CSC 240 LECTURE 10
LANGUAGE THEORY

Let Σ denote a finite alphabet i.e. set of letters.
Recall that Σ^* denotes the set of all (finite length) strings over Σ.
If $\Sigma = \{a,b\}$, then $\Sigma^* = \{ \lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$,
where λ is the empty string of length 0. It is sometimes denoted by ε.

A language over Σ is a subset of Σ^*
i.e. it is a set of strings over Σ.

Concatenation

if x and y are strings then $x \cdot y$ (or xy)
is the string consisting of all the letters in x followed by all the letters in y.

If x = aab and y = ba then $x \cdot y$ = aabba and $y \cdot x$ = baaab
$x \cdot \lambda = \lambda \cdot x = x$, so λ is an identity

If L and L' are languages then
$L \cdot L' = L\,L' = \{ x \cdot y \mid x \in L \text{ and } y \in L' \}$

example
If $L = \{a,bb\}$ and $L' = \{\lambda, c\}$
then $L \cdot L' = \{ac,bbc,a,bb\} \neq \{ca,cbb,a,bb\} = L' \cdot L$

$L^0 = \{\lambda\} \neq \lambda$
$L^1 = L$
$L^{i+1} = L^i \cdot L = L \cdot L^i$
$L^* = \cup \{L^i \mid i \geq 0\}$
$L^+ = \cup \{L^i \mid i \geq 1\}$
so $L^* = L^+ \cup \{\lambda\}$
$L^* = L^+$ if and only if $\lambda \in L$

x is a prefix of y if there exists a string x' such that $x \cdot x' = y$.
x is a suffix of y if there exists a string x' such that $x' \cdot x = y$.
They are proper if $x' \neq \lambda$.
x is a substring of y if there exist strings x' and x" such that $x' \cdot x \cdot x'' = y$.

It is proper if $x' \neq \lambda$ or $x'' \neq \lambda$.

Other operations on languages
Let L, L' be a language over Σ.
union: $L \cup L' = \{ x \mid (x \in L) \text{ OR } (x \in L')\}$
intersection: $L \cap L' = \{ x \mid (x \in L) \text{ AND } (x \in L')\}$
difference: $L - L' = \{ x \mid (x \in L) \text{ AND } (x \notin L')\}$
complementation: $\overline{L} = \Sigma^* - L = \{ x \in \Sigma^* \mid x \notin L\}$

Regular Expressions

a concise way of describing some languages

Let Σ be a finite alphabet.
Let R be the following inductively defined set of strings:
$\phi, \lambda \in R$
$\Sigma \subseteq R$
If $r,r' \in R$, then $(r+r') \in R$, $(r \cdot r') \in R$, and $r^* \in R$

R is called the set of regular expressions over Σ.

A generalized regular expression allows complementation, intersection, difference, and $^+$.

The language denoted by a regular expression r is $\mathscr{L}(r)$,
where $\mathscr{L}: R \to \{ L \mid L \subseteq \Sigma^*\}$ is defined inductively, as follows:

$\mathscr{L}(\phi) = \phi$

$\mathscr{L}(\lambda) = \{ \lambda \}$

$\mathscr{L}(a) = \{ a \}$ for each $a \in \Sigma$

$\mathscr{L}((r+r')) = \mathscr{L}(r) \cup \mathscr{L}(r')$ for $r,r' \in R$

$\mathscr{L}((r \cdot r')) = \mathscr{L}(r) \cdot \mathscr{L}(r')$ for $r,r' \in R$

$\mathscr{L}(r^*) = (\mathscr{L}(r))^*$ for $r \in R$

Similarly for generalized regular expressions.

$\mathscr{L}((r \cap r')) = \mathscr{L}(r) \cap \mathscr{L}(r')$

$\mathscr{L}((r-r')) = \mathscr{L}(r) - \mathscr{L}(r')$

$\mathscr{L}(\bar{r}) = \Sigma^* - \mathscr{L}(r)$

$\mathscr{L}(r^+) = (\mathscr{L}(r))^+$

Note that $\bar{r}$ is a shorthand for $\Sigma^* - r$

so ϕ is a shorthand for $\overline{\Sigma^*}$

Note that brackets can be removed when there is no ambiguity

For example, $r_1 \cdot r_2 \cdot r_3$

A language L is regular if and only if $L = \mathscr{L}(r)$ for some $r \in R$.

Two regular expressions r and r' are equivalent, $r \equiv r'$,

if they denote the same language, i.e. $\mathscr{L}(r) = \mathscr{L}(r')$.

Examples:

strings over {a,b,c} that start with ab

ab(abc)* X   abc is not in $\mathscr{L}$(ab(abc)*) ={ab, ababc, ababcabc,...}

ab · {a,b,c}* X

a · b · (a+b+c)*

$\mathscr{L}(a+b+c) = \{a,b,c\} = \{a\} \cup \{b\} \cup \{c\}$

strings over {0,1} with even parity, i.e. with an even number of 1's

(0*10*1)*0*

first and last symbols are different over the alphabet {0,1}

0(0+1)*1 + 1(0+1)*0

first and last symbols are different over the alphabet {0,1,2}

(0+1+2)* - (0 (0+1+2)* 0 + 1 (0+1+2)* 1 + 2 (0+1+2)* 2) generalized regular expression, not a regular expression

0(0 + 1 + 2)*1 + 0(0 + 1 + 2)*2 + 1(0 + 1 + 2)*0 + 1(0 + 1 + 2)*2 + 2(0 + 1 + 2)*1 + 2(0 + 1 + 2)*0 √

0(0+1+2)*(1+2) + 1(0+1+2)*(0+2) + 2(0+1+2)*(0+1)

Let $L = \{0^i 1^n \mid i+n \text{ is odd}\}$

((00)* + 1(11)*) + (0(00)* + (11)*) = (00)* + 1(11)* + 0(00)* + (11)* x

(00)*(0+1)(11)* √

r = (00)*0(11)* + (00)*1(11)*

Prove $L = \mathscr{L}(r)$.

To do so, prove $\mathscr{L}(r) \subseteq L$ and $L \subseteq \mathscr{L}(r)$.

Let $x \in L$.
Then $x = 0^i 1^n$ for some $i,n \in \mathbb{N}$ such that $i+n$ is odd.
There are 2 cases:
1. $i = 2a+1$ is odd and $n = 2b$ is even
   Then $x = (00)^a 0 (11)^b \in \mathscr{L}((00)^*0(11)^*) \subseteq \mathscr{L}(r)$,
   because $(00)^a \in \mathscr{L}((00)^*)$, so $(00)^a 0 \in \mathscr{L}((00)^*0)$, and $(11)^b \in \mathscr{L}((11)^*)$
2. $i = 2a$ is even and $n = 2b + 1$ is odd
   Then $x = (00)^a(11)^b 1 \in \mathscr{L}((00)^*(11)^*1) \subseteq \mathscr{L}(r)$.

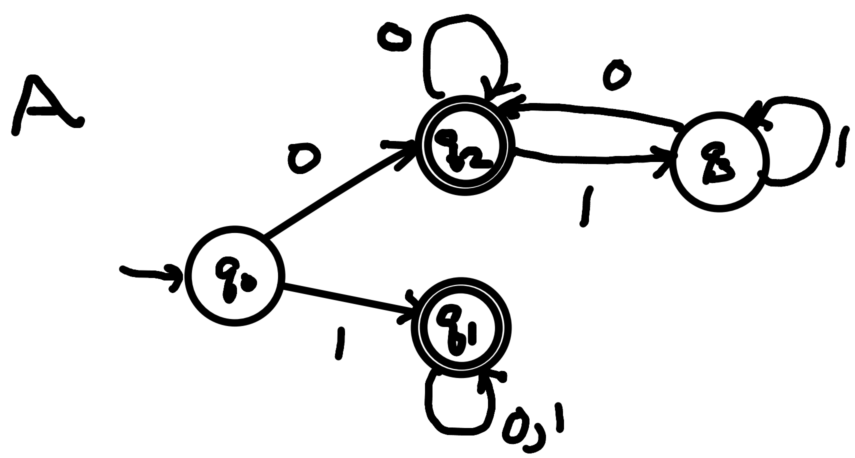
Thus $L \subseteq \mathscr{L}(r)$.

Let $x \in \mathscr{L}(r) = \mathscr{L}((00)^*0(11)^*) \cup \mathscr{L}((00)^*(11)^*1)$
Then either $x \in \mathscr{L}((00)^*0(11)^*)$ or $x \in \mathscr{L}((00)^*(11)^*1)$
either $x = (00)^a 0 (11)^b$ or $x = (00)^a(11)^b 1$ for some $a,b \in \mathbb{N}$.
In the first case $x = 0^i 1^n$, where $i = 2a+1$ and $n = 2b$ so $i+n$ is odd.
In the second case $x = 0^i 1^n$, where $i = 2a$ and $n = 2b+1$, so $i+n$ is odd.
In both cases, $x \in L$.
Thus $\mathscr{L}(r) \subseteq L$.

FINITE AUTOMATA

A (deterministic) finite (state) automaton (DFA or DFSA) is another way of describing a language. It uses a very simple model of a machine.

Example 1:



It has a finite set of states $Q = \{q_0, q_1, q_2, q_3\}$.
q_0 is the initial state denoted by an arrow pointing into it

q_1, q_2 are final states, denoted by a double circle
a set of final states, $F = \{q_1, q_2\}$
$\sum = \{0,1\}$ finite input alphabet (set of letters), labels that can occur on edges
Each directed edge represents a transition from a state to a state.
The label on the edge says what letter causes the transition.

The transitions can be described by a transition function $\delta: Q \times \sum \rightarrow Q$
$\delta(q_0,0) = q_2$
$\delta(q_0,1) = q_1$
$\delta(q_1,0) = q_1$
$\delta(q_1,1) = q_1$
$\delta(q_2,0) = q_2$
$\delta(q_2,1) = q_3$
$\delta(q_3,0) = q_2$
$\delta(q_3,1) = q_3$

Formally, a finite automaton is a 5-tuple $M = (Q,\sum,\delta,q_0,F)$, , where

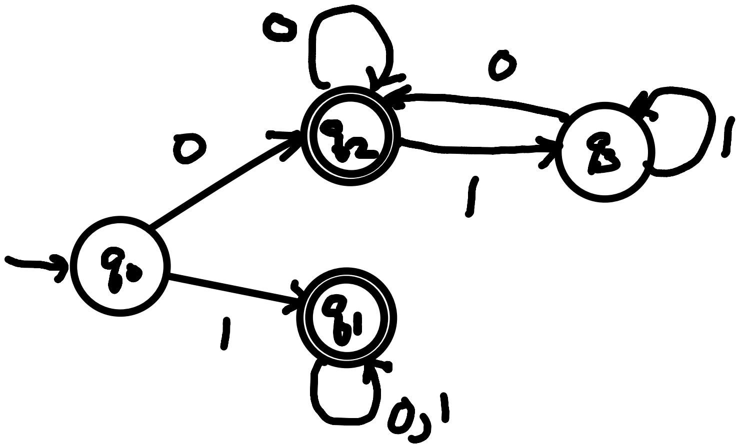
Q is a finite set of states,
$F \subseteq Q$ is the set of final states
$q_0 \in Q$ is the initial state

$\sum$ is a finite alphabet
$\delta: Q \times \sum \rightarrow Q$ is the transition function

Given an input string $a_1\ a_2\ \cdot\ \cdot\ \cdot\ a\_n \in \sum^*$,
the finite automaton operates as follows:
-it starts in the initial state
-it reads the input string from left to right, 1 letter at a time,
and changes state according to the transition function
(following the edge labelled by the letter)
-when all the letters have been read, a deterministic finite automaton
accepts the string if it is in a final state, i.e. a state in F
rejects the string if it is in a state in Q-F

examples
0110, 100 accepted
0101 is rejected

A



If M is a DFA, then the language accepted by M is defined to be L(M) = {x ∈ $\sum^*$ | x accepts M}

For the example,
L(A) = { x ∈ {0,1}* | x begins and ends with 0 or x begins with 1}

define the extended transition function $\delta^*: Q \times \sum^* \rightarrow Q$ by
$\delta^*(q,\lambda) = q$
and for all letters a ∈ ∑ and all strings x ∈ $\sum^*$

$\delta^*(q,xa) = \delta(\delta^*(q,x),a)$
or, equivalently,

$\delta^*(q,ax) = \delta^*(\delta(q,a),x)$

L(M) = { x ∈ $\sum^*$ | $\delta^*(q_0,x) \in F$}

To prove that L(A) = {x ∈ {0,1}* | x begins and ends with 0 or x begins with 1}, associate a set of strings L_i with each state q_i such that
$L_1 \cup L_2$ = { x ∈ {0,1}* | x begins and ends with 0 or x begins with 1}.
Then prove by structural induction or by induction on the length of x that
$\forall$ i ∈ {0,1,2,3}. (L_i = { x ∈ $\sum^*$ | $\delta^*(q_0,x) = q_i$}).

$L_0 = \{\lambda\}$
L_1 = { x ∈ {0,1}* | x begins with 1} = $\mathscr{L}(1(0+1)^*)$
L_2 = { x ∈ {0,1}* | x begins and ends with 0} = $\mathscr{L}(0(0+1)^*0 + 0)$
L_3 = { x ∈ {0,1}* | x begins with 0 and ends with 1} = $\mathscr{L}(0(0+1)^*1)$

Example 2

Give a deterministic finite automaton that accepts the language
$\mathscr{L}$ ($(0+1)^*1(0+1)$) = { x in $\{0,1\}^*$ | the second last letter of x is 1}.

7 states:

$\{\lambda\}$

$L_0 = \{0\}$

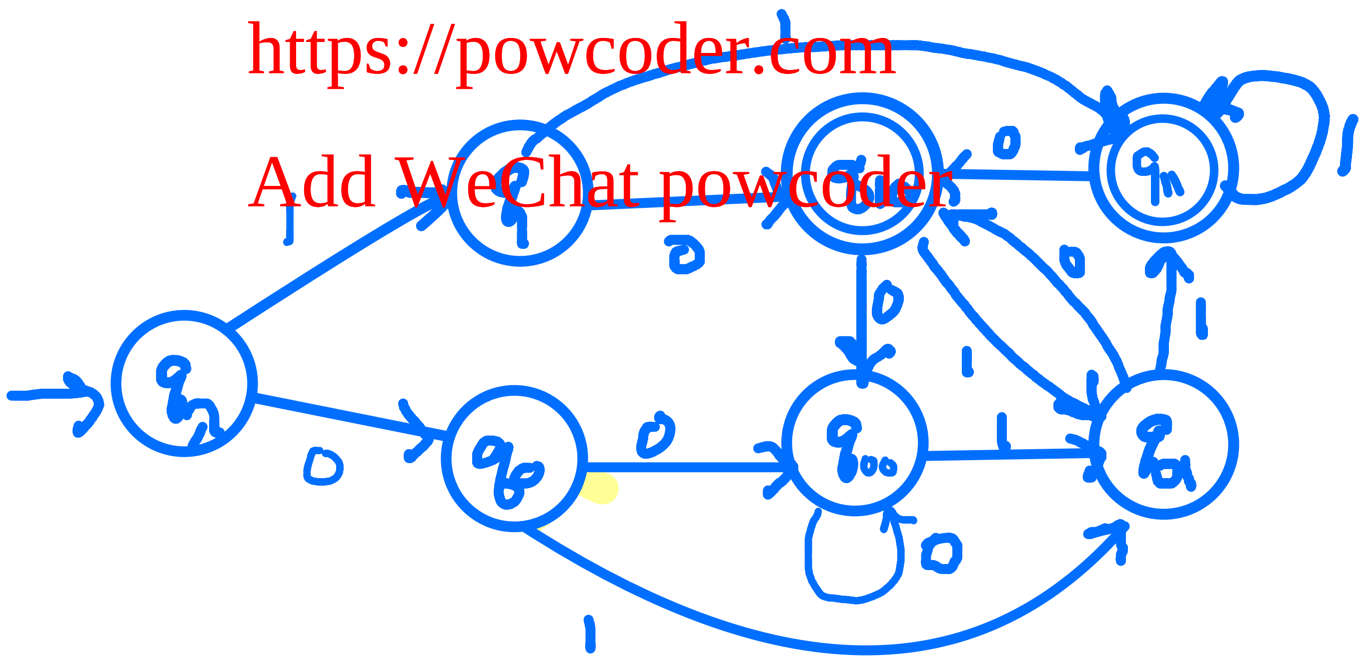
$L_1 = \{1\}$

$L_{00} = \{x \in \{0,1\}^* \mid$ x ends in 00$\} = \mathscr{L}((0+1)^*00)$

$L_{01} = \{x \in \{0,1\}^* \mid$ x ends in 01$\} = \mathscr{L}((0+1)^*01)$

$L_{10} = \{x \in \{0,1\}^* \mid$ x ends in 10$\} = \mathscr{L}((0+1)^*10)$

$L_{11} = \{x \in \{0,1\}^* \mid$ x ends in 11$\} = \mathscr{L}((0+1)^*11)$



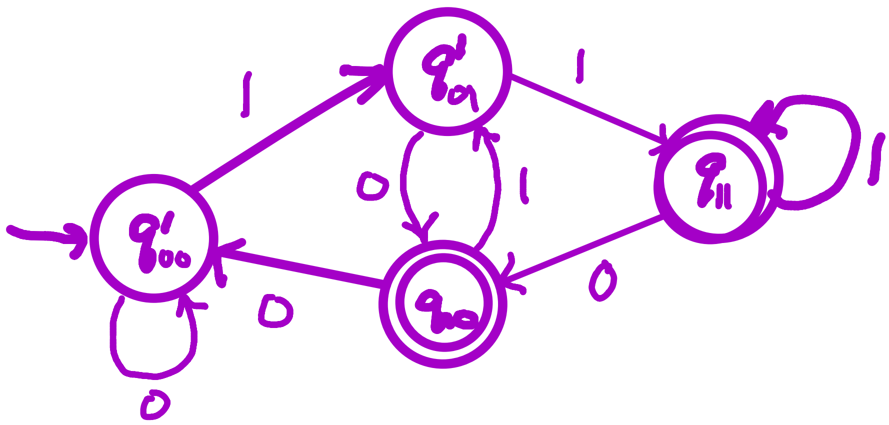
4 states:

$L'_{00} = \{x \in \{0,1\}^* \mid$ x ends in 00$\} \cup \{\lambda, 0\}$

$L'_{01} = \{x \in \{0,1\}^* \mid$ x ends in 01$\} \cup \{1\}$

$L_{10} = \{x \in \{0,1\}^* \mid$ x ends in 10$\}$

$L_{11} = \{x \in \{0,1\}^* \mid$ x ends in 11$\}$



The set of states Q of a finite automaton can be thought of as an object with different fields.
For example, Q = S x L, where L stores the last letter read and S stores the second last letter read.

Nondeterministic Finite Automata (NFA, NFSA)

Like a deterministic finite automaton, a nondeterministic finite (state) automaton is a 5-tuple $M=(Q,\Sigma,\delta,q_0, F)$, but
$\delta:Q \times \Sigma \rightarrow \mathscr{P}(Q)$
the range of δ is $\mathscr{P}(Q) = \{ Q' \mid Q' \subseteq Q\}$ instead of Q

-allows moves to different states or no states on a given letter
-models choice, for example a robot walking through a maze

define the extended transition function
$\delta^*:Q \times \Sigma^* \rightarrow \mathscr{P}(Q)$ by
$\delta^*(q,\lambda) = \{q\}$
and for all letters a and all strings x
$\delta^*(q,xa) = \cup \{\delta(q',a) \mid q' \in \delta^*(q,x)\}$
or, equivalently,
$\delta^*(q,ax) = \cup \{\delta^*(q',x) \mid q' \in \delta(q,a)\}$

A string x is accepted by a finite automaton if there is a path from the start state to an accept state labelled by x.

How is L(M) defined for a nondeterministic finite automaton M?
$L(M) = \{ x \in \Sigma^* \mid \delta^*(q_0,x) \cap F \neq \phi\}$

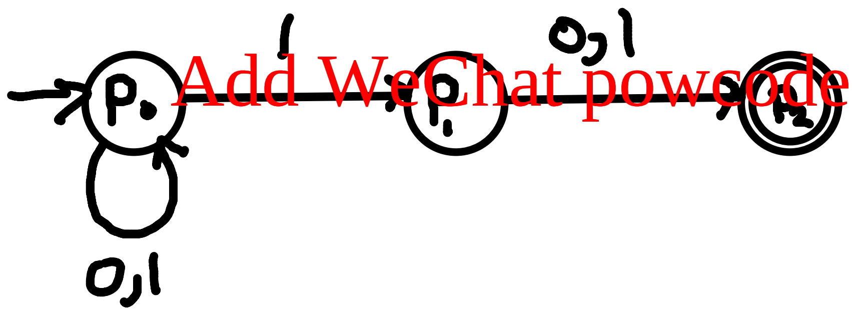
M accepts the string x if there is a sequence of lucky guesses it can make to bring it from the start state q_0 to a final state.

a nondeterministic finite automaton that accepts the language
$\mathscr{L}((0+1)^*1(0+1)) = \{ x \in \{0,1\}^* \mid \text{the second last letter of } x \text{ is } 1\}$

1. The deterministic finite automaton we talked about earlier can be viewed as a nondeterministic finite automaton.

Observation Every deterministic finite automaton can be viewed as a nondeterministic finite automaton by changing its transition function from $\delta(q,a) = q'$ to $\delta(q,a) = \{q'\}$
for all $q \in Q$ and all $a \in \Sigma$.

2. 3 states
p_0: initial state with self loops on 0,1, and edge to p_1 on 1
p_1: edge to p_2 on 0,1
p_2: final state, no outedges



0010
1010 are both accepted

Note: it can be much easier to construct a nondeterministic finite automaton than a deterministic finite automaton for some languages.

Are there some languages that can be accepted by nondeterministic finite automata, but not by deterministic finite automata?

Theorem For every NFA $M = (Q,\sum,\delta, q_0,F)$,
there is a DFA $M' = (Q',\sum,\delta', q'_0,F')$
that accepts the same language i.e. $L(M) = L(M')$.

Proof (subset construction):

Use generalization:
Let $M = (Q,\sum,\delta, q_0,F)$ be an arbitrary NFA.
The idea is to construct a DFA M' that keeps track of the states that M could be in as it reads the input string.

Let $M' = (Q',\sum,\gamma,q'_0,F')$ be defined as follows:

$Q' = \mathscr{P}(Q)$

$q'_0 = \{q_0\}$

$\gamma(S,a) = \cup\{\delta(q,a) \mid q \in S\}$ for all $S \in \mathscr{P}(Q)$ and $a \in \sum$

$F' = \{S \in \mathscr{P}(Q) \mid S \cap F \neq \phi\}$.

$L(M) = L(M')$.

For all $w \in \sum^*$, let $P(w) =$ "$\gamma^*(\{q_0\},w) = \delta^*(q_0,w)$"

In other words, $(q \in \gamma^*(\{q_0\},w))$ IFF $(q \in \delta^*(q_0,w))$.

Base case: $w = \lambda$.
By definition of extended transition function for a DFA, $\gamma^*(\{q_0\},\lambda) = \{q_0\}$.
By definition of extended transition function for an NFA, $\delta^*(q_0,\lambda) = \{q_0\}$.
Thus $P(\lambda)$ is true.

Constructor case: $w = xa$, where $x \in \sum^*$ and $a \in \sum$.
Assume $P(x)$ is true, so $\gamma^*(\{q_0\},x) = \delta^*(q_0,x)$.
By definition of extended transition function for a DFA,

$\gamma^*(\{q_0\},w) = \gamma(\gamma^*(\{q_0\},x),a)$

$= \cup\{\delta(q,a) \mid q \in \gamma^*(\{q_0\},x)\}$ by construction

$= \cup\{\delta(q,a) \mid q \in \delta^*(q_0,x)\}$ by substitution

$= \delta^*(q_0,w)$ by definition of extended transition function for an NFA.

Thus $P(w)$ is true.

By structural induction, $\forall\ w \in \Sigma^*.\ P(w)$.

$w \in L(M')$ if and only if $\gamma^*(\{q_0\},w) \in F'$ by definition

since $L(M') = \{\ x \in \Sigma^* \mid \gamma^*(\{q_0\},x) \in F'\}$

if and only if $\gamma^*(\{q_0\},w) \cap F \neq \phi$ by construction

since $F' = \{S \in \mathscr{P}(Q) \mid S \cap F \neq \phi\}$

if and only if $\delta^*(q_0,w) \cap F \neq \phi$ by substitution

since $\gamma^*(\{q_0\},w) = \delta^*(q_0,w)$

if and only if $w \in L(M)$ by definition

since $L(M) = \{\ x \in \Sigma^* \mid \delta^*(q_0,x) \cap F \neq \phi\}$.

Thus $L(M') = L(M)$.