $\hat{\delta}(q_0, w$

IN M FOR $w = a_1 a_2 \ldots a_m$.

SETTING $p_i = \hat{\delta}(q_0, a_1 a_2 \ldots a_i)$ AND $p_0 = q_0$



SINCE $m \geq n$ AND EACH p_i $(0 \leq i \leq m)$ BELONGS TO Q

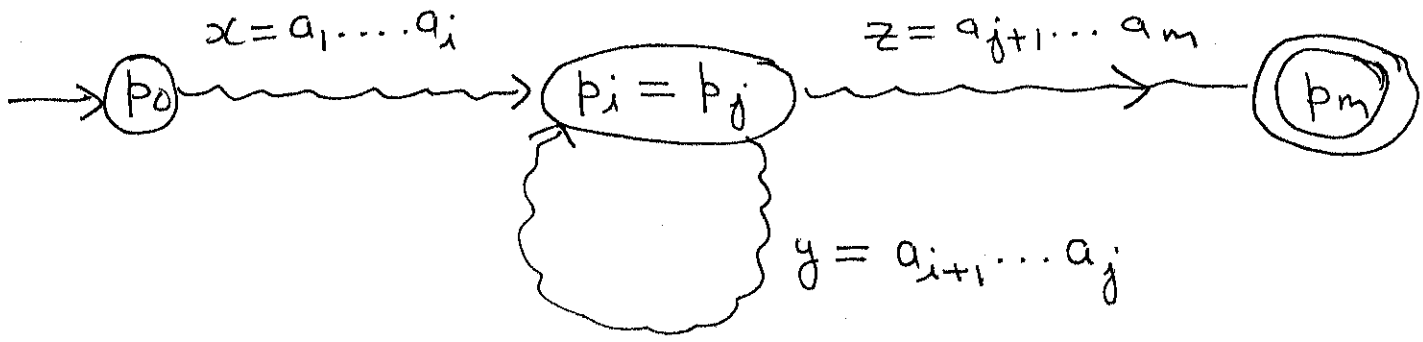
WHICH HAS ONLY n STATES

BY PIGEON-HOLE $p_0, p_1, \ldots p_m$ NOT ALL DISTINCT.

LET $0 \leq i < j \leq n$ BE SUCH THAT

$$p_i = p_j$$

CYCLE



OBSERVE a) $|y| = j - i \geq 1$ (SINCE $i < j$)

b) $|xy| = j \leq n$ (CHOOSE $i < j \leq n$)

c) $\forall k > 0$, $x y^k z \in L(M)$ (PUMPING CYCLE)

DONE

REMARK COMPUTERS HAVE FIXED/BOUNDED MEMORY AND ARE LIKE DFA'S, SO CANNOT DO UNBOUNDED COUNTING AND ONLY ACCEPT REGULAR LANGUAGES.

BUT SINCE MEMORY IS REALLY LARGE & CAN ADD MORE MEMORY ON DEMAND, WE CHOOSE TO VIEW AS "INFINITE" MEMORY IN LATER ABSTRACTIONS.

CLOSURE PROPERTIES

THEOREM IF L_1, L_2 ARE REGULAR, THEN SO ARE

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

PROOF REGULAR L_1, L_2 MUST HAVE R.E. R_1, R_2 SUCH THAT

$$L_1 = L(R_1)$$
$$L_2 = L(R_2)$$

THEN

- $L_1 \cup L_2 = L(R_1) \cup L(R_1) = L(R_1 + R_2)$
- $L_1 \cdot L_2 = L(R_1) \cdot L(R_2) = L(R_1 \cdot R_2)$
- $L_1^* = L(R_1)^* = L(R_1^*)$

MUST ALL BE REGULAR AS THEY ARE GENERATED BY R.

THUS CLASS OF REGULAR LANGUAGES IS CLOSED UNDER OPERATIONS $\cup, \cdot, *$ —— IN THAT, WE CANNOT CREATE LANGUAGES OUTSIDE THIS CLASS BY APPLYING THE OPERATIONS TO LANGUAGES INSIDE THE CLASS.

REMARK IN ABOVE PROOF, WE USED R.E. REPRESENTATION OF REGULAR LANGUAGES — BUT IN GENERAL WE CAN USE ANY THAT IS CONVENIENT:

DFA | NFA | ϵ-NFA | R.E.

COMPLEMENT

DEFN $\overline{L} = \Sigma^* - L$

THEOREM REGULAR LANGUAGES ARE CLOSED UNDER COMPLEMENTATION.

PROOF CONSIDER REGULAR L

ASSUME DFA $M = (Q, \Sigma, \delta, q_0, F)$ ACCEPTS L

CONSTRUCT DFA $\overline{M} = (Q, \Sigma, \delta, q_0, Q - F)$

CLEARLY $L(\overline{M}) = \overline{L(M)} = \overline{L}$

THUS $\overline{L}$ HAS DFA AND IS REGULAR.

INTERSECTION

THEOREM L_1, L_2 REGULAR $\Longrightarrow$ $L_1 \cap L_2$ REGULAR

PROOF DE MORGAN'S LAW $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

USE CLOSURE UNDER $\cup$ & $\overline{}$

QUESTION DO YOU SEE HOW TO MAKE THIS PROOF CONSTRUCTIVE?

REMARK SEE BOOK FOR DIRECT CONSTRUCTION ON DFAS — INTERESTING IDEA OF A PRODUCT OF 2 DFAS.

REVERSAL

DEFN

- $\epsilon^R = \epsilon$
- $w^R = (a_1 a_2 \cdots a_n)^R = a_n a_{n-1} \cdots a_1$
- $L^R = \{ w^R \mid w \in L \}$

THEOREM REGULAR LANGUAGES ARE CLOSED UNDER REVERSA

PROOF EXTEND REVERSAL TO R.E., INDUCTIVELY

BASIS

$$\begin{cases} \epsilon^R = \epsilon \\ \emptyset^R = \emptyset \\ a^R = a \quad \forall a \in \Sigma \end{cases}$$

INDUCTION

$$\begin{cases} (E_1 + E_2)^R = E_1^R + E_2^R \\ (E_1 \cdot E_2)^R = E_2^R E_1^R \\ (E_1^*)^R = (E_1^R)^* \end{cases}$$

VERIFY VIA INDUCTION, IN EACH STEP ABOVE,

$$L(E^R) = L(E)^R$$

EXAMPLE

$$E = abc + bc^*a$$

$$\Rightarrow E^R = cba + ac^*b.$$

QUESTION THINK ABOUT REVERSING AN ϵ-NFA!

SEE BOOK FOR OTHER OPERATIONS:

— HOMOMORPHISMS

— INVERSE HOMOMORPHISMS

DECISION PROBLEMS

FIX ANY PROPERTY P OF LANGUAGES

INPUT REGULAR LANGUAGE L

OUTPUT DOES L HAVE PROPERTY P? — YES/NO.

DECISION ALGORITHM

INPUT ANY REPRESENTATION OF L (DFA/NFA/ϵ-NFA/R.E.)
— CAN EASILY CONVERT BETWEEN THESE.

OUTPUT YES/NO.

MUST ALWAYS TERMINATE IN FINITE TIME WITH CORRECT ANSWER.

EMPTINESS

PROBLEM DECIDE IF A REGULAR LANGUAGE L IS EMPTY.

ASSUME WE CONVERT REPRESENTATION OF L INTO R.E. R.

ALGORITHM RECURSIVELY COMPUTES BOOLEAN PREDICATE

$$\text{EMPT}(R) \begin{cases} \text{TRUE} \\ \text{FALSE} \end{cases}$$

RECURSIVE ALGORITHM

BASIS
$$\begin{cases} \text{EMPTY}(\emptyset) = \text{TRUE} \\ \text{EMPTY}(\epsilon) = \text{FALSE} \\ \text{EMPTY}(a) = \text{FALSE}, \ \forall a \in \Sigma \end{cases}$$

INDUCTION
$$\begin{cases} \text{EMPTY}(R^*) = \text{FALSE} \\ \text{EMPTY}(R+S) = \text{EMPTY}(R) \wedge \text{EMPTY}(\\ \text{EMPTY}(R \cdot S) = \text{EMPTY}(R) \vee \text{EMPTY}(\end{cases}$$

ALTERNATIVE USE DFA REPRESENTATION AND CHECK IF ANY STATE IN F IS REACHABLE FROM q_0 (SEE BOOK).

MEMBERSHIP

PROBLEM GIVEN $w \in \Sigma^*$ AND $L \subseteq \Sigma^*$, WHERE L IS REGULAR, DOES w BELONG TO L?

ALGORITHM ASSUME DFA M FOR L, AND SIMULATE M ON w.

SEE BOOK ON SIMULATING NFA/ϵ-NFA DIRECTLY ON w, WITHOUT FIRST CONVERTING TO DFA, FOR A POSSIBLE INCREASE IN EFFICIENCY.

EQUALITY

GIVEN REGULAR LANGUAGES L_1 AND L_2

DETERMINE WHETHER $L_1 = L_2$

IDEA **CONSIDER** $L = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$

CLEARLY L IS REGULAR (VIA CLOSURE PROPERTIES)

ALSO $L_1 = L_2 \iff L = \emptyset$

ALGORITHM
1) COMPUTE L (USING CLOSURE CONSTRUCTION)
2) COMPUTE R.E. FOR L, SAY R
3) COMPUTE EMPTY(R).

SEE BOOK FOR DFA-BASED ALGORITHM

FINITENESS

PROBLEM GIVEN REGULAR L, IS $|L|$ FINITE?

ASSUME DFA M FOR L HAS $|Q| = n$.

THEOREM L IS INFINITE IF AND ONLY IF

$\exists\, w \in L$ SUCH THAT $n \leq |w| < 2n$.

ALGORITHM $\forall\, w \in \Sigma^*$ WITH $n \leq |w| < 2n$, TEST MEMBERSHIP OF w IN L.

PROOF ($\Leftarrow$) GIVEN $w \in L$ WITH $n \leq |w|$

APPLY P.L. $w = xyz$ WITH $y \neq \epsilon$

SUCH THAT $\forall\, k \geq 0$, $xy^kz \in L$

THUS L IS INFINITE.

PROOF ($\Rightarrow$) SUPPOSE L IS INFINITE

CLEARLY $\exists\, w \in L$, SUCH THAT $|w| \geq n$

CHOOSE $\hat{w}$ TO BE THE SHORTEST STRING IN L OF LENGTH AT LEAST n.

CLAIM $|\hat{w}| < 2n$

SUPPOSE $|\hat{w}| \geq 2n$

BY P.L. $\hat{w} = xyz$ WITH $|xy| \leq n$, $|y| \geq 1$ AND

$xy^0z = xz \in L$

BUT (a) $|xz| \geq n$ SINCE $|xz| = |\hat{w}| - |y| \geq 2n - n =$

(b) $|xz| < |\hat{w}|$ SINCE $|y| \geq 1$

THIS CONTRADICTS CHOICE OF $\hat{w}$ AS SHORTEST SUCH STRI

THEREFORE $\exists\, \hat{w} \in L$, WITH $n \leq |\hat{w}| < 2n$ DONE

DFA MINIMIZATION

GIVEN DFA $M = (Q, \Sigma, \delta, q_0, F)$

FIND DFA M' WITH FEWEST POSSIBLE NUMBER OF STATES, SUCH THAT $L(M') = L(M)$.

IDEA PARTITION Q INTO EQUIVALENCE CLASSES & COLLAPSE

EQUIVALENCE $p \equiv q$ IF FOR ALL $w \in \Sigma^*$,

$$\delta(p,w) \in F \iff \delta(q,w) \in F$$

CLASSES? PARTITION $Q = C_1 \cup C_2 \cup \cdots \cup C_k$ SUCH THAT FOR ALL $p \in C_i$, $q \in C_j$,

$$p \equiv q \implies i = j$$

FINDING C_i? IDEA — IDENTIFY INEQUIVALENT PAIRS (p,q)

OBSERVE "PROOF" OF INEQUIVALENCE FOR p,q IS $w \in \Sigma^*$ SUCH THAT ONE OF $\delta(p,w)$ AND $\delta(q,w)$ IS FINAL AND THE OTHER IS NON-FINAL

PROOF'S LENGTH $|w| = m$.

$$\left.\begin{array}{l} p \xrightarrow{w_1} p_1 \xrightarrow{w_2} p_2 \xrightarrow{w_3} \cdots \xrightarrow{w_{m-1}} p_{m-1} \xrightarrow{w_m} p_m \\ q \xrightarrow{w_1} q_1 \xrightarrow{w_2} q_2 \xrightarrow{w_3} \cdots \xrightarrow{w_{m-1}} q_{m-1} \xrightarrow{w_m} q_m \end{array}\right\} \text{ONE IS FINA BUT NOT THE OTHER.}$$

NOTE $w_{i+1} \cdots w_m$ IS LENGTH $n-i$ PROOF OF INEQUIV. OF p_i, q_i

---

DEFN SAY $p \equiv_i q$ IF FOR $|w| \le i$

$$\delta(p,w) \in F \iff \delta(q,w) \in F$$

MEANS NO INEQUIVALENCE PROOF OF LENGTH $\le i$

CLEARLY $p \not\equiv_{i+1} q$ IF & ONLY IF FOR SOME $a \in \Sigma$

$$\delta(p,a) \not\equiv_i \delta(q,a)$$

ALGORITHM (TO RECURSIVELY DETERMINE $\equiv_i$)

STEP 0 PARTITION $Q = C_1 \cup C_2$, WITH $C_1 = F$, $C_2 = Q - F$

(SINCE $p \not\equiv_0 q$ IF & ONLY IF ONE IS FINAL, BUT NOT BOTH

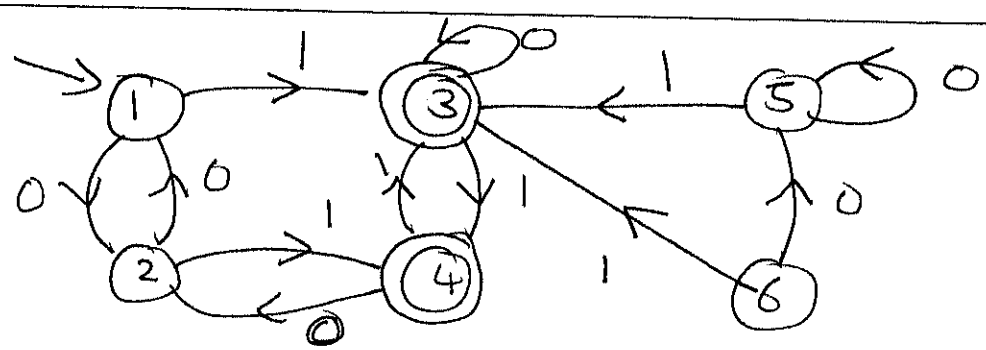
STEP $i+1$ DECIDE $p \equiv_{i+1} q$ IF & ONLY IF $\forall a \in \Sigma$,

$$\delta(p,a) \equiv_i \delta(q,a)$$

OBTAIN REFINED PARTITION $C_1, C_2, C_3, \ldots$

TERMINATE WHEN STOP DISCOVERING NEW CLASSES, IN AT MOST $|Q|$ STEPS.

EXAMPLE



ALGORITHM

STEP 0 $C_1 = \{1,2,5,6\}$ $C_2 = \{3,4\}$

STEP 1 $C_1 = \{1,2,5,6\}$ $C_2 = \{3\}$ $C_3 = \{4\}$

STEP 2 $C_1 = \{1,5,6\}$ $C_2 = \{2\}$ $C_3 = \{3\}$ $C_4 = \{4\}$

STEP 3 $C_1 = \{1\}$ $C_2 = \{2\}$ $C_3 = \{3\}$ $C_4 = \{4\}$

$C_5 = \{5,6\}$

STEP 4 NO CHANGE — TERMINATE

CONSTRUCTING M'

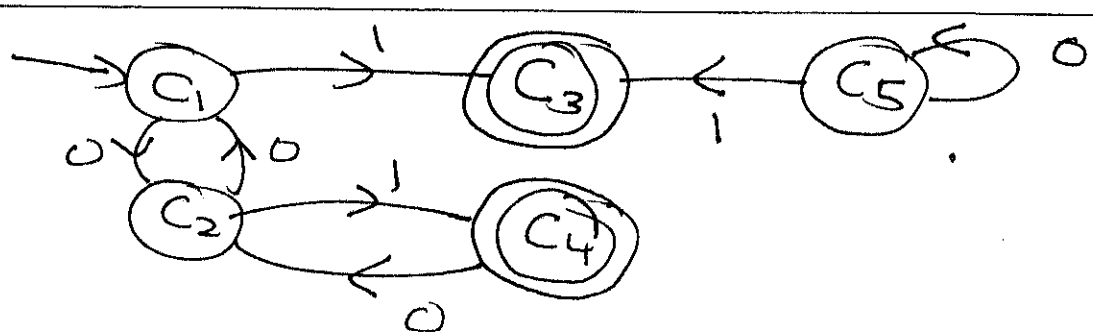
1) GET PARTITION $Q = C_1 \cup C_2 \cup \cdots C_k$ IN M
2) CONSTRUCT M'
   - STATES $Q' = \{C_1, C_2, \ldots, C_k\}$
   - TRANSITIONS IF $\delta(p, a) = q$ IN M
     THEN ADD $\delta(C_i, a) = C_j$ TO M'
     WHERE $p \in C_i$, $q \in C_j$
   - START STATE C_i CONTAINING q_0 FROM M
   - FINAL STATES ANY C_j CONTAINING A FINAL STATE FROM M.

VERIFY EACH STEP WORKS GIVEN NOTION OF EQUIVALENCE

E.G. IN DEFINING TRANSITIONS, ALL STATES IN C_i WILL GO TO A STATE IN C_j ON INPUT a

OR IF C_j CONTAINS FINAL STATE, THEN ALL ITS STATES ARE FINAL.

EXAMPLE



OBSERVE C_5 IS NOT REACHABLE FROM START STATE AND MUST BE REMOVED!

MYHILL-NERODE THM ABOVE PROCESS GIVES SMALLEST POSSIBLE DFA FOR A LANGUAGE — AND THIS IS UNIQUE