THEOREM : EQUIVALENCE OF NFA - DFA

THEOREM

If L is accepted by an NFA then there exists a DFA that accepts L.

PROOF:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA where

Q is Finite non empty set of states.

Σ is Finite non empty set of input symbols.

δ is the transition function which is mapping

From $\delta : Q \times \Sigma \to 2^Q$

q_0 is the initial state

$F \subseteq Q$ is set of Final states.

Now we construct a DFA as follows:

$M' = (Q', \Sigma, \delta', [q_0], F')$

where

i) $Q' = 2^Q$ (Power set of elements), $[q_1, q_2 \cdots q_j]$ is a single state in Q' For several states in Q.

ii) Σ is same.

iii) δ' is a transition fun/. which is mapping From

$\delta' : Q' \times \Sigma \to Q'$ defined by

$\delta'([q_1, q_2 \cdots q_j], a) = [p_1, p_2 \cdots p_k]$

iff

$\delta(\{q_1, q_2 \cdots q_j\}, a) = \{p_1, p_2 \cdots p_k\}$

iv) q_0 is initial state (The initial state of NFA becomes initial state of DFA).

$F' \subseteq Q'$ which contain atleast one element of F.

THEOREM : EQUIVALENCE OF NFA - DFA

we have to show that

$\delta'([q_0], x) = [P_1, P_2 \ldots P_j]$

iff $\delta(\{q_0\}, x) = \{P_1, P_2 \ldots P_j\}$ for a string of any length.

we have to Prove this by mathematical Induction.

| Induction
| $n=1$
| "n" Assume
| '$n+1$'

Induction:

Let x = 'a' of length 1, then

$\delta'([q_0], x) = \delta(\{q_0\}, x)$

$\delta'([q_0], a) = \delta(\{q_0\}, a)$ this is true for the length of string $|x| = 1$.

/* Let x = 'w' of length n */

we assume that it is true for 'x' of length n. i.e, $\delta'[[q_0], x) = \delta(\{q_0\}, x)$

Let the string be 'xa' which is of length $n+1$

we have to Prove that $\delta'([q_0], xa) = \delta(\{q_0\}, xa)$

$\delta'([q_0], xa) = \delta'(\delta'([q_0], x), a)$

$= \delta'([q_1, q_2 \ldots q_j], a)$ ($q_1, q_2 \ldots q_j$ are the states in $\delta(\{q_0\}, x)$)

iff $\delta(\{q_0\}, x) = \{q_1, q_2 \ldots q_j\}$

$= [P_1, P_2 \ldots P_k]$

iff $\delta(\{q_1, q_2 \ldots q_j\}, a) = \{P_1, P_2 \ldots P_k\}$

| $\delta'(\delta'([q_0], x), a)$
| $\delta'(\delta(\{q_0\}, x), a)$
| $\delta'(\delta(\{q_0\}, x), a)$
| $= \delta(\{q_0\}, xa)$

which establish our Hypothesis.

$\delta'([q_0], xa) \in F'$

iff $\delta(\{q_0\}, xa)$ should contain atleast one Final state of F

THEOREM : EQUIVALENCE OF NFA WITH EPSILON MOVES TO NFA WITHOUT EPISLON MOVES

THEOREM: If L is accepted by an NFA with ε-moves then there exists an NFA without ε-moves that accepts L.

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA with ε-moves that accepts $L(L(M))$ where

Q is Finite non-empty set of states.

Σ is finite non-empty set of i/p symbols

δ is transition Fun% which is mapping

From $\delta: Q \times \Sigma \cup \{\epsilon\} \longrightarrow 2^Q$

q_0 is initial state

$F \subseteq Q$ is a set of Final states.

Now we construct a NFA without ε-moves as follows:

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

where

1) Q is the finite set of states.

2) $\Sigma' = \Sigma - \{\epsilon\}$

3) $\delta'(q, a) = \epsilon\text{-cl}(\delta(\epsilon\text{-cl}(q), a))$

4) $q_0' = \epsilon\text{-cl}(q_0)$, set of all states which are reachable from q_0, all become the initial state of NFA without ε-moves.

5) $F' = \epsilon\text{-cl}(Q)$ which contains atleast one Final state of F.

Now we conclude this theorem by showing

$$\delta(q_0, x) = \delta'(q_0, x)$$

THEOREM : EQUIVALENCE OF NFA WITH EPSILON MOVES TO NFA WITHOUT EPISLON MOVES

Basic: since $\delta(q_0, \epsilon) \neq \delta'(q_0, \epsilon)$

we start induction by considering a string 'x' of length 1, i.e, $|x| = 1$

then $\delta(q_0, a) = \delta'(q_0, a)$ which is true for length 1.

Now we assume that it is true for word 'ω' of length n, i.e, $|\omega| = n$

$$\delta(q_0, \omega) = \delta'(q_0, \omega)$$

Now we show that it is true for $x = '\omega a'$ for length $n+1$, i.e, $|\omega a| = n+1$.

$$\delta(q_0, \omega a) = \delta(\delta(q_0, \omega), a) \quad [\because \delta'(\delta'(q_0, \omega), a) = \delta'(q_0, \omega a)]$$

$$= \delta(\delta'(q_0, \omega), a)$$

$$= \delta(P, a) \text{ where } \delta'(q_0, \omega) = P$$

$$= \bigcup_{q_i \text{ in } P} \delta(q_i, a) = \bigcup_{q_i \text{ in } P} \delta'(q_i, a)$$

$$= \delta'(\delta'(q_i, \omega), a)$$

$$= \delta'(q_i, \omega a)$$

Hence by induction $\delta(q_0, \omega a) = \delta'(q_i, \omega a)$

we claim that ϵ-cl(Q) contain a state of F then it is also treated as a Final state in NFA without ϵ-moves.

$$\delta'(q_i, x) \subseteq F \Rightarrow q_i \in F'$$

ϵ-cl(q_0) = the set of all initial states of NFA without ϵ-moves, Hence $L(M) = L(M')$

THEOREM : EQUIVALENCE OF REGULAR EXPERISSION - NFA

THEOREM:

Let γ be a regular expression then their exists a NFA with ϵ transitions that accepts L.

PROOF:

we prove the theorem by induction on no of operators.

Consider a regular exp^{n} γ that their is an NFA M with ϵ transitions having one final state & no of transitions out of this final state such that $L(M) = L(\gamma)$

BASIS:

a) 0 operators:

The exp^{n} γ must be ϵ, $\emptyset$, 'a' for some symbol in Σ

The NFA is shown below

$\gamma = \epsilon$ $\gamma = \emptyset$ $\gamma = a$

INDUCTION: (one or more operators)

Assume that Theorem is true for R.E with $< i$ operators $(i \geq 1)$

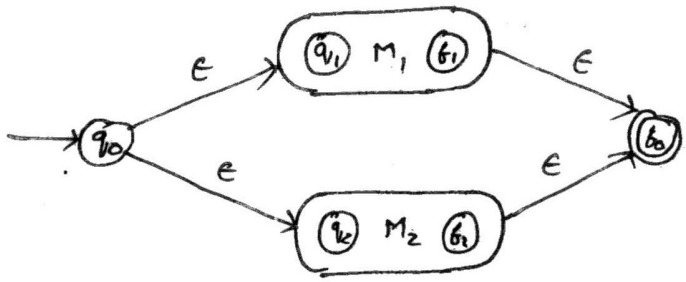
These are 3 cases depending on the form of γ.

<u>case i</u>: $\gamma = \gamma_1 + \gamma_2$

with γ_1 & γ_2 must have $< i$ operators then their exists an NFA $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ with $L(M_1) = L(\gamma_1)$ & $L(M_2) = L(\gamma_2)$.

THEOREM : EQUIVALENCE OF REGULAR EXPERISSION - NFA

The NFA is given below:



where δ is defined as:

1) $\delta(q_0, \epsilon) = \{q_1, q_2\}$

$\delta(q_1, a) = \delta_1(q_1, a)$ where $q_1 \in Q_1 - \{F_1\}$ & $a \in \Sigma$

$\delta(q_1, a) = \delta_2(q_1, a)$ where $q_1 \in Q_2 - \{F_2\}$ & $a \in \Sigma$

$\delta(F_1, \epsilon) = \delta(F_2, \epsilon) = f_0$

Any Path in M from q_0 to f_0 should begin either from q_1 or q_2 on ϵ.

If the Path goes to q_1, it may follow any Path in M_1 to F_1. These are the only paths from q_0 to f_0.

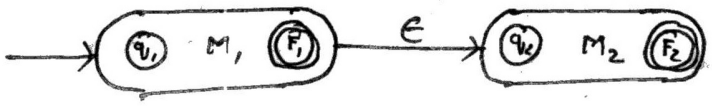
Hence $L(M) = L(M_1) + L(M_2)$

case ii:

Let $r = r_1 r_2$ where r_1 & r_2 must have $< i$ operators then their exists an NFA

$M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ & $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$

THEOREM : EQUIVALENCE OF REGULAR EXPERISSION - NFA



where δ is defined as:

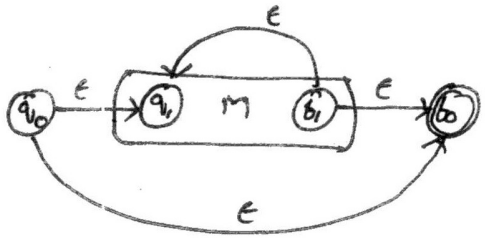
$$\delta(q_1, a) = \delta_1(q_1, a) \quad \text{where } q_1 \in Q_1 - \{F_1\}$$

$$\delta(F_1, \epsilon) = \{q_{12}\}$$

$$\delta(q_1, a) = \delta_2(q_1, a) \quad \text{where } q_1 \in Q_2 - \{F_2\} \text{ \& } a \in \Sigma_2$$

Every Path in M from q_1 & F_2 is a Path labelled by the string x from q_1 to F_1 & followed by an edge F_1 to q_{12} on e & then followed by some string y from q_{12} to F_2.

Case iii : $r = r_1^*$ with r_1 must have $< i$ operations then their exists an NFA $M_1 = (Q, \Sigma, \delta_1, q_1, F_1)$ such that $L(M_1) = L(r_1^*)$



$$M = (Q \cup \{q_{10}, f_0\}, \Sigma, \delta, q_{10}, f_0)$$

where δ is defined as

$$\delta(q_{10}, \epsilon) = \{q_1, f_0\}$$

$$\delta(F_1, \epsilon) = \{q_1, f_0\}$$

$$\delta(q_1, a) = \delta_1(q_1, a) \text{ where } q_1 \in Q - \{F_1\}$$

Any Path from q_{10} to F_0 consist either a Path from q_{10} to F_0 on e or any other transition on M_1 to reach F_1 followed by reading an e we can get back to q_1 or goto F_0, Hence $L(M_1) = L(r_1^*)$

CLOSURE PROPERTIES OF REGULAR SETS:

1) Union
2) Concatenation
3) Closure
4) Transpose
5) Intersection
6) Complementation

TRANSPOSE:

THEOREM: If L is regular, L^T is also regular.

PROOF:

If L is regular then we construct a FA $(Q, \Sigma, \delta, q_0, F)$ such that $T(M) = L$.

We construct a transition system 'L' by starting with the state diagram M. By reversing the direction of the directed edge.

The set of initial states of M' defined as set $\{F\}$ & q_0 is defined as the final state of M'.

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

If $\omega \in T(M)$, we have a Path from q_0 to some Final state F.

That implies $\delta(q_0, \omega) = P \in F$, accepted by M.

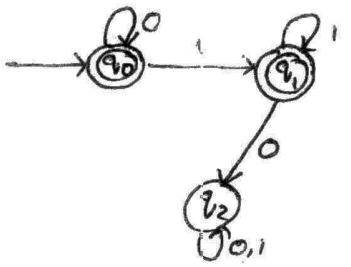
By reversing the edges we get a Path M' for some final state in F to q_0.

That implies $\delta'(P, a) = q$ iff $\delta(q, a) = P$.

The Path value is ω^T which $\in T(M')$

THUS PROVED.

1) Construct the FA given in the following Fig & Show that $[T(M)]^T$ is regular.



The lang for T(M) is

$L_1 = \{0^i / i \geq 0\}$

$L_2 = \{1^j / j \geq 0 \text{ or } 1\}$

$L_1 \cup L_2 = \{0^i 1^j / i \geq 0, j \geq 1\}$

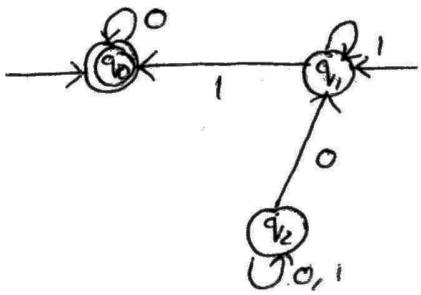
$T(M) = (Q, \Sigma, \delta, q_0, F)$

$= (\{q_0, q_1, q_2\}, \{0,1\}, \delta, \{q_0\}, \{q_0, q_1\})$

δ is shown in above Fig:

$T(M') = (\{q_0, q_1, q_2\}, \{0,1\}, \delta', \{q_0, q_1\}, \{q_0\})$

Here δ' is shown below



COMPLEMENT:-

If L is regular set over 'Σ' then $\Sigma^* - L$ (complement) is also regular over 'Σ'.

PROOF:-

If L is regular, then we can const: a DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepts L that is $L = T(M)$

Now we const: another DFA $M' = (Q, \Sigma, \delta, q_0, F')$

where $F' = Q - F$

M & M' reside only on their Final states.

A Final State of M' is a non Final state of M & vicevessa.

The state diag. M & M' are same except Final state

$$\omega \in T(M') \Rightarrow \delta(q_0, \omega) \in F' = Q - F$$

Thus $T(M) = \Sigma^* - L$

INTERSECTION:

Theorem:- If X and Y are two regular sets over Σ then $X \cap Y$ is also regular over Σ

PROOF:-

By Demorgan's law Theorem

$$X \cap Y = \Sigma^* - ((\Sigma^* - X) \cup (\Sigma^* - Y))$$

From the above theorem $\Sigma^* - X$, $\Sigma^* - Y$ are regular, we know that union of two regular exp's is regular

Hence $(\Sigma^* - X) \cup (\Sigma^* - Y)$ is also regular.

From the above Theorm, the complement of a regular exp. is regular.

Hence $\Sigma^* - ((\Sigma^* - X) \cup (\Sigma^* - Y))$ is regular.

$\therefore X \cap Y$ is regular.

Notes on Rice's Theorem

Consider any kind of software testing problem. Its description will typically start as "For a given program decide whether the function it computes is ...". In the setting of Turing machines, we often encounter natural problems of the form "Decide if the language recognized by a given Turing machine $\langle M \rangle$ ... ." Rice's theorem proves in one clean sweep that *all* these problems are undecidable. That is, whenever we have a decision problem in which we are given a Turing machine and we are asked to determine a property of the language recognized by the machine, that decision problem is always undecidable. The only exceptions will be the trivial properties that are always true or always false.

We use the following notation. If M is a Turing machine with input alphabet Σ, then $L(M) \subseteq \Sigma^*$ is the language recognized by M, that is, the set of strings that are accepted by M.

Theorem 1 *Let $\mathcal{C}$ be a set of languages. Consider the language $L_{\mathcal{C}}$ defined as follows*

$$L_{\mathcal{C}} = \{\langle M \rangle \mid L(M) \in \mathcal{C}\} .$$

Then either $L_{\mathcal{C}}$ is empty, or it contains the descriptions of all Turing machines, or it is undecidable.

To make sense of the statement of the theorem, think of a property of languages that you would like to test. For example the property of being regular. Then define $\mathcal{C}$ to be the set of all languages with that property. (In the example, $\mathcal{C}$ would be the set of regular languages.) Now, $L_{\mathcal{C}}$ is the language of (representations of) Turing machines that recognize languages having the property. (In the example, $L_{\mathcal{C}}$ would be the language of Turing machines that recognize regular languages.) The theorem says that unless every Turing machine recognizes a language with the property (not true for regular languages) and unless no Turing machine recognizes a language with the property (not true for regular languages), then $L_{\mathcal{C}}$ is undecidable. (So it is undecidable to tell whether a given Turing machine recognizes a regular language or not.)

Think of all the corollaries that you can infer from Rice's Theorem. It is undecidable to determine whether a given Turing machine accepts a finite or infinite number of inputs. It is undecidable to determine whether a given Turing machine accepts only (representations of) prime numbers, and so on.

PROOF: Suppose towards a contradiction that for same class $\mathcal{C}$ the language $L_{\mathcal{C}}$ is not empty, it does not contain the descriptions of all Turing machines, and it is decidable. Then $L_{\bar{\mathcal{C}}}$ is also not empty, not containing all Turing machines, and decidable.

Suppose that $\emptyset \notin \mathcal{C}$, otherwise apply the argument below to $L_{\bar{\mathcal{C}}}$ instead of $L_{\mathcal{C}}$.

Let M_{in} be a machine such that $\langle M_{in} \rangle$ is in $L_{\mathcal{C}}$.

We will show that the Acceptance problem is decidable, and so we will reach a contradiction.

Given an input $(\langle M \rangle, w)$ for the Acceptance problem, we constract a new Turing machine M_w that does the following: on input x, M_w first simulates the behaviour of M on input w and

- If M on input w loops, then so does M_w;
- If M on input w rejects, then so does M_w;

- If M on input w accepts, then M_w continues with a simulation of M_{in} on input x.

In summary:

- If M accepts w, then M_w behaves like M_{in}, and M_w accepts an input x if and only if M_{in} does. In other words, $L(M_w) = L(M_{in}) \in \mathcal{C}$ and so $\langle M_w \rangle \in L_{\mathcal{C}}$;
- if M does not accept w, then M_w does not accept any input, and $L(M_w) = \emptyset \notin \mathcal{C}$, which implies $\langle M_w \rangle \notin L_{\mathcal{C}}$.

We have proved that $(\langle M \rangle, w) \in A$ if and only if $\langle M_w \rangle \in L_{\mathcal{C}}$, and so A would be decidable if $L_{\mathcal{C}}$ were decidable. We have reached a contradiction. $\square$