COURSE OVERVIEW

GOAL I DEVELOP ABSTRACT, MATHEMATICAL MODELS OF COMPUTATION, SUCH AS:

- FINITE STATE MACHINES
- PUSH DOWN MACHINES
- TURING MACHINES

NOTE PHYSICALLY REALIZABLE COMPUTERS CORRESPOND TO THE SIMPLEST SUCH MODEL — F.S.M.

HOWEVER REST ARE USEFUL ABSTRACTIONS OF MACHINES WE COULD BUILD IN PRINCIPLE, GIVEN UNLIMITED RESOURCES

GOAL II UNDERSTAND PROPERTIES & POWER OF THESE MODELS, ESPECIALLY IN TERMS OF THEIR ABILITY TO SOLVE COMPUTATIONAL PROBLEMS.

QS. A WHAT PROBLEMS CAN THEY SOLVE AT ALL?

QS. B WHAT PROBLEMS CAN THEY SOLVE EFFICIENTLY?

HISTORY

1930s ALAN TURING DEFINED MACHINES MORE POWERFUL THAN ANY IN EXISTENCE, OR EVEN ANY THAT WE COULD IMAGINE — GOAL WAS TO ESTABLISH THE BOUNDARY BETWEEN WHAT WAS AND WAS NOT COMPUTABLE

1940s/50s IN AN ATTEMPT TO MODEL "BRAIN FUNCTION" RESEARCHERS DEFINED FINITE STATE MACHINES

LATE 1950s LINGUIST NOAM CHOMSKY BEGAN THE STUDY OF FORMAL GRAMMARS

WE SEE A CONVERGENCE OF ALL THIS INTO A FORMAL THEORY OF COMPUTER SCIENCE, WITH VERY DEEP PHILOSOPHICAL IMPLICATIONS AS WELL AS PRACTICAL APPLICATIONS (COMPILERS, WEB SEARCHING, HARDWARE, A.I., ALGORITHM DESIGN, SOFTWARE ENGG, ...)

<u>CULMINATION</u> IN 1970s, STEVE COOK EXTENDED ALL THIS TO THE "THEORY OF NP-COMPLETENESS" WHICH SEPARATED OUT A CLASS (IN FACT, MOST INTERESTING PROBLEMS IN PRACTICE) WHICH:

- COULD BE SOLVED, IN PRINCIPLE
- BUT CANNOT BE <u>EFFICIENTLY</u> SOLVED, EVEN GIVEN <u>MOORE'S LAW</u> FOR HARDWARE.

<u>HIDDEN AGENDA</u> TO TEACH YOU HOW TO THINK <u>PRECISELY</u> AND DEVELOP POWERS OF REASONING IN A PRECISE / FORMAL / ABSTRACT FASHION.

<u>KEY</u> THIS IS WHAT SEPARATES MERE <u>PROGRAMMERS</u> FROM COMPUTER <u>SCIENTISTS</u>

<u>PRECISION IS KEY</u> COMPUTERS | PROGRAMS ARE VERY UNFORGIVING OF "FUZZY" THINKING.

<u>AT THE SAME TIME</u> EVERYTHING YOU LEARN WILL BE EMINENTLY PRACTICAL AND USEFUL IN REAL LIFE

<u>EXCEPT</u> YOU MAY NOT REALIZE THIS TILL A YEAR OR TWO FROM NOW!

<u>VERY IMPORTANT.</u> REVIEW VARIOUS TYPES OF PROOFS IN CHAPTER 1 — THIS IS CRITICAL TO UNDERSTANDING THE MATERIAL PRESENTED IN CLASS.

EXAMPLES APPLICATIONS OF F.S.M.

1) SOFTWARE FOR DESIGNING | VERIFYING DIGITAL CIRCUIT.
2) LEXICAL ANALYZERS OF COMPILERS
3) SCANNING/SEARCHING LARGE BODIES OF TEXT (WEB SEARCH ENGINES, GREP, NAPSTER, ...)
4) DESIGN/VERIFICATION/IMPLEMENTATION OF SOFTWARE SYSTEMS INVOLVING INTERACTION (e.g. NETWORK PROTOCOLS, ELECTRONIC COMMERCE, ...)

⋮

<u>PROBLEMS?</u>

<u>THIS COURSE</u> WE TAKE THE FOLLOWING VIEW OF A PROBLEM.

<u>LANGUAGE</u> L — SET OF STRINGS

<u>INPUT</u> STRING x

<u>PROBLEM</u> DECIDE WHETHER $x \in L$ OR NOT.

<u>MACHINE</u> SOLVES PROBLEM L BY ACCEPTING / REJECTING x.

<u>REMARK</u> WHILE REAL-LIFE PROBLEMS ARE OFTEN NOT A SIMPLE LANGUAGE RECOGNITION PROBLEM, WE CAN CAPTURE THEIR ESSENTIAL STRUCTURE IN SOME L.

<u>EXAMPLE</u> $L =$ VALID C PROGRAMS

<u>IN COMPILER COURSE</u> YOU WILL SEE THAT A MACHINE WHICH CAN CHECK "LEGALITY" OR "VALIDITY") OF A PROGRAM, CAN BE ADAPTED TO GENERATE "OBJECT CODE") IN THE PROCESS OF DOING SO.

REVIEW OF BASIC DEFINITIONS

ALPHABET ANY FINITE SET OF SYMBOLS — Σ

- $\Sigma = \{0,1\}$
- $\Sigma = \{a, b, c, \dots z\}$
- $\Sigma =$ ASCII CHARACTERS.

STRING FINITE SEQUENCE OF SYMBOLS FROM Σ

$$w = w_1 \dots w_n \quad \text{WHERE} \quad \forall i,\ w_i \in \Sigma$$

EXAMPLES

- 1001
- string
- \$1,000,000

SPECIAL STRINGS

ϵ EMPTY STRING

$\cancel{b}$ BLANK OR SPACE

LENGTH $|w| = |w_1 \dots w_n| = n$

- $|\epsilon| = 0$
- $|\cancel{b}| = 1$
- $|\text{string}| = 6$.

CARTESIAN PRODUCT.

$$\Sigma^k = \underbrace{\Sigma \times \Sigma \times \dots \times \Sigma}_{k \text{ TIMES}}$$

MEANS ALL STRINGS OF LENGTH k FROM Σ

DEFN — CLOSURE $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots$

MEANS ALL FINITE-LENGTH STRINGS FROM Σ

BUT Σ^* ITSELF IS INFINITE.

$$\Sigma^0 = \{\epsilon\}$$
$$\Sigma^1 = \Sigma$$

EXAMPLE

$$\Sigma = \{0,1\}$$
$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$$

CONCATENATION

$$x = x_1 \ldots x_n \in \Sigma^*$$
$$y = y_1 \ldots y_m \in \Sigma^*$$

$$\Longrightarrow \quad x.y = x_1 \ldots x_n y_1 \ldots y_m$$

NOTE $\epsilon x = x \epsilon = x$

NOTE $|x.y| = |x| + |y|$

LANGUAGE L ANY COLLECTION OF STRINGS $L \subseteq \Sigma^*$

SUBTLE POINT L, Σ^* ARE POTENTIALLY INFINITE IN SIZE, BUT CONTAIN ONLY FINITE-LENGTH STRINGS

EXAMPLES

$$\begin{cases} \Sigma = \{a, b, c, \ldots, z\} \\ L = \text{ALL ENGLISH WORDS} \end{cases}$$

$$\begin{cases} \Sigma = \{0,1\} \\ L = \{\epsilon, 01, 0011, 000111, \ldots\} \end{cases}$$

ALL STRINGS WITH EQUAL # OF 0'S & 1'S, BUT WITH 0'S PRECEDING THE 1'S.

$$\begin{cases} \Sigma = \text{ASCII} \\ L = \text{COMPILABLE C PROGRAMS} \end{cases}$$

FINITE STATE MACHINES

FINITE AUTOMATA

- SIMPLEST MODEL OF COMPUTATION
- DESCRIBES CLASS OF LANGUAGES CALLED "REGULAR"
- OPERATION
    - ALWAYS IN ONE OF FINITELY-MANY STATES
    - CHANGES STATE IN RESPONSE TO INPUT
    - ACCEPTS INPUT BY ENDING UP IN ONE OF SO-CALLED FINAL OR ACCEPTING STATES

EXAMPLE F.A. BELOW SCANS HTML DOCUMENTS, LOOKING FOR A LIST OF WHAT COULD BE TITLE-AUTHOR PAIRS, PERHAPS IN READING LIST FOR SOME COURSE.

NOTE

- ACCEPTS WHEN IT FINDS END OF A LIST
- OBSERVE STRINGS THAT MATCHED THE TITLE (BEFORE ƀ BY ƀ) AND AUTHOR (AFTER ƀ BY ƀ) WOULD BE STORED IN A TABLE OF SUCH PAIRS BEING ACCUMULATED.
- IN HTML <OL> — NUMBERED / ORDERED LIST
  <OL> — UNNUMBERED / UNORDERED LIST.

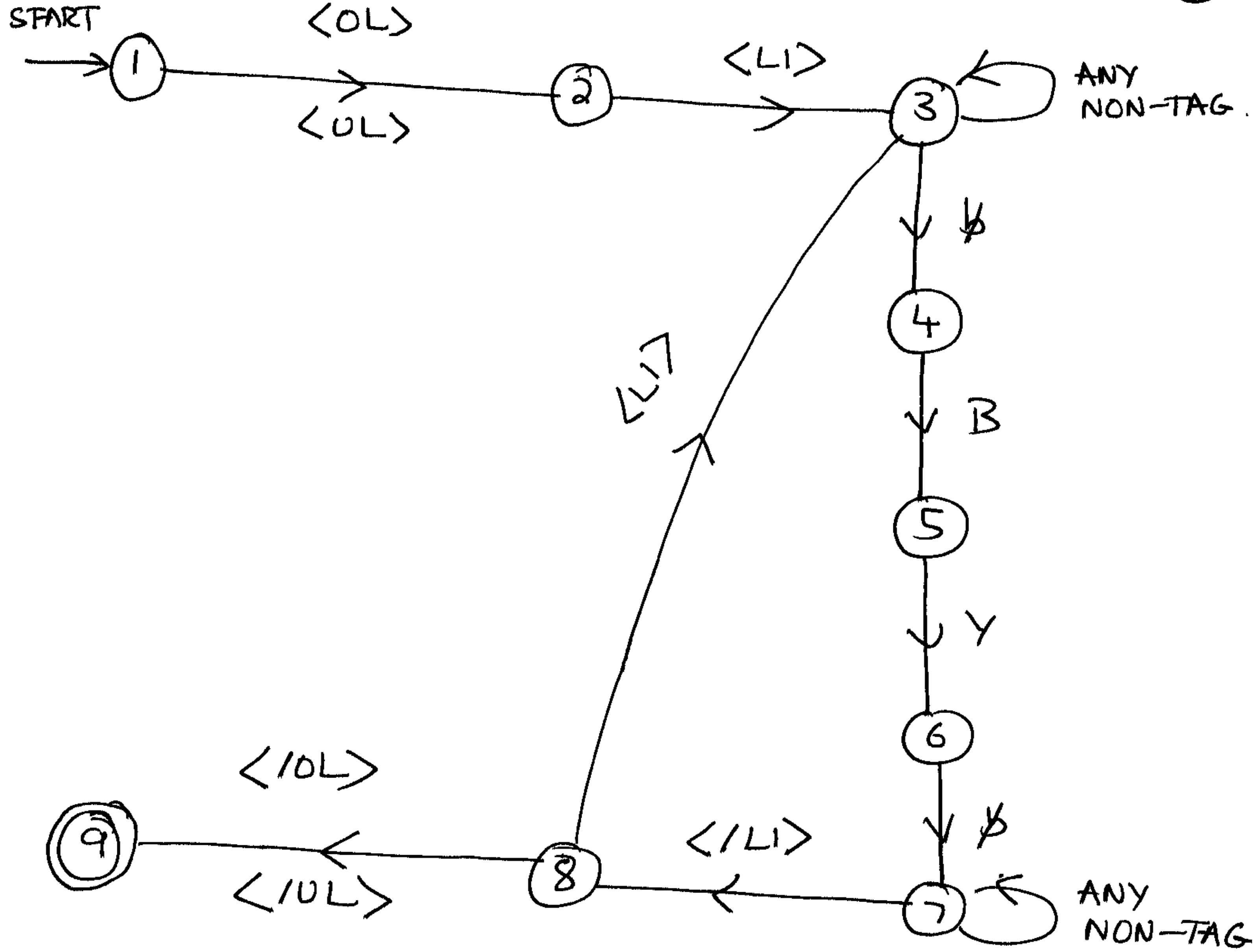
EXAMPLE

<OL>

<LI> OTHELLO ƀ BY ƀ SHAKESPEARE <ILI>

<LI> FOUNDATION ƀ BY ƀ ASIMOV </LI>

</OL>



NOTATION STATE TRANSITION DIAGRAM

- STATE (6)
- START STATE → (1) (ALSO CALLED INITIAL STATE)
- FINAL STATE ((9)) (ALSO CALLED ACCEPTING)
- TRANSITION $(4) \xrightarrow{B} (5)$

MEANS IN STATE 4, IF THE F.A. SEES "B" IN THE INPUT, IT MOVES TO STATE 5.

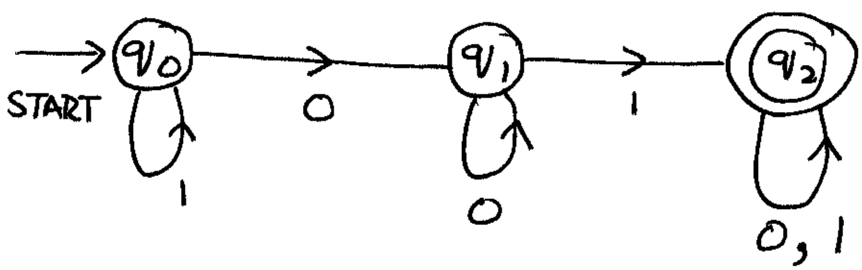
EXAMPLE

CONSIDER PROBLEM OF CHECKING WHETHER A BINARY STRING ω CONTAINS THE PATTERN 01.

HERE
$$\Sigma = \{0,1\}$$
$$L = \{\omega \in \Sigma^* \mid \omega \text{ HAS SUBSTRING } 01\}$$
$$= \{x01y \mid x, y \in \Sigma^*\}$$

NOTE
$$\begin{cases} 11010 \in L \\ 000111 \in L \\ 111000 \notin L \end{cases}$$

F.A.



MEANING
q_0: WAITING FOR FIRST 0
q_1: SEEN 0, WAITING FOR 1
q_2: SEEN 01, WAITING FOR END OF INPUT.

OBSERVE F.A. SCANS INPUT ω IN L-TO-R ORDER (CANNOT BACK UP!), SYMBOL-BY-SYMBOL, MAKING STATE TRANSITIONS

ACCEPTS IF IT IS IN ACCEPTING/FINAL STATE WHEN IT REACHES THE END OF THE INPUT

— ELSE IT REJECTS THE INPUT.

NOTE $L = \{\omega \in \Sigma^* \mid \text{F.A. ACCEPTS } \omega\}$

FORMALLY F.A. $M = (Q, \Sigma, \delta, q_0, F)$ WHERE:

1) **Q** FINITE SET OF STATES

   [E.G. $Q = \{q_0, q_1, q_2\}$]

2) **Σ** INPUT ALPHABET

   [E.G. $\Sigma = \{0, 1\}$]

3) **q_0** INITIAL OR START STATE, $q_0 \in Q$

4) **F** SET OF FINAL OR ACCEPTING STATES, $F \subseteq Q$

   [E.G. $F = \{q_2\}$]

5) **δ** STATE TRANSITION FUNCTION.

TRANSITION FUNCTION $\delta : Q \times \Sigma \longrightarrow Q$

HERE $\delta(q, a) = p$ IS SAME AS $q \xrightarrow{a} p$

CAN REPRESENT AS

- DIAGRAM
- TRANSITION TABLE

E.G.

δ	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

EXTENDED TRANSITION FUNCTION

GOAL EXTEND δ TO MULTIPLE TRANSITIONS

IDEA

- δ: SINGLE TRANSITION ON INPUT SYMBOL $a \in \Sigma$
- $\hat{\delta}$: SEQUENCE OF TRANSITIONS ON SUBSTRING $x \in \Sigma^*$

FORMALLY $\hat{\delta} : Q \times \Sigma^* \longrightarrow Q$

MEANING $\hat{\delta}(q, x) = p$ DENOTES THAT STARTING AT STATE q, PORTION x OF INPUT STRING WILL TAKE F.A. TO STATE p.

EXAMPLE

$$\begin{cases} \hat{\delta}(q_0, 111) = q_0 \\ \hat{\delta}(q_0, 11100) = q_1 \\ \hat{\delta}(q_0, 1110011) = q_2 \end{cases}$$

$\delta(p_0, a_1) = p_1$; $\delta(p_1, a_2) = p_2$; $\ldots$ $\delta(p_{n-1}, a_n) = p_n$

$\Longleftrightarrow$

$\delta(p_0, a_1 a_2 \cdots a_n) = p_n$

QUESTION HOW DO WE GET $\hat{\delta}$ FROM δ?

INDUCTIVE DEFN $\forall q \in Q, \forall a \in \Sigma, \forall x \in \Sigma^*,$

BASIS $\hat{\delta}(q, \epsilon) = q$

INDUCTION $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

APPLICATION

$$\hat{\delta}(q_0, \epsilon) = q_0$$

$$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1) = \delta(q_0, 1) = q_0$$

$$\hat{\delta}(q_0, 10) = \delta(\hat{\delta}(q_0, 1), 0) = \delta(q_0, 0) = q_1$$

$$\hat{\delta}(q_0, 101) = \delta(\hat{\delta}(q_0, 10), 1) = \delta(q_1, 1) = q_2$$

FACT $\forall a \in \Sigma,\ \forall q \in Q,$

$$\hat{\delta}(q, a) = \delta(q, a)$$

PROOF

$$\hat{\delta}(q,a) = \delta(\hat{\delta}(q,\epsilon), a) = \delta(q, a).$$

THUS $\hat{\delta}$ AND δ AGREE ON STRINGS OF LENGTH 1, AND δ IS ONLY DEFINED FOR SUCH STRINGS

— CONVENTION CAN CALL $\hat{\delta}$ AS $\hat{\delta}$ WITHOUT ANY CONFUSION.

EXERCISE PROVE THAT

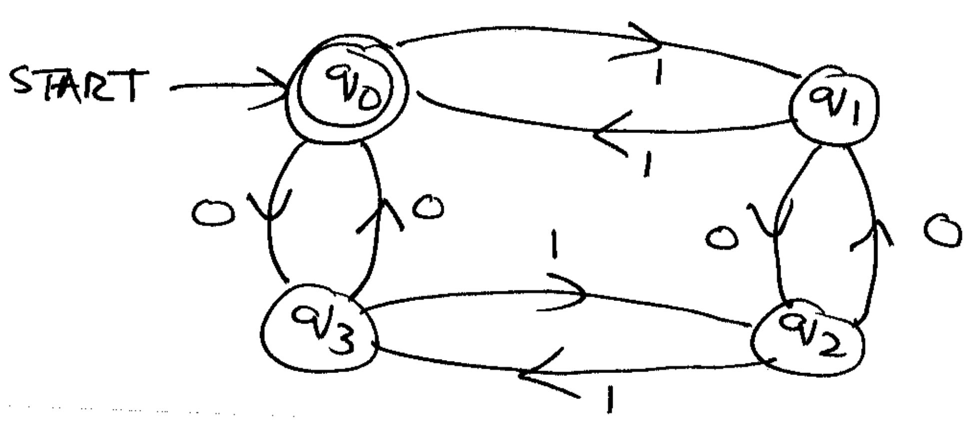
$$\forall q \in Q,\ \forall x, y \in \Sigma^*,$$
$$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q,x), y).$$

LANGUAGE OF F.A. M.

- $M = (Q, \Sigma, \delta, q_0, F)$
- $L(M)$ = ALL STRINGS ACCEPTED BY M

DEFN $L(M) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F \}$

EXERCISE CONSIDER M BELOW — WHAT IS $L(M)$?



NON — DETERMINISM

DETERMINISTIC F.A. (DFA)

- $\delta(q, a)$ IS UNIQUE (EACH q HAS EXACTLY ONE ARROW GOING OUT FOR EACH $a \in \Sigma$)
- MEANS FOR SPECIFIC INPUT w, EXECUTION IS TOTALLY PREDICTABLE & REPEATABLE.

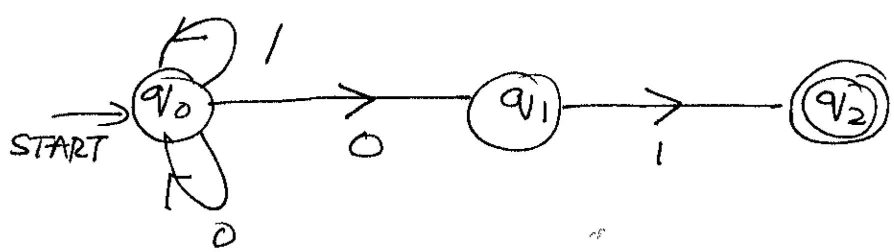
NON-DETERMINISTIC F.A. (NFA)

- $\delta(q, a)$ IS A SET OF STATES
  - EMPTY SET IS POSSIBLE
  - MULTIPLE STATES ARE POSSIBLE
- THUS q COULD HAVE MULTIPLE (OR NO) ARROWS GOING OUT FOR EACH $a \in \Sigma$
- MEANS MULTIPLE CHOICES ALLOWS NFA TO "GUESS" THE RIGHT ACTION, INSTEAD OF HAVING IT HARDWIRED IN ADVANCE.

EXAMPLE $L_{01} = \{ w \mid w \text{ ENDS IN } 01 \}$

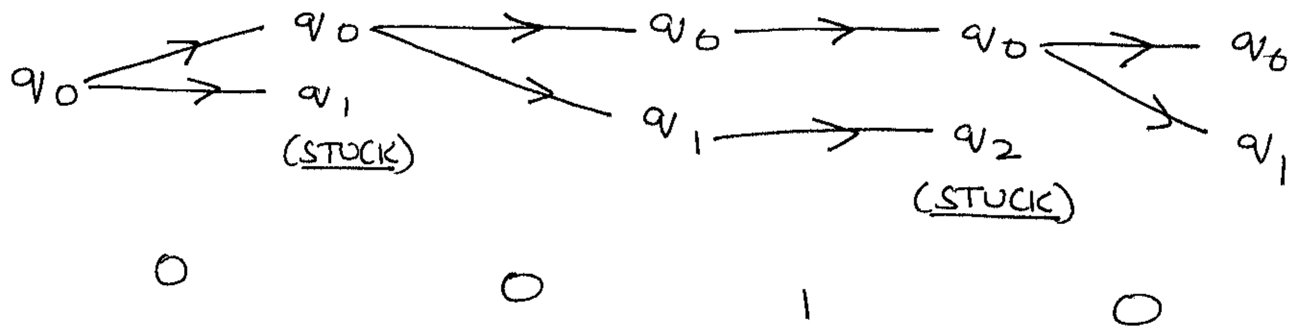
IDEA NFA N "GUESSES" THE END OF INPUT AND THEN LOOKS FOR 01 — USING NONDETERMINISM

N_{01}

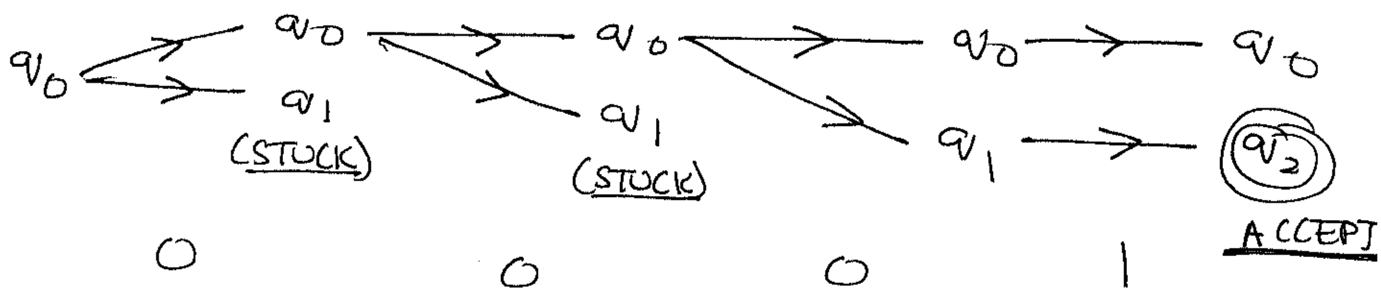


OBSERVE NON-DETERMINISM IMPLIES THAT INSTEAD OF UNIQUE EXECUTION TRACE (AS IN DFA'S), WE HAVE A TREE OF POSSIBLE EXECUTIONS.

EXAMPLE FOR INPUT $w = 0010$



BUT FOR $w = 0001$



ACCEPTANCE WHEN THERE EXISTS AT LEAST ONE EXECUTION PATH ENDS IN A FINAL STATE

REJECTION WHEN ALL POSSIBE EXECUTION PATHS EITHER GET "STUCK" OR END IN A NON-FINAL STATE

INTERPRETATION

VIEW 1 N ALWAYS MAKES THE RIGHT CHOICES TO ENSURE ACCEPTANCE — ASSUMING AN ACCEPTING PATH EXISTS

VIEW 2 IT SPAWNS OFF MULTIPLE COPIES OF ITSELF TO EXPLORE ALL POSSIBLE PATHS

VIEW 3 IT EXPLORES MULTIPLE PATHS IN PARALLEL

CRITICAL POINT AN NFA N FOR LANGUAGE L MUST ENSURE:

$\forall x \notin L$ ALL PATHS ARE REJECTING

$\forall x \in L$ AT LEAST ONE PATH IS ACCEPTING

THUS WHILE N IS FREE TO "GUESS", IT MUST VERIFY ITS GUESSES ARE "CORRECT".

FORMALLY NFA $N = (Q, \Sigma, \delta, q_0, F)$

WHERE EVERYTHING IS SAME AS IN DFA EXCEPT δ

$$\delta : Q \times \Sigma \longrightarrow 2^Q \qquad (2^Q : \text{POWERSET OF } Q)$$

THAT IS $\delta(q, a)$ IS SUBSET OF Q.

ABOVE EXAMPLE N_{01} HAS FOLLOWING δ-TRANSITION TABLE

	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
q_2	ϕ	ϕ

EXTENDING δ TO $\hat{\delta}$ $\hat{\delta}(q, w)$: $\Big\{$ STATES THAT CAN BE REACHED FROM q ON INPUT w

EXAMPLES
- $\hat{\delta}(q_0, 001) = \{q_0, q_2\}$
- $\hat{\delta}(q_0, 000) = \{q_0, q_1\}$

INDUCTIVE DEFN $\forall q \in Q, \forall x \in \Sigma^*, \forall a \in \Sigma$

BASIS $\hat{\delta}(q, \epsilon) = \{q\}$

INDUCTION SUPPOSE $\hat{\delta}(q, x) = \{p_1, p_2, \dots, p_k\}$

SUPPOSE $\delta(p_i, a) = S_i \quad (\forall i = 1, \dots k)$

THEN $\delta(q, wa) = S_1 \cup S_2 \cup \dots \cup S_k$

SHORTHAND $\hat{\delta}(q, xa) = \bigcup_{p_i \in \hat{\delta}(q,x)} \delta(p_i, a)$.

AS IN DFAs FOR $q \in Q$, $a \in \Sigma$

$$\hat{\delta}(q, a) = \delta(q, a)$$

SO CAN DENOTE BOTH BY δ WITHOUT FEAR OF AMBIGUIT

EXAMPLE

- $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 00) = \bigcup_{p_i \in \hat{\delta}(q_0, 0)} \delta(p_i, a) = \delta(q_0, 0) \cup \delta(q_1, 0)$
  $= \{q_0, q_1\} \cup \emptyset$
  $= \{q_0, q_1\}$
- $\hat{\delta}(q_0, 001) = \bigcup_{p_i \in \hat{\delta}(q_0, 00)} \delta(p_i, 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$
  $= \{q_0\} \cup \{q_2\}$
  $= \{q_0, q_2\}$

OBSERVE $\hat{\delta}(q_0, 001)$ CONTAINS FINAL STATE q_2

$\Longrightarrow$ 001 IS ACCEPTED

LANGUAGE OF AN NFA

GIVEN NFA $N = (Q, \Sigma, \delta, q_0, F)$

THEN $L(N) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset \}$

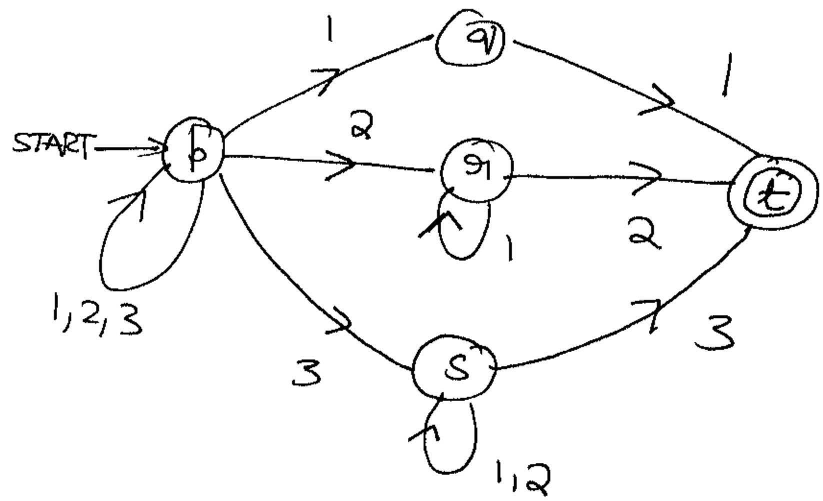
VERIFY THIS IS CONSISTENT WITH EARLIER EXPLANATIONS & 3 VIEWS

$L_{123} \subseteq \{1,2,3\}^*$ (1

LANGUAGE ALL STRINGS $w \in \{1,2,3\}^*$ SUCH THAT THE LA[ST] SYMBOL IN w APPEARS PREVIOUSLY WITHOUT ANY INTERVENING LARGER SYMBOL

E.G.
- … 11
- … 2112
- … 3121213

N_{123}



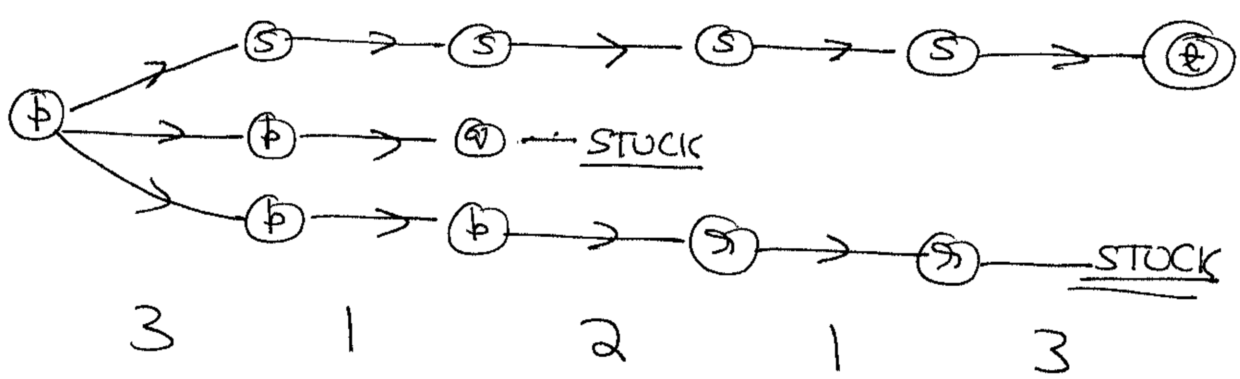
IN p HAVEN'T GUESSED YET

IN s GUESSED LAST SYMBOL IS A 3, AND JUST SAW THE PRECEDING 3 — WAITING TO VERIFY THAT INTERVENING SYMBOLS ARE LESS THAN 3

SIMILARLY q, r.

	1	2	3
p	$\{p,q\}$	$\{p,r\}$	$\{p,s\}$
q	$\{t\}$	$\emptyset$	$\emptyset$
r	$\{r\}$	$\{t\}$	$\emptyset$
s	$\{s\}$	$\{s\}$	$\{t\}$
t	$\emptyset$	$\emptyset$	$\emptyset$

EXAMPLE $w = 31213$



NOTE $3121 \notin L_{123}$.

COMPARING THE POWER OF DFAs & NFAs

POWER? ABILITY TO ACCEPT LANGUAGES

OBSERVE DFA IS A SPECIAL CASE OF NFA WITH $|\delta(q,a)| = 1$

$\Longrightarrow$ POWER (DFA) $\leq$ POWER (NFA).

THEOREM FOR EVERY NFA N THERE IS A DFA M WHICH ACCEPTS EXACTLY THE SAME LANGUAGE

THUS POWER (NFA) = POWER (DFA)

QUESTION THEN WHY BOTHER WITH NFAS, WHICH WE CANT REALLY IMPLEMENT DIRECTLY?

PROOF OF THEOREM CONVERTS AN NFA N WITH k STATES INTO A DFA M WITH 2^k STATES.

— BEST POSSIBLE, IN THAT FOR MANY NFAS NEED SUCH A BLOWUP IN STATES WHEN GETTING DFA

IN GENERAL

NFAs EASIER TO CONSTRUCT | SPECIFY | COMPREHEND
— DUE TO SUCCINCTNESS

DFAs CAN BE IMPLEMENTED IN REAL-LIFE

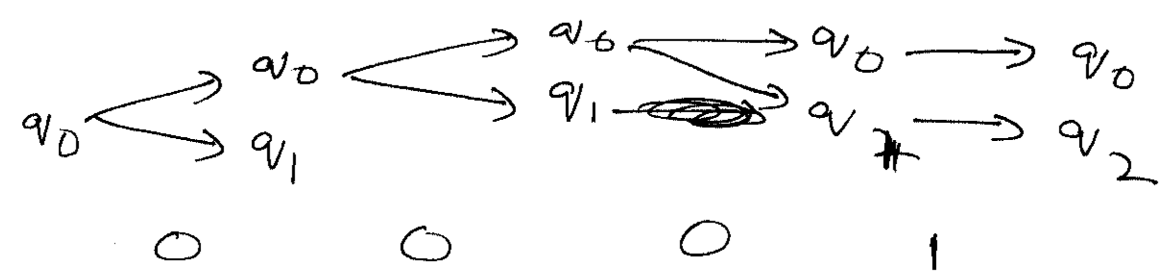
THUS WE USE NFA'S TO CAPTURE PATTERNS IN STRING PROCESSORS (GREP | LEXICAL ANALYZERS)
— BUT CONVERT TO DFAs IN FINDING PATTERNS

THEOREM FOR EVERY NFA N, THERE EXISTS A DFA M WITH

$$L(M) = L(N)$$

PROOF IDEA GIVEN N, M WILL SIMULATE ENTIRE EXECUTION TREE OF N IN ONE EXECUTION

IN N



IN M

$$\{q_0\} \xrightarrow{0} \{q_0, q_1\} \xrightarrow{0} \{q_0, q_1\} \xrightarrow{0} \{q_0, q_1\} \xrightarrow{1} \{q_0, q_2\}$$

TRICK A STATE IN M WILL CORRESPOND TO A SUBSET OF N'S STATES

FORMALLY **GIVEN** $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$

CONSTRUCT $M = (Q_M, \Sigma, \delta_M, \{q_0\}, F_M)$

SUCH THAT

$$Q_M = 2^{Q_N} \quad \left(\begin{array}{l}\text{ALL SUBSETS OF} \\ \text{THE SET } Q_N\end{array}\right)$$

$$F_M = \{ S \subseteq Q_N \mid S \cap F_N \neq \emptyset \}$$

WHAT ABOUT δ_M?

$$\delta_M(\{p_1, \ldots p_k\}, a) = \delta_N(p_1, a) \cup \delta_N(p_2, a) \cup \cdots \cup \delta_N(p_k,$$

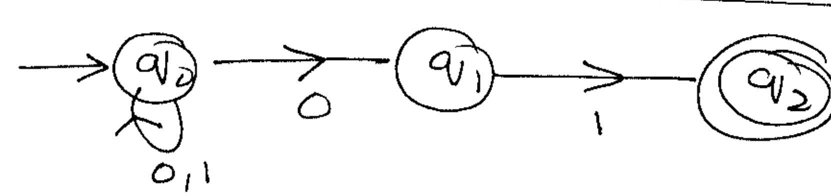
OR

$$\delta_M(S, a) = \bigcup_{p_i \in S} \delta_N(p_i, a).$$

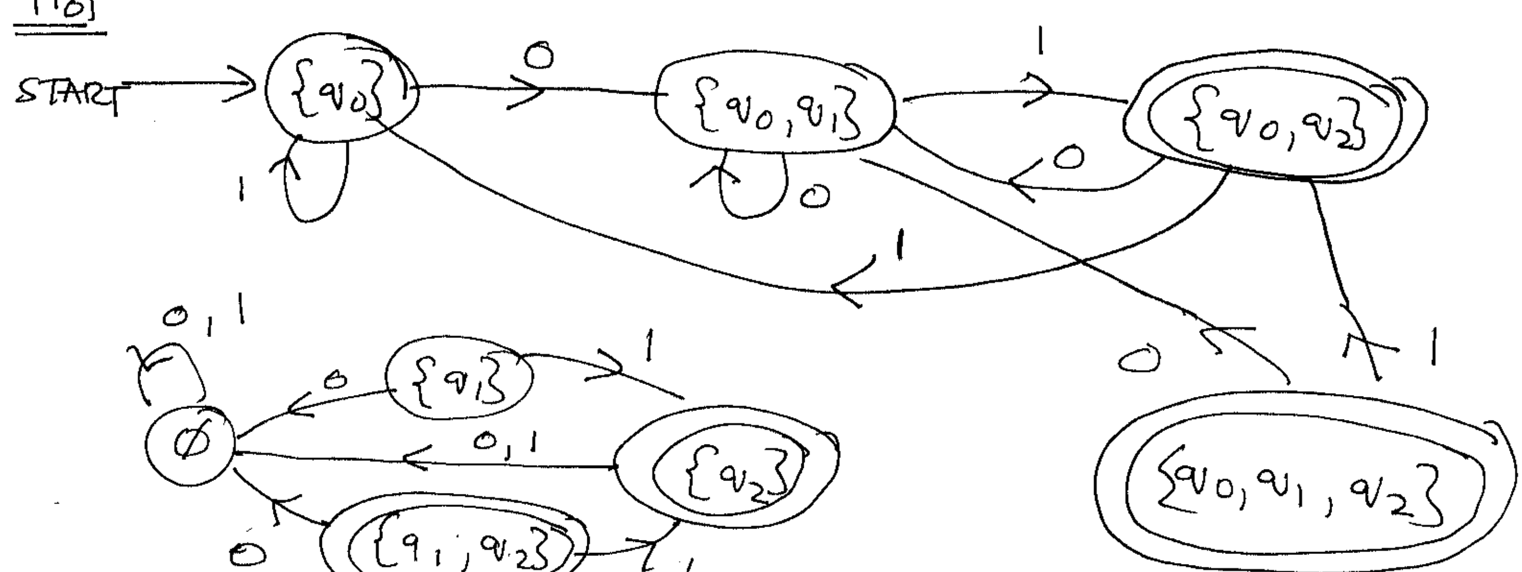
MEANS $\delta_M(S, a)$ IS THE SET OF STATES IN N REACHABLE FROM $p_i \in S$ ON INPUT SYMBOL a.

EXAMPLE

N_{01}



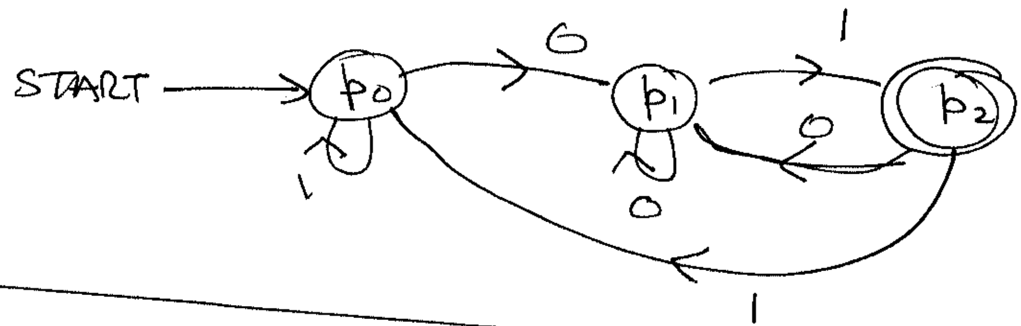
M_{01}



NOTE SOME STATES CAN'T BE REACHED FROM START STATE & HENCE CAN BE ELIMINATED AS BEING NON-ESSENTIAL STATES

DEAD STATE $\emptyset$ CAN NEVER LEAVE IT (STUCK)

SIMPLIFIED DFA



LEMMA $\forall q \in Q_N, \forall w \in \Sigma^*,$

$$\hat{\delta}_M(\{q\}, w) = \hat{\delta}_N(q, w)$$

PROOF BY INDUCTION ON LENGTH OF w.

BUT FIRST LETS FINISH PROOF OF THEOREM.

$$L(M) = \{ w \in \Sigma^* \mid \hat{\delta}_M(\{q_0\}, w) \in F_M \}$$

$$= \{ w \in \Sigma^* \mid \hat{\delta}_M(\{q_0\}, w) \cap F_N \neq \emptyset \}$$

(BY DEFN OF F_M)

$$= \{ w \in \Sigma^* \mid \hat{\delta}_N(q_0, w) \cap F_N \neq \emptyset \}$$

(BY LEMMA)

$$= L(N)$$

(BY DEFN OF $L(N)$)

DONE

PROVING LEMMA [INDUCTION ON LENGTH OF w]

BASIS [$|w| = 0$ OR $w = \epsilon$]

$$\hat{\delta}_M(\{q\}, \epsilon) = \bigcup_{p \in \{q\}} \hat{\delta}_N(q, \epsilon) = \hat{\delta}(q, \epsilon)$$

INDUCTION ASSUME LEMMA FOR $|w| = n-1$

SHOW FOR $|w| = n$.

CONSIDER $|w| = n$ AND WRITE $w = x.a$

($|x| = n-1$, $|a| = 1$)

CLEARLY $\hat{\delta}_M(\{q\}, x) = \hat{\delta}_N(q, x)$

(BY I.H.)

NOW
$$\hat{\delta}_M(\{q\}, w) = \hat{\delta}_M(\{q\}, xa) \quad [w = xa]$$

$$= \delta_M(\hat{\delta}_M(\{q\}, x), a)$$

[DEFN OF $\hat{\delta}$]

$$= \delta_M(\hat{\delta}_N(q, x), a)$$

[BY I.H.]

$$= \bigcup_{p_i \in \hat{\delta}_N(q, x)} \delta_N(p_i, a)$$

[DEFN OF δ_M]

$$= \delta_N(\hat{\delta}_N(q, x), a)$$

[DEFN OF δ_N]

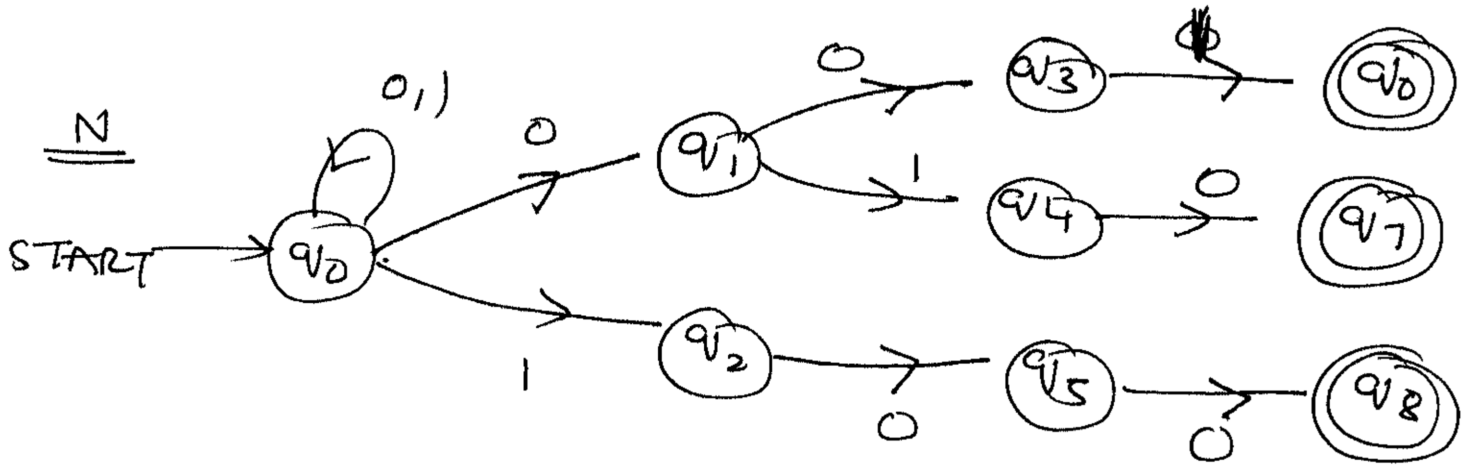
$$= \hat{\delta}_N(q, xa) \quad [\text{DEFN OF } \hat{\delta}_N]$$

$$= \hat{\delta}_N(q, w)$$

DONE

EXAMPLE

$$L = \{ w \mid w \text{ ENDS IN } 001, 010 \text{ OR } 100 \}$$



<u>EXERCISE</u> CONVERT TO DFA

<u>EXERCISE</u> TRY TO WRITE A DFA DIRECTLY.

<u>OBSERVE</u> DFA IS QUITE COMPLEX & SECOND EXERCISE IS PRETTY TOUGH

<u>BUT</u> WRITING NFA IS QUITE EASY & CONVERTING TO DFA IS AN AUTOMATED PROCESS