COMP 330

Name: Yuhao Wu
ID Number: 260711365

2018-9-14

DATE: 9-14 LECTURE 6

NFA:

Given a Fixed alphabet Σ, $N = (Q, Q_0, \Delta, F)$

- Q: Finite set of states
- $\Delta : Q \times \Sigma \to P(Q)$ means the power set of Q, the collection of all subsets of Q
- $Q_0 \subseteq Q$: start state
- $F \subseteq Q$ is the set of accept states
- $\Delta^* : 2^Q \times \Sigma^* \to 2^Q$

We define that:

$$\Delta^*(A, \varepsilon) = A \qquad \Delta^*(A, wa) = \bigcup_{q \in \Delta^*(A,w)} \Delta(q, a)$$

Then we can have the fact:

- $\Delta^*(A \cup B, \varepsilon) = \Delta^*(A, \varepsilon) \cup \Delta^*(B, \varepsilon)$
- $\Delta^*(A, xy) = \Delta^*(\Delta^*(A, x), y)$

$$Define \quad L(N) = \{w | \Delta^*(Q_0, w) \cap F \neq \emptyset\} \qquad w \text{ means a word}$$

THEOREM: Given an NFA, $N = (Q, Q_0, \Delta, F)$ as above, $\exists$ DFA $M = (S, S_0, \delta, \widehat{F})$, such that $L(M) = L(N)$

Proof. We make

- $S = P(Q)$ Every state of M is a set of states of N ($P(Q)$ is the set of subsets of Q)
- $S_0 = \{Q_0\}$ M starts in the state corresponding to the collection containing just the start state of N
- $\widehat{F} = \{A \in Q | A$ contains an accept state of $N\}$
- $\delta(A, a) = \bigcup_{q \in A} \Delta(q, a) = \Delta^*(A, a)$
  For $A \in S$ and $a \in \Sigma$, let $\delta(A, a) = \{q \in A | q \in \Delta(r, a)$ for some $r \in A\}$
  If A is a state of M, it is also a set of state of N. When M reads a symbol a in state A, it shows where a takes each states in A. Because each state may go to a set of states, we take a union of all these sets.

Now, we must prove $L(M) = L(N)$
Lemma:

$$\Delta^*(A, w) = \delta^*(A, w) \qquad \forall w \in \Sigma^*$$

prove by induction:

- Base Case: $w = \varepsilon$, $\Delta^*(A, \varepsilon) = A = \delta^*(A, \varepsilon)$
- Induction Case: Let $w = xa$ and assume $\forall A \subseteq Q$

$$\Delta^*(A, x) = \delta^*(A, x)$$

$$\begin{aligned}
\delta^*(A, xa) &= \delta(\delta^*(A, x), a) \quad \text{definition of } \delta^* \\
&= \delta(\Delta^*(A, x), a) \quad \text{induction hyp} \\
&= \Delta^*(\Delta^*(A, x), a) \quad \text{defined in question} \\
&= \Delta^*(A, xa) \quad \text{Fact 2}
\end{aligned}$$

Lemma proved!

$$\begin{aligned}
L(N) &= \{w | \Delta^*(Q_0, w) \cap F \neq \emptyset\} \\
&= \{w | \Delta^*(Q_0, w) \in \widehat{F}\} \quad \text{def of } \widehat{F} \\
&= \{w | \delta^*(Q_0, w) \in \widehat{F}\} \quad \text{by Lemma} \\
&= \{w | \delta^*(s_0, w) \in \widehat{F}\} \quad \text{by definition of} S_0 \\
&= L(M)
\end{aligned}$$

□

DATE: 9-17 LECTURE 6

Regular Expression:

Algebra with terms defined inductively:

- $\emptyset$ is a regular expression
- ε is a regular expression
- Any letter from Σ is a regular expression
- If R, S is a regular expression, then $R + S$ is a regular expression
- If R, S is a regular expression, then $R \cdot S$ is a regular expression
- If R is a regular expression, then R^* is a regular expression

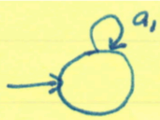
REMARK: A regular expression describes a subset of Σ^*

- $\emptyset$ defines the empty set
- ε defines the set $\{\varepsilon\}$
- $'a'$ defines $\{a\}$
- $R + S := \{w \in R\} \cup \{w \in S\}$
- $R \cdot S := \{w_1 w_2 | w_1 \in R, w_2 \in S\}$
- $R^* := \{w_1 w_2 \cdot w_n | \text{each } w_i \in R\} \cup \{\varepsilon\}$

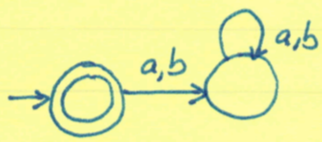
THEOREM:

- Language defined by any regular expression is regular language

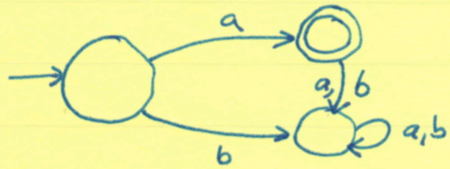
- Any regular language can be expressed by some regular expression

(i) 

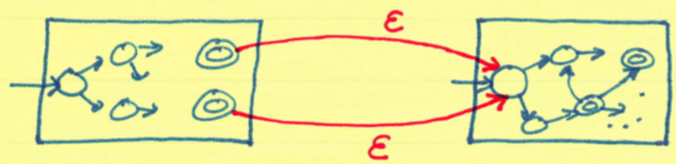
recognizes ϕ

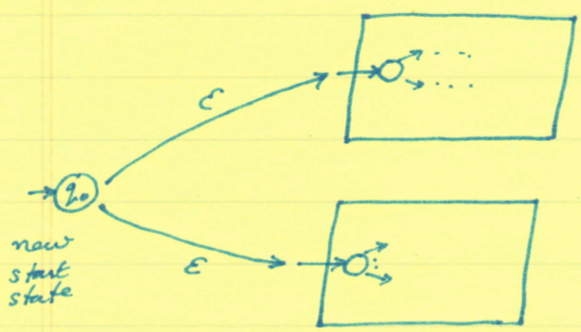
(ii) 

recognizes ε

(iii) 

recognizes a

(iv) 



$M_1 = (S_1, s_1, \delta_1, F_1)$

$M_2 = (S_2, s_2, \delta_2, F_2)$

New machine (NFA+ε)

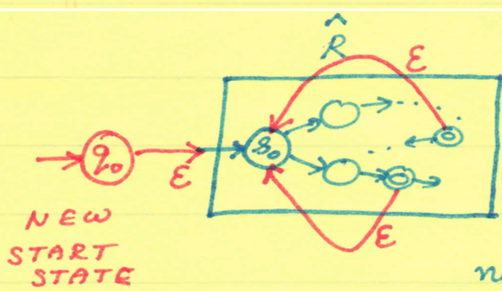
States $= S_1 \cup S_2 \cup \{q_0\}$

Start state $= q_0$

$$\Delta(q,a) = \begin{cases} \{\delta_1(q,a)\} & \text{if } q \in S_1 \\ \{\delta_2(q,a)\} & \text{if } q \in S_2 \\ \{s_1, s_2\} & \text{if } q = q_0 \ \& \ a = \varepsilon. \end{cases}$$

$a \in \Sigma \cup \{\varepsilon\}$

Final states $= F_1 \cup F_2$

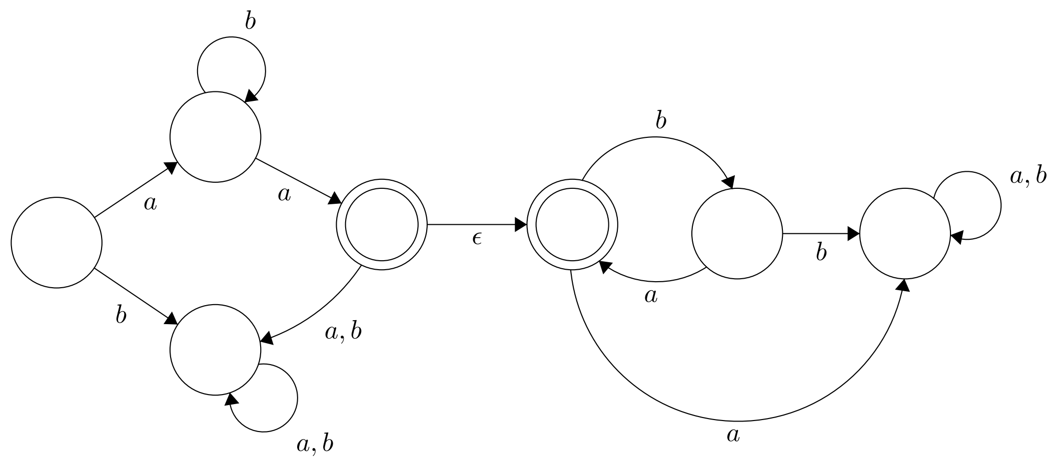


Add a new start state and make it an accept state (ε is always in $\hat{R}^*$). Put new ε-moves from the old accept states to the old start state.

NOTE: q_0 is the new start state as well as the accept state. It can accept word ε, and the next ε is the ε move.

Example:

$$L_1 = ab^*a \qquad L_2 = (ba)^* \qquad L_1 \cdot L_2$$



From DFA to Regular expression:

- Let $Q = \{q_1, q_2, \ldots, q_n\}$ be the set of all automatons states
- Suppose $R_{ij}^{(k)}$ represents the set of all strings that transition the automaton from q_i to q_j without passing through any state higher than q_k
- We can construct the language R_{ij} by successively constructing $R_{ij}^1, R_{ij}^2, \ldots, R_{ij}^n = R_{ij}$
- R_{ij}^k is recursively defined as:

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1}$$

- Assuming we have initialized R_{ij}^0 to be

$$R_{ij}^0 = \begin{cases} r & \text{if } i \neq j \text{ and r transitions NFA from } q_i \text{ to } q_j \\ r + \varepsilon & \text{if } i = j \text{ and r transitions NFA from } q_i \text{ to } q_j \\ \emptyset & o.w. \end{cases}$$

- We always start in the order of $R_{ij}^0, R_{ij}^1, R_{ij}^2, \ldots, R_{ij}^n$

Equation between regular expression:

- $R \cdot (S + T) = R \cdot S + R \cdot T \qquad R + (S + T) = (R + S) + T \qquad R + R = R$
- $R + \emptyset = R \qquad R + S = S + R \qquad R \cdot \varepsilon = R$
- $\varepsilon \cdot R = R \cdot \varepsilon \qquad R \cdot (S \cdot T) = (R \cdot S) \cdot T$
- $\varepsilon + R^* R = \varepsilon + RR^* = R^*$

DATE: 9-19 LECTURE 7

Minimization of DFA

To begin with, we define an equivalence relation on state
Use equivalence classes to define a new smaller machine.

Definition: Given a DFA, $M = \{S, S_0, \delta, F\}$ over Σ, we say $P \approx S$, if

$$\forall x \in \Sigma^*, \delta^*(p, x) \in F \Leftrightarrow \delta^*(q, x) \in F$$

REMARK: NOT equivalent?

$$\exists x \in \Sigma^* \begin{cases} \delta^*(p, x) \in F \text{ while } \delta^*(q, x) \notin F \\ \delta^*(p, x) \notin F \text{ while } \delta^*(q, x) \in F \end{cases}$$

Lemma A: $p \approx q \implies \forall a \in \Sigma, \delta(p, a) \approx \delta(q, a)$

Proof. Suppose that $\delta^*(\delta(p, a), x) \in F$, then we can say that $\delta^*(p, ax) \in F$
As we know that $p \approx q$, according to the definition we can say $\delta^*(p, ax) \in F \Leftrightarrow \delta^*(q, ax) \in F$
So, we can say that $\delta^*(\delta(q, a), x) \in F$
Thus, we can say $\delta(p, a) \approx \delta(q, a)$

□

Lemma A*: $\forall x \in \Sigma^*, \delta^*([p], x) = [\delta^*(p, x)]$
We can prove this via induction on the length of the string

REMARK: $p \approx q$ can be written as $[p] = [q]$, so the lamma can be written as $[p] = [q] \implies [\delta(p, a)] = [\delta(q, a)]$

quotient machine

Next, we define the concept of a **quotient machine**: $M/\approx = (S', s'_0, \delta', F')$ (with the same alphabet Σ), where:

- $S' = \{[s] \quad | s \in S\}$ $\qquad (s/\approx)$ (the set of all equivalence classes)
- $s'_0 = [s_0]$
- $F' = \{[p] | p \in F\}$ (the set of equivalence classes for accept states)
- $\delta' : Q' \times \Sigma \to Q'$ is defined by $\delta'([p], a) = [\delta(p, a)]$

Lemma B: $p \in F, p \approx q \implies q \in F$
Using definition and ε to prove this.

Lemma C: $\forall w \in \Sigma^*, \delta'^*([p], w) = [\delta^*(p, w)]$

Proof. Prove by induction:

- Base Case: $w = \varepsilon$, then we have $\delta'^*([p], \varepsilon) = [p] = [\delta^*(p, \varepsilon)]$

- Inductive Step: Suppose that $\delta'^*([p],w) = [\delta^*(p,w)]$, then we need to prove that: $\delta'^*([p],wa) = [\delta^*(p,wa)]$

$$\begin{aligned}
\delta'^*([p],wa) &= \delta'\Bigg(\delta'^*([p],w),a\Bigg) \\
&= \delta'\Bigg([\delta^*(p,w)],a\Bigg) \textbf{ according to hypothesis} \\
&= \Bigg[\delta(\delta^*(p,w),a)\Bigg] \textbf{ according to the definition of } \delta' \\
&= [\delta(p,wa)]
\end{aligned}$$

$\square$

Theorem: $L(M') = L(M)$

Proof.

$$\begin{aligned}
x \in L(M') &\Leftrightarrow \delta'^*([s_0],x) \in F' \\
&\Leftrightarrow [\delta^*(s_0,x)] \in F' \\
&\Leftrightarrow \delta^*(s_0,x) \in F \\
&\Leftrightarrow x \in L(M)
\end{aligned}$$

$\square$

Distinguishable:

$p \quad \bowtie \quad q := \exists w \in \Sigma$ such that $\begin{cases} \delta*(p,w) \in F \quad \delta*(q,w) \notin F \\ (or) \quad \delta*(p,w) \notin F \quad \delta*(q,w) \in F \end{cases}$

Fact: if $\exists a \in \Sigma$, such that $\delta(p,a) \quad \bowtie \quad \delta(q,a)$, then $p \quad \bowtie \quad q$

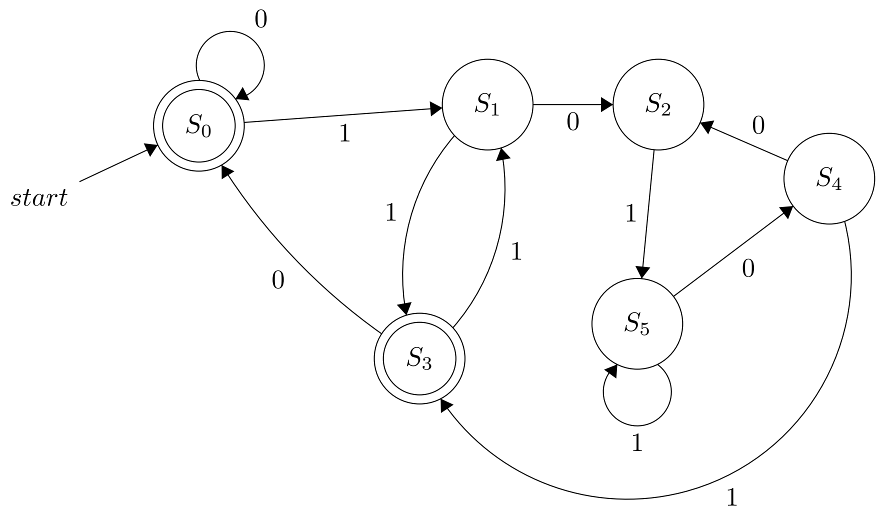
Algorithm:

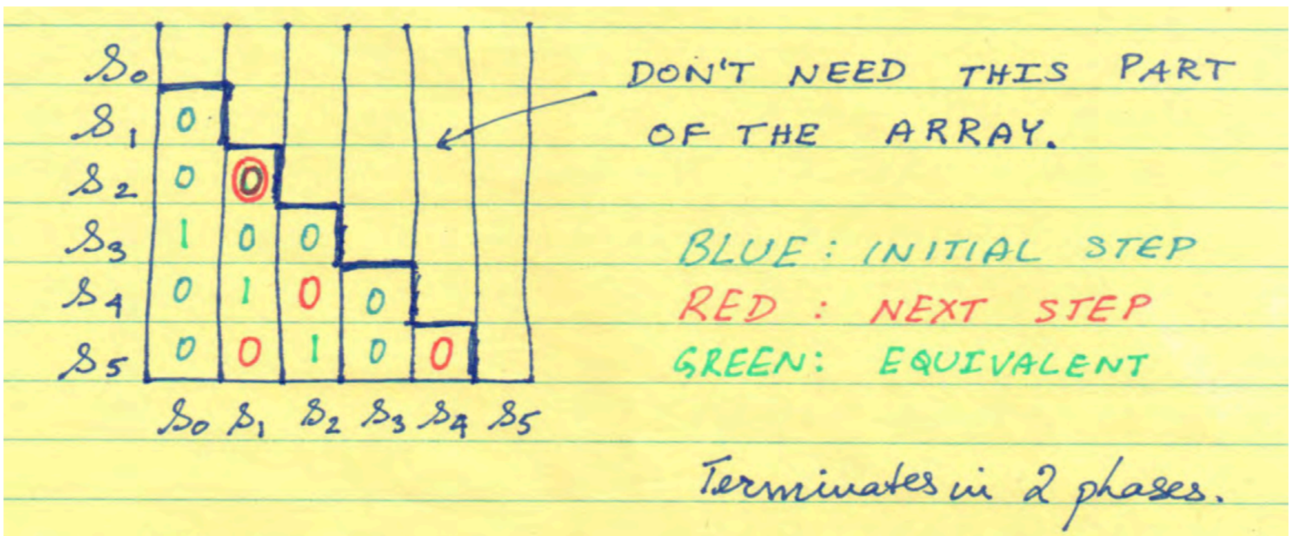
Define a $S \times S$ array of booleans:

1 : for every pair (p, q) such that $p \in F$ and $q \notin F$, we put a 0 at position (p,q) of the matrix

2 : Repeat until no more change:
For each pair of (p,q) not marked 0, check if $\exists a \in \Sigma$, such that $(\delta(p,a), \delta(q,a))$ is 0. If yes, then mark (p,q) as 0

3 : mark everything else as 1





	S_0	S_1	S_2	S_3	S_4	S_5
S_0						
S_1	0					
S_2	0	0				
S_3	1	0	0			
S_4	0	1	0	0		
S_5	0	0	1	0	0	

DON'T NEED THIS PART OF THE ARRAY.

BLUE : INITIAL STEP

RED : NEXT STEP

GREEN: EQUIVALENT

Terminates in 2 phases.

The first step we will make (0,1), (0,2), (0,4), (0,5), (1,3), (2,3), (3,4), (3,5) be 0

The second step we will make (1,2), (1,5), (2,4), (4,5) be 0