Week 2 at a glance

Textbook reading: Sections 1.1, 1.2

Before Monday, read pages 41-43 (Figures 1.18, 1.19, 1.20) for examples of automata and languages.

Before Wednesday, read pages 48-50 (Figures 1.27, 1.29) which introduces nondeterminism.

Before Friday, read pages 45-46 (Theorem 1.25) that we'll refer to as a "closure proof".

For Week 3 Monday: Theorem 1.47 + 1.48, Theorem 1.39 "Proof Idea", Example 1.41, Example 1.56.

We will be learning and practicing to:

- Clearly and unambiguously communicate computational ideas using appropriate formalism. Translate across levels of abstraction.
  - Give examples of sets that are regular (and prove that they are).
    - **State the definition of the class of regular languages**
    - **Give examples of regular languages, using each of the three equivalent models of computation for proving regularity.**
  - Describe and use models of computation that don't involve state machines.
    - **Given a DFA or NFA, find a regular expression that describes its language.**
    - **Given a regular expression, find a DFA or NFA that recognizes its language.**
  - Use precise notation to formally define the state diagram of finite automata.
  - Use clear English to describe computations of finite automata TM informally.
    - **Design an automaton that recognizes a given language**
    - **Specify a general construction for DFA based on parameters**
    - **Design general constructions for DFA**
    - **Motivate the use of nondeterminism**
    - **State the formal definition of NFA**
    - **Trace the computation(s) of a NFA on a given string using its state diagram**
    - **Determine if a given string is in the language recognized by a NFA**
    - **Translate between a state diagram and a formal definition of a NFA**

TODO:

#FinAid Assignment on Canvas (complete as soon as possible) and read syllabus on Canvas

Schedule your Test 1 Attempt 1, Test 2 Attempt 1, Test 1 Attempt 2, and Test 2 Attempt 2 times at PrairieTest (http://us.prairietest.com)

Homework 1 submitted via Gradescope (https://www.gradescope.com/), due Tuesday 10/8/2024

Review Quiz 2 on PrairieLearn (http://us.prairielearn.com), complete by Sunday 10/13/2024 automata ← plural

Monday: Finite automaton constructions

← singular

Review: Formal definition of DFA: $M = (Q, \Sigma, \delta, q_0, F)$

- Finite set of states Q
- Alphabet Σ
- Transition function δ : $Q \times \Sigma \to Q$
- Start state q_0
- Accept (final) states F

loop counts as 1 arrow a

a,b counts as 2 arrows a b

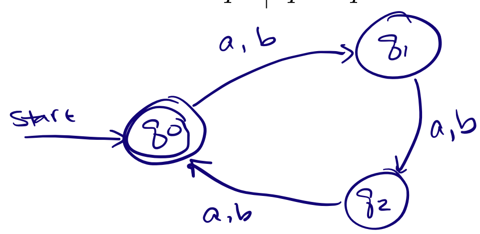
Quick check: In the state diagram of M, how many outgoing arrows are there from each state? $|\Sigma|$

Note: We'll see a new kind of finite automaton. It will be helpful to distinguish it from the machines we've been talking about so we'll use **Deterministic Finite Automaton** (DFA) to refer to the machines from Section 1.1.

$M = (\{q0, q1, q2\}, \{a, b\}, \delta, q0, \{q0\})$ where δ is (rows labelled by states and columns labelled by symbols):

δ	a	b
$q0$	$q1$	$q1$
$q1$	$q2$	$q2$
$q2$	$q0$	$q0$

The state diagram for M is



Give two examples of strings that are accepted by M and two examples of strings that are rejected by M:

over $\{a,b\}$

ε, abbaaa

a, b

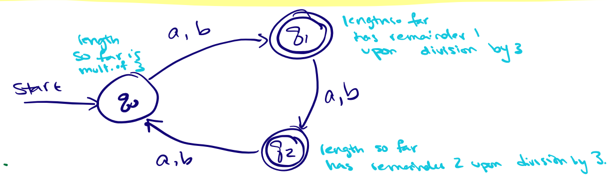
A regular expression describing $L(M)$ is

see Week 1

$L(M)$ language recognized by M
$= \{w \in \Sigma^* \mid w$ accepted by $M\}$.
$= \{w \in \Sigma^* \mid$ length of w is an int multiple of $3\}$

A state diagram for a finite automaton recognizing

$$\{w \mid w \text{ is a string over } \{a, b\} \text{ whose length is not a multiple of } 3\}$$ = complement of $L(M)$



COMPUTATION ON ε

q_0 accepting? No!

this machine rejects ε.

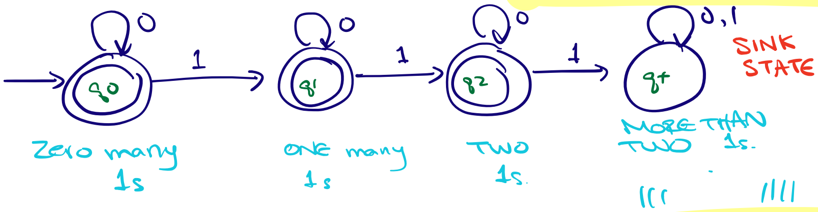
Extra example: Let n be an arbitrary positive integer. What is a formal definition for a finite automaton recognizing

$$\{w \mid w \text{ is a string over } \{0, 1\} \text{ whose length is not a multiple of } n\}?$$

Consider the alphabet $\Sigma_1 = \{0, 1\}$.

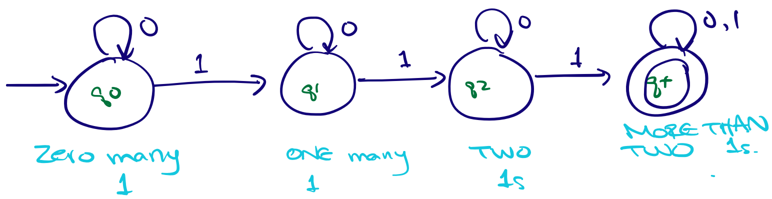
101 COMPUTATION q_0, q_1, q_1, q_2 110 COMPUTATION q_0, q_1, q_2, q_2 ε COMPUTATION q_0

A state diagram for a finite automaton that recognizes $\{w \mid w \text{ contains at most two 1's}\}$ is



111 1111 ~~ε~~

A state diagram for a finite automaton that recognizes $\{w \mid w \text{ contains more than two 1's}\}$ is



Strategy: Add "labels" for states in the state diagram, e.g. "have not seen any of desired pattern yet" or "sink state". Then, we can use the analysis of the roles of the states in the state diagram to work towards a description of the language recognized by the finite automaton.

Or: decompose the language to a simpler one that we already know how to recognize with a DFA or NFA.

Textbook Exercise 1.14: Suppose A is a language over an alphabet Σ. If there is a DFA M such that $L(M) = A$ then there is another DFA, let's call it M', such that $L(M') = \overline{A}$, the complement of A, defined as $\{w \in \Sigma^* \mid w \notin A\}$.

Proof idea:

Keep states, start state, transition function the same
Flip accept/reject status of the states

A useful bit of terminology: the **iterated transition function** of a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ is defined recursively by

$$\delta^*(\ (q, w)\ ) = \begin{cases} q & \text{if } q \in Q, w = \varepsilon \\ \delta(\ (q, a)\ ) & \text{if } q \in Q,\ w = a \in \Sigma \\ \delta(\ (\delta^*(q, u)), a)\ ) & \text{if } q \in Q,\ w = ua \text{ where } u \in \Sigma^* \text{ and } a \in \Sigma \end{cases}$$

Basis Step
Basis Step
Recursive Step
start state
last character.

Using this terminology, M accepts a string w over Σ if and only if $\delta^*(\ (q_0, w)\ ) \in F$. set of accept state.

Proof: Consider arbitrary language A over alphabet Σ.
Assume there is DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $L(M) = A$.
Want to show (WTS) there is DFA M' with $L(M') = \overline{A}$
Define witness DFA $M' = (Q, \Sigma, \delta, q_0, \{q \in Q \mid q \notin F\}$
WTS $L(M') = \{w \in \Sigma^* \mid w \notin L(M)\}$

Need to show two directions of implication

Goal ① $L(M') \subseteq \{w \in \Sigma^* \mid w \notin L(M)\}$

or equivalently, each string accepted by M' is rejected by M.

Let x be an arbitrary string over Σ and assume M' accepts x. By definition, this means $\delta^*((q_0, x)) \in \{q \in Q \mid q \notin F\}$

Since δ is the transition function for M as well as M', and since q_0 is the start state for M as well as M', the computation of M on x ends in this same state $\delta^*((q_0, x))$. Since F is the set of accepting states of M and $\delta^*((q_0, x)) \notin F$, M rejects x, as required.

Goal ② $L(M') \supseteq \{w \in \Sigma^* \mid w \notin L(M)\}$

or equivalently, each string rejected by M' is accepted by M.

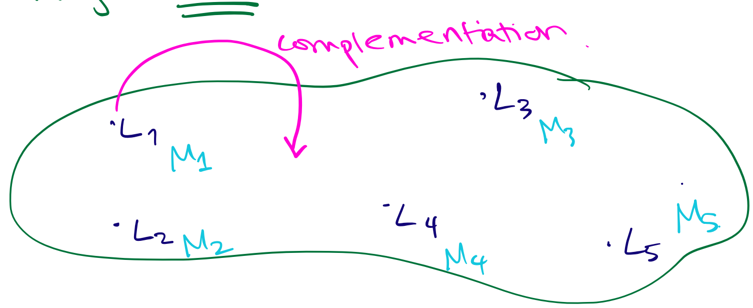
Let x be an arbitrary string over Σ and assume M' rejects x. By definition, this means $\delta^*((q_0, x)) \notin \{q \in Q \mid q \notin F\}$

Since δ is the transition function for M as well as M', and since q_0 is the start state for M as well as M', the computation of M on x ends in this same state $\delta^*((q_0, x))$. Since F is the set of accepting states of M and $\delta^*((q_0, x)) \in F$, M accepts x, as required.

Notice: $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$

M accepts all strings in this set
M rejects all strings <u>not</u> in this set.

Collection of languages each recognizable by some DFA.



Set operations we care about: $\cup$, $\circ$, $*$ etc.

Wednesday: $\boxed{\text{Nondeterministic automata}}$

We saw that whenever a language is recognized by a DFA, its complement is also recognized by some (other) DFA.

L

$\exists$ DFA M so that $L = L(M)$

Another way to say this is that the collection of languages that are each recognizable by a DFA is **closed** under complementation.

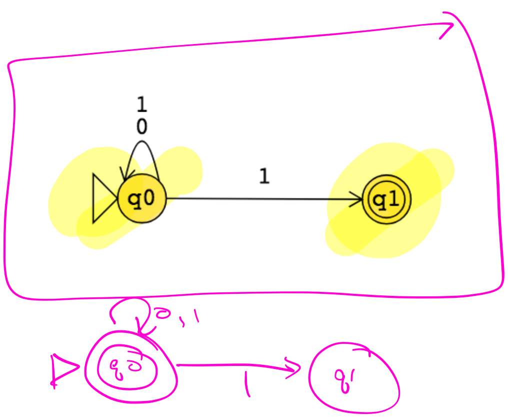
Deterministic: computation has exactly one choice for next step given current state and char to read

Nondeterministic finite automaton (Sipser Page 53) Given as $M = (Q, \Sigma, \delta, q_0, F)$

Finite set of states Q	Can be labelled by any collection of distinct names. Default: $q0, q1, \ldots$
Alphabet Σ	Each input to the automaton is a string over Σ.
Arrow labels Σ_ε	$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. ε labels spontaneous move: step in computation that doesn't read character. Arrows in the state diagram are labelled either by symbols from Σ or by ε
Transition function δ	$\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ gives the **set of possible next states** for a transition from the current state upon reading a symbol or spontaneously moving.
Start state q_0	Element of Q. Each computation of the machine starts at the start state.
Accept (final) states F	$F \subseteq Q$.

M accepts the input string $w \in \Sigma^*$ if and only if **there is** ($\exists$) a computation of M on w that processes the whole string and ends in an accept state. $L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$.

The formal definition of the NFA over $\{0, 1\}$ given by this state diagram is:



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

where $\delta : \{q_0, q_1\} \times \{0, 1, \varepsilon\} \to \mathcal{P}(\{q_0, q_1\})$

is given by the table

	0	1	ε
q_0	$\{q_0\}$	$\{q_0, q_1\}$	$\emptyset$
q_1	$\emptyset$	$\emptyset$	$\emptyset$

labels of arrows

states

The language over $\{0, 1\}$ recognized by this NFA is: $\{w \in \{0,1\}^* \mid w \text{ ends in } 1\} = L(\Sigma^* 1)$

0 computation: q_0, q_0 not in $\{q_1\}$. Reject.

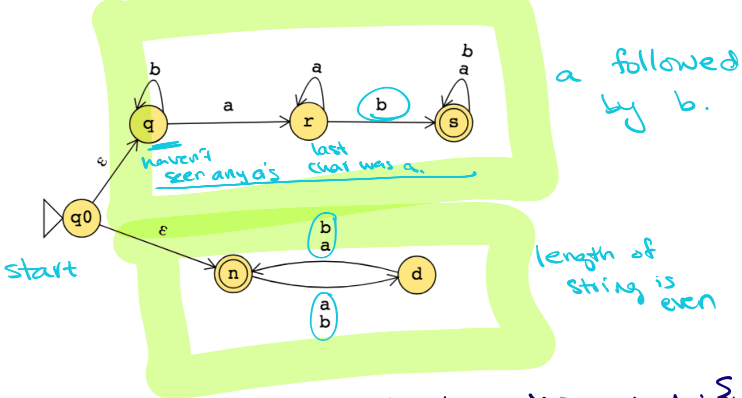
01 computations: q_0, q_0, q_0 not in $\{q_1\}$ AND $\boxed{q_0, q_0, q_1 \text{ in } \{q_1\}}$ Accept

11 computations: $q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_0$ AND $\boxed{q_0 \xrightarrow{1} q_0 \xrightarrow{1} q_1}$ AND $q_0 \xrightarrow{1} q_1$ stuck.

Change the transition function to get a different NFA which accepts the empty string (and potentially other strings too).

Bonus example: $L(\Sigma^* 1) = L(\Sigma^* 1 1^*)$

The state diagram of an NFA over $\{a, b\}$ is below. The formal definition of this NFA is: $(\{q_0, q, r, s, n, d\}, \{a,b\}, \delta, q_0, \{n, s\})$ with δ given by



bab
aba
ab

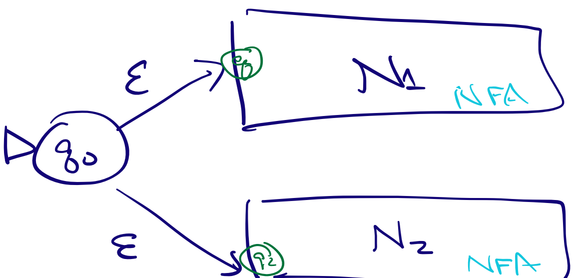
ε
ab

	a	b	ε
q_0	$\emptyset$	$\emptyset$	$\{q, n\}$
q	$\{r\}$	$\{q\}$	$\emptyset$
r	$\{r\}$	$\{s\}$	$\emptyset$
s	$\{s\}$	$\{s\}$	$\emptyset$
n	$\{d\}$	$\{d\}$	$\emptyset$
d	$\{n\}$	$\{n\}$	$\emptyset$

Language recognized by this NFA: $\{w \in \{a,b\}^* \mid w \text{ has a followed by b}\} \cup \{w \in \{a,b\}^* \mid w \text{ has even length}\}$

Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a NFA N_1 such that $L(N_1) = A_1$ and NFA N_2 such that $L(N_2) = A_2$, then there is another NFA, let's call it N, such that $L(N) = A_1 \cup A_2$.

Proof idea: Use nondeterminism to choose which of N_1, N_2 to run.



Formal construction: Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ and assume $Q_1 \cap Q_2 = \emptyset$ and that $q_0 \notin Q_1 \cup Q_2$. Construct $N = (Q, \Sigma, \delta, q_0, F_1 \cup F_2)$ where

- $Q =$ $Q_1 \cup Q_2 \cup \{q_0\}$
- $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is defined by, for $q \in Q$ and $x \in \Sigma_\varepsilon$:

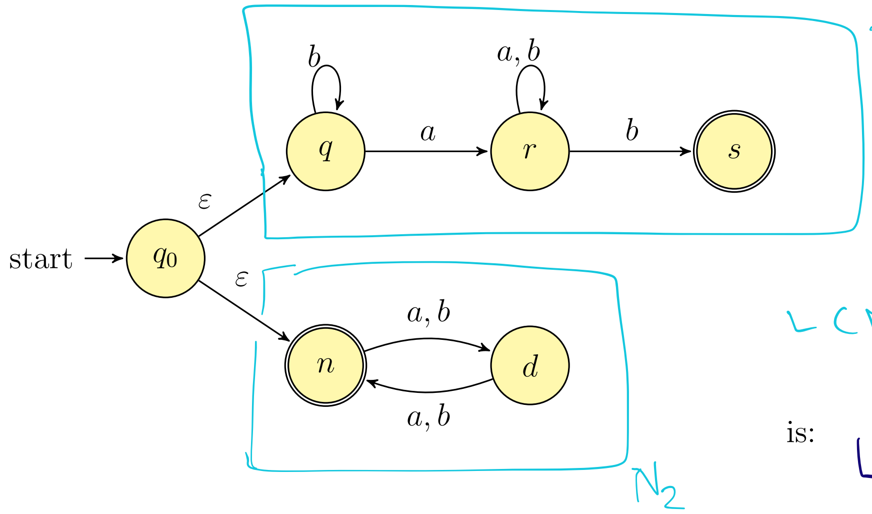
$\Sigma \cup \{\varepsilon\}$
input alphabet
special label

$$\delta((q,x)) = \begin{cases} \{q_1, q_2\} & \text{if } q = q_0 \quad x = \varepsilon \\ \emptyset & \text{if } q = q_0 \quad x \in \Sigma \\ \delta_1((q,x)) & \text{if } q \in Q_1 \quad x \in \Sigma_\varepsilon \\ \delta_2((q,x)) & \text{if } q \in Q_2 \quad x \in \Sigma_\varepsilon \end{cases}$$

Proof of correctness would prove that $L(N) = A_1 \cup A_2$ by considering an arbitrary string accepted by N, tracing an accepting computation of N on it, and using that trace to prove the string is in at least one of A_1, A_2; then, taking an arbitrary string in $A_1 \cup A_2$ and proving that it is accepted by N. Details left for extra practice.

Friday: Automata constructions

Review: The language recognized by the NFA over $\{a, b\}$ with state diagram



$L(N_1) = L(b^* a (a \cup b)^* b)$

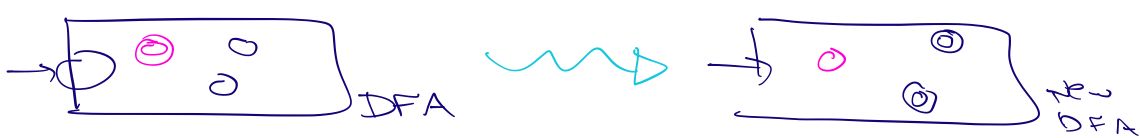
$= L(\{w \in \{a,b\}^* \mid w \text{ has an } a \text{ and ends in } b\})$

$L(N_2) = \{w \in \{a,b\}^* \mid w \text{ has even length}\}$

is: $L(N_1) \cup L(N_2)$

So far, we know:

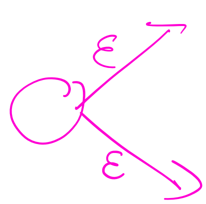
- The collection of languages that are each recognizable by a DFA is **closed** under complementation.
  Could we do the same construction with NFA? No//



L is in the collection of langs that are each recognizable by some DFA

$\overline{L}$ is also in this collection

- The collection of languages that are each recognizable by a NFA is **closed** under ~~complementation~~ union
  Could we do the same construction with DFA? No//

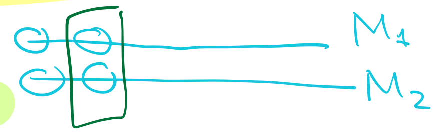


Not available in DFA

Happily, though, an analogous claim is true!

Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a DFA M_1 such that $L(M_1) = A_1$ and DFA M_2 such that $L(M_2) = A_2$, then there is another DFA, let's call it M, such that $L(M) = A_1 \cup A_2$. *Theorem 1.25 in Sipser, page 45*

Proof idea: Keep track of both computations.



Formal construction:

Consider A_1 over Σ, recognized by $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
and A_2 over Σ, recognized by $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

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

$Q = \{ (q, q') \mid q \in Q_1 \text{ and } q' \in Q_2 \} = Q_1 \times Q_2$

$q_0 = (q_1, q_2)$ $\quad F = \{ (q, q') \mid \overset{q \in Q}{q \in F_1} \text{ or } \overset{q' \in Q_2}{q' \in F_2} \}$

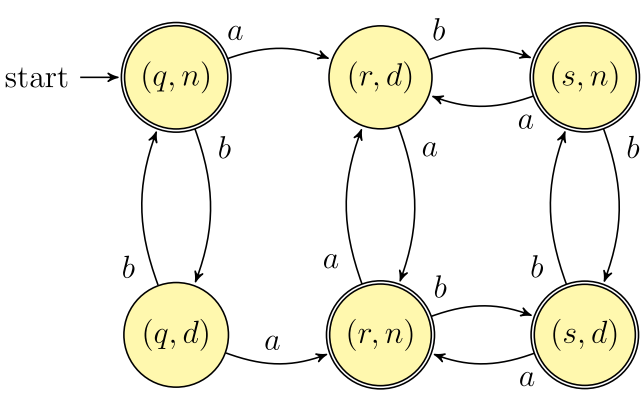
$= F_1 \times Q_2 \cup Q_1 \times F_2$

and $\delta : Q \times \Sigma \to Q$ is defined by

$$\delta(((q, q'), x)) = (\delta_1((q, x)), \delta_2((q', x)))$$

where $q \in Q_1$ $\quad q' \in Q_2$ $\quad x \in \Sigma$

Example: When $A_1 = \{w \mid w \text{ has an } a \text{ and ends in } b\}$ and $A_2 = \{w \mid w \text{ is of even length}\}$.



DFA with language $A_1 \cup A_2$.

Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a DFA M_1 such that $L(M_1) = A_1$ and DFA M_2 such that $L(M_2) = A_2$, then there is another DFA, let's call it M, such that $L(M) = A_1 \cap A_2$. *Footnote to Sipser Theorem 1.25, page 46*

Proof idea: Same construction, change F to $F_1 \times F_2$

Formal construction:

copy
paste for
Q, Σ, δ, q_0
from previous construction

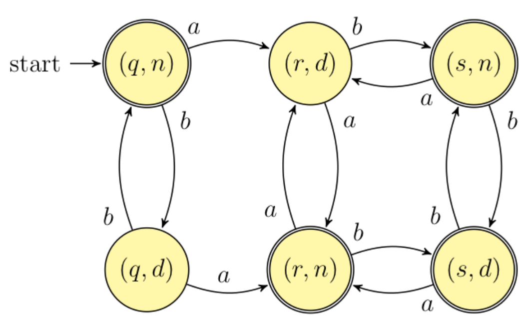
Bonus example:

Example: When $A_1 = \{w \mid w \text{ has an } a \text{ and ends in } b\}$ and $A_2 = \{w \mid w \text{ is of even length}\}$.

Design a DFA that recognizes $A_1 \cap A_2$

Hint:

modify



Where we are . . .

Closure of class of languages recognizable by DFA under
- complementation
- union
- intersection.

Closure of class of languages recognizable by NFA under
- union