Computational Models — Lecture 2[1]

Handout Mode

Ronitt Rubinfeld and Iftach Haitner.

Tel Aviv University.

March 07/09, 2016

[1] Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

Computational Models - Lecture 2

- Non-Deterministic Finite Automata (NFA)
- Closure of Regular Languages Under $\bigcup$, $\|$, *
- Regular expressions
- Equivalence with finite automata

- Sipser's book, $1.1 - 1.3$

Part I

Non-Deterministic Finite Automata

DFA – formal definition, (reminder)

Definition 1 (DFA)

A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set called the states,
- Σ is a finite set called the alphabet,
- $\delta : Q \times \Sigma \mapsto Q$ is the transition function,
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ is the set of accept states.

Formal model of computation, (reminder)

Definition 2

$M = (Q, \Sigma, \delta, q_0, F)$ accepts $w \in \Sigma^*$ if $\widehat{\delta}(q_0, w) \in F$.

Definition 3 ($\widehat{\delta}$)

For DFA $M = (Q, \Sigma, \delta, q_0, F)$, define $\widehat{\delta}\colon Q \times \Sigma^* \mapsto Q$ by

$$\widehat{\delta}(q, w) = \begin{cases} \delta(\widehat{\delta}(q, w_{1,\ldots,n-1}), w_n), & n = |w| \geq 1 \\ q, & w = \varepsilon. \end{cases}.$$

The language of a DFA, (reminder)

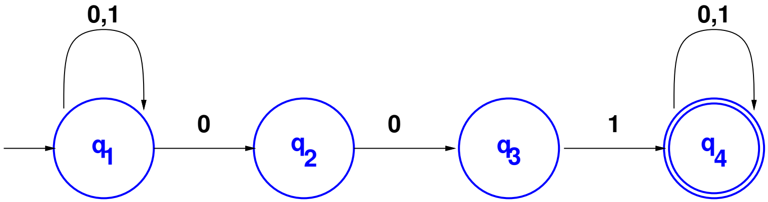
Definition 4

The language of a DFA M, denoted $\mathcal{L}(M)$, is the set of strings that M accepts.

Definition 5

A language is called regular, if some deterministic finite automaton accepts it.

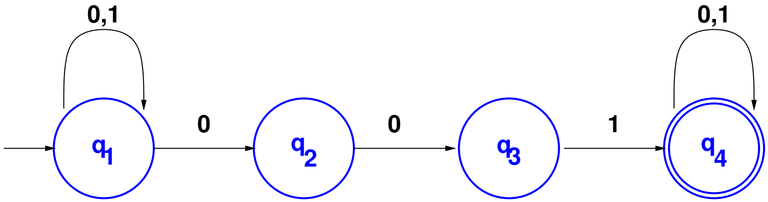
NFA — non-deterministic Finite Automata



- May have more than one transition labeled with the same symbol,
- May have no transitions labeled with a certain symbol,
- May have transitions labeled with ε, the symbol of the empty string. Will deal with this latter

Every DFA is also an NFA.

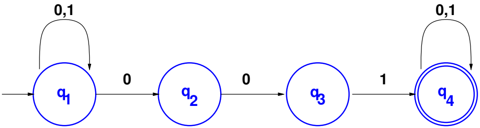
Non-deterministic computation

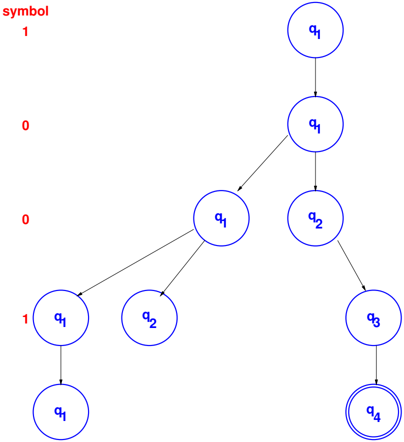


What happens when more than one transition is possible?

- The machine "splits" into multiple copies
- Each branch follows one possibility
- Together, branches follow all possibilities.
- If the input doesn't appear, that branch "dies".
- Automaton accepts if some branch accepts.

Computation on 1001





Why non-determinism?

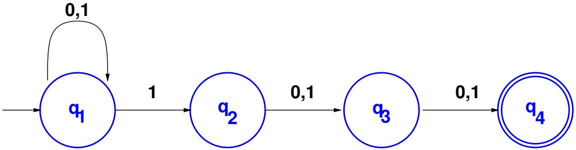
Theorem 6 (Informal, to be proved soon)

Deterministic and non-deterministic finite automata, accept exactly the same set of languages.

Q.: So why do we need NFA's?

Design a finite automaton for the language $\mathcal{L}$ — all binary strings with a 1 in their third-to-the-last position?

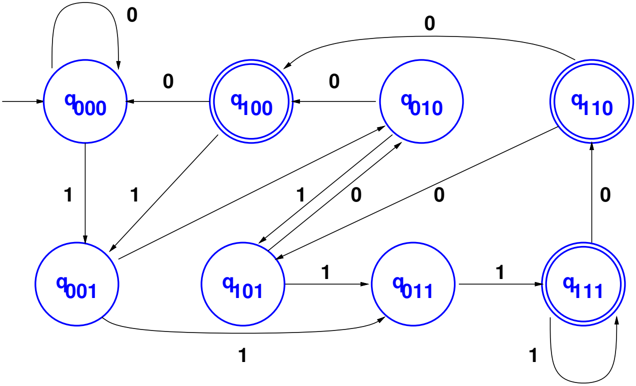
NFA for $\mathcal{L}$



- "Guesses" which symbol is third from the last, and
- checks that indeed it is a 1.
- If guess is premature, that branch "dies", and no harm occurs.

DFA for $\mathcal{L}$

- Have 8 states, encoding the last three observed letters.
- A state for each string in $\{0,1\}^3$.
- Add transitions on modifying the suffix, give the new letter.
- Mark as accepting, the strings $1 * *$



DFA has few bugs...

NFA – Formal Definition

Let $\mathcal{P}(Q)$ denote the powerset of Q (i.e., all subsets of Q).

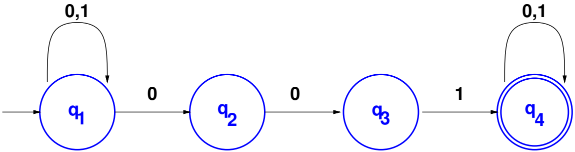
Definition 7 (NFA)

A non-deterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, S, F)$, where

- Q is a finite set called the states
- Σ is a finite set called the alphabet
- $\delta\colon Q \times \Sigma \mapsto \mathcal{P}(Q)$ is the transition function
- $S \subseteq Q$ is the set of starting states
- $F \subseteq Q$ are the set of accepting states

We sometimes consider an NFA $(Q, \Sigma, \delta, q_0, F)$.
This is merely a “syntactic sugar” for the NFA $(Q, \Sigma, \delta, \{q_0\}, F)$

Example



$N_1 = (Q = \{q_1, q_2, q_3, q_4\}, \Sigma = \{0, 1\}, \delta, S = \{q_1\}, F = \{q_4\})$

for δ defined by

	0	1
q_1	$\{q_1, q_2\}$	$\{q_1\}$
q_2	$\{q_3\}$	$\emptyset$
q_3	$\emptyset$	$\{q_4\}$
q_4	$\{q_4\}$	$\{q_4\}$

Not that $\emptyset$ is a valid output for δ

Formal model of computation

Definition 8

$N = (Q, \Sigma, \delta, S, F)$ accepts $w \in \Sigma^*$, if $\widehat{\delta}_N(S, w) \cap F \neq \emptyset$.

Definition 9 ($\widehat{\delta}$)

For NFA $N = (Q, \Sigma, \delta, S, F)$, define $\widehat{\delta}_N \colon P(Q) \times \Sigma^* \mapsto \mathcal{P}(Q)$ by:

for $Q' \subseteq Q$, $\widehat{\delta}_N(Q', w) = \begin{cases} Q', & w = \varepsilon, \\ \bigcup_{q \in \widehat{\delta}_N(Q', w_{1,\ldots,n-1})} \delta(q, w_n), & n = |w| \geq 1. \end{cases}$.

When clear from he context we will write $\widehat{\delta}$ (i.e., omitting the $_N$).

An equivalent definition

Definition 10 (Equivalent definition)

$N = (Q, \Sigma, \delta, S, F)$ accepts $w = w_1, \ldots, w_n \in \Sigma^n$, if if $\exists r_0, \ldots, r_n \in Q$ s.t.

- $r_0 \in S$
- $r_n \in F$
- $r_{i+1} \in \delta(r_i, w_{i+1})$, for all $0 \leq i < n$.

Equivalence of NFA's and DFA's

Easy: For any DFA M there exists a NFA N such that $\mathcal{L}(N) = \mathcal{L}(M)$.

Other direction is also true.

Theorem 11

For any NFA N there exists a DFA M such that $\mathcal{L}(N) = \mathcal{L}(M)$.

- Given an NFA N, we construct a DFA M, that accepts the same language.
- Make DFA emulates all possible NFA states.
- As consequence of the construction, if the NFA has k states, the DFA has 2^k states (an exponential blow up).

Equivalence of NFA's and DFA's, the DFA

Let $N = (Q, \Sigma, \delta, S, F)$.

Construction 12 ($M = (Q_M, \Sigma, \delta_M, d_0, F_M)$)

- $Q_M = \{[R] \colon R \subseteq Q\}$.
- $d_0 = [S]$
- $F_M = \{[R] \in Q_M \colon R \cap F \neq \emptyset\}$
- For $[R] \in Q_M$ and $\sigma \in \Sigma$, let $\delta_M([R], \sigma) = [\widehat{\delta}_N(R, \sigma)] \quad (= [\bigcup_{r \in R} \delta(r, \sigma)])$.

To prove equivalence, we need to prove that

$$\widehat{\delta}_N(S, w) \cap \mathcal{F} \neq \emptyset \Longleftrightarrow \widehat{\delta}_M(d_0, w) \in F_M$$

The above is an immediate corollary of the following claim:

Claim 13

$[\widehat{\delta}_N(S, w)] = \widehat{\delta}_M(d_0, w)$ for every $w \in \Sigma^*$.

Proving $[\widehat{\delta}_N(S, w) = \widehat{\delta}_M(d_0, w)]$

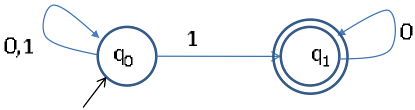
The proof is by induction on the length of w.

- $|w| = 0$, by definition.
- Assume for words of length $(m-1)$, and let $x = y\sigma$, where y is a word of length $(m-1)$ and $\sigma \in \Sigma$.
- Let $Q_y = \widehat{\delta}_N(S, y)$ and $d_y = \widehat{\delta}_M(d_0, y)$.
- Compute

$$\begin{aligned}
\widehat{\delta}_M(d_0, x) &= \delta_M(d_y, \sigma) && \text{(By definition of } \widehat{\delta}_M \text{)}\\
&= \delta_M([Q_y], \sigma) && \text{(By i.h)}\\
&= [\widehat{\delta}_N(Q_y, \sigma)] && \text{(By definition of } \delta_M \text{)}\\
&= [\bigcup_{q \in Q_y} \delta(q, \sigma)] && \text{(By definition of } \widehat{\delta}_N \text{)}\\
&= [\bigcup_{q \in \widehat{\delta}_N(S,y)} \delta(q, \sigma)] &&\\
&= [\widehat{\delta}_N(S, x)]. \quad \square && \text{(By definition of } \widehat{\delta}_N \text{)}
\end{aligned}$$

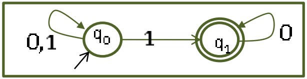
Example: NFA $\Rightarrow$ DFA

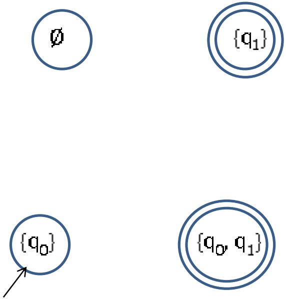
Non-Deterministic Automata:



Example: NFA $\Rightarrow$ DFA

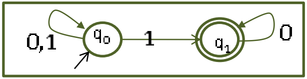
Deterministic automata - set of states:

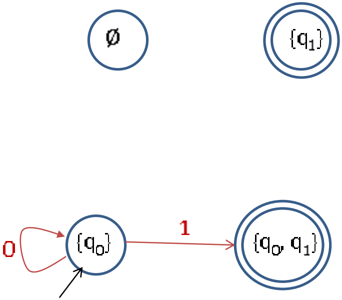




Example: NFA $\Rightarrow$ DFA

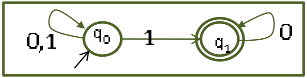
Transitions from $[\{q_0\}]$:

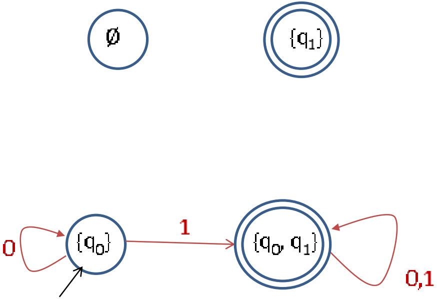




Example: NFA $\Rightarrow$ DFA

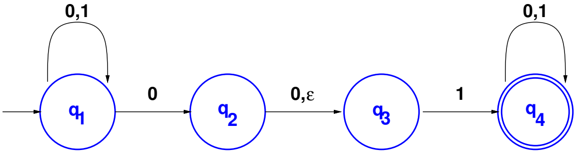
Transitions from $[\{q_0, q_1\}]$:





Transitions from $[\emptyset]$ and $[\{q_1\}]$?

NFA with ε-moves

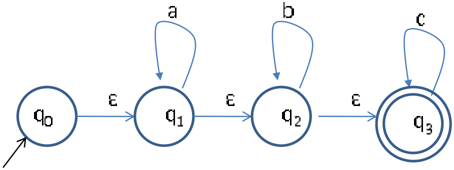


What is the interpretation of ε transitions ?

What will happen with 101 ?

Example: NFA with ε-moves

$$\mathcal{L} = \{a^i b^j c^k | i, j, k \geq 0\}$$



NFA — Formal definition with ε-moves

Transition function δ is going to be different.

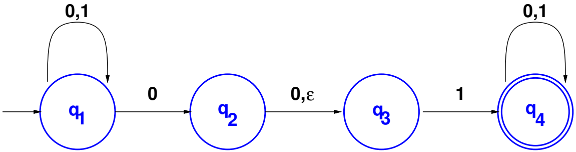
- Let $\mathcal{P}(Q)$ denote the powerset of Q.
- Let Σ_ε denote the set $\Sigma \cup \{\varepsilon\}$.

Definition 14 (NFA, with ε-moves)

A non-deterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, S, F)$:

- Q is a finite set called the states
- Σ is a finite set called the alphabet
- $\delta : Q \times \Sigma_\varepsilon \mapsto \mathcal{P}(Q)$ is the transition function
- $S \subseteq Q$ is the set of starting state
- $F \subseteq Q$ is the set of accepting states

Example



$N_1 = (Q, \Sigma, \delta, S, F)$:

- $Q = \{q_1, q_2, q_3, q_4\}$, $\Sigma = \{0, 1\}$, $S = \{q_1\}$ and $F = \{q_4\}$.
- δ is

	0	1	ε
q_1	$\{q_1, q_2\}$	$\{q_1\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

Formal model of computation, with ε-moves

Definition 15

$N = (Q, \Sigma, \delta, S, F)$ accepts $w \in \Sigma^*$, if $\widehat{\delta}_N(S, w) \cap F \neq \emptyset$.

Definition 16

For NFA $N = (Q, \Sigma, \delta, S, F)$, let
$E(q) = \{q' \in Q \colon q'$ can be reached from q by 0 or more ε transitions$\}$
(i.e., $\{q' \colon \exists q_1, \ldots, q_k \in Q \text{ s.t. } q_1 = q \wedge q_k = q' \wedge \forall i \in [k-1] \;\; q_{i+1} \in \delta(q_i, \varepsilon)\}$)

$E(Q') = \bigcup_{q \in Q'} E(q)$.

Q: is it always the case that $q \in E(q)$? Yes

Definition 17 ($\widehat{\delta}$)

For NFA $N = (Q, \Sigma, \delta, S, F)$, define $\widehat{\delta}_N \colon P(Q) \times \Sigma^* \mapsto \mathcal{P}(Q)$ by:

for $Q' \subseteq Q$, $\widehat{\delta}_N(Q', w) = \begin{cases} E(Q'), & w = \varepsilon, \\ E\left(\bigcup_{r \in \widehat{\delta}(Q', w_1, \ldots, w_{n-1})} \delta(r, w_n)\right), & n = |w| \geq 1. \end{cases}$.

When does N accept the empty string?

An equivalent definition

For $a \in (\Sigma_\varepsilon)^*$, let $d(a) \in \Sigma^*$ be a without the ε symbols.

Example: $d(\varepsilon 01 \varepsilon\varepsilon 3 \varepsilon) = 013$

Definition 18 (Equivalent definition)

$N = (Q, \Sigma, \delta, S, F)$ accepts $w \in \Sigma^*$, if exist $a = (a_1 a_2 \ldots a_k) \in (\Sigma_\varepsilon)^k$ and $r_0, \ldots, r_k \in Q$ s.t.

- $w = d(a)$.
- $r_0 \in S$
- $r_k \in F$
- $r_{i+1} \in \delta(r_i, a_{i+1})$, for all $0 \leq i < k$.

Removing ε-transitions

Given NFA $N = (Q, \Sigma, \delta, S, F)$ with ε-transitions, we create an equivalent NFA $N' = (Q, \Sigma, \delta', S', F)$ with no ε-transitions.

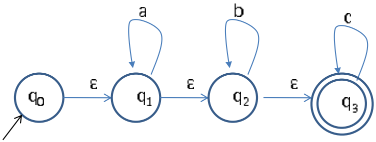
- $S' = E(S)$
- $\delta'(q, a) = E(\delta(q, a))$

It is not hard to prove that $\widehat{\delta}_N(S, w) = \widehat{\delta}_{N'}(S', w)$ for any $w \in \Sigma^*$.

Thus, $\mathcal{L}(N) = \mathcal{L}(N')$.

Example: Removing ε-transitions

Non-Deterministic Automata with ε-transitions



The non-Deterministic automata without ε-transitions

- $S' = \{q_0, q_1, q_2, q_3\}$
- δ' is

	a	b	c
q_0	$\emptyset$	$\emptyset$	$\emptyset$
q_1	$\{q_1, q_2, q_3\}$	$\emptyset$	$\emptyset$
q_2	$\emptyset$	$\{q_2, q_3\}$	$\emptyset$
q_3	$\emptyset$	$\emptyset$	$\{q_3\}$

Part II

Closure of Regular Languages, Revisited

Regular languages, revisited

By definition, a language is regular if it is accepted by some DFA.

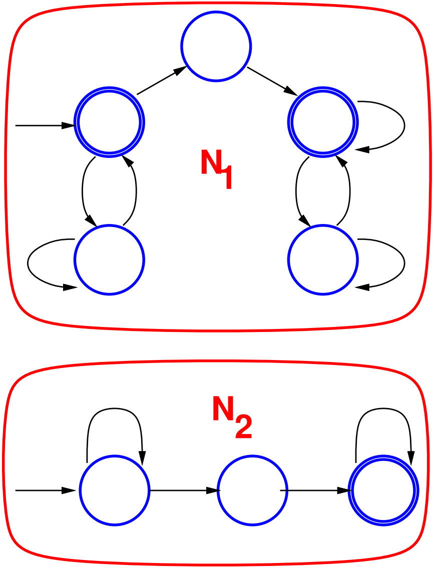
Corollary 19

A language is regular if and only if it is accepted by some NFA.

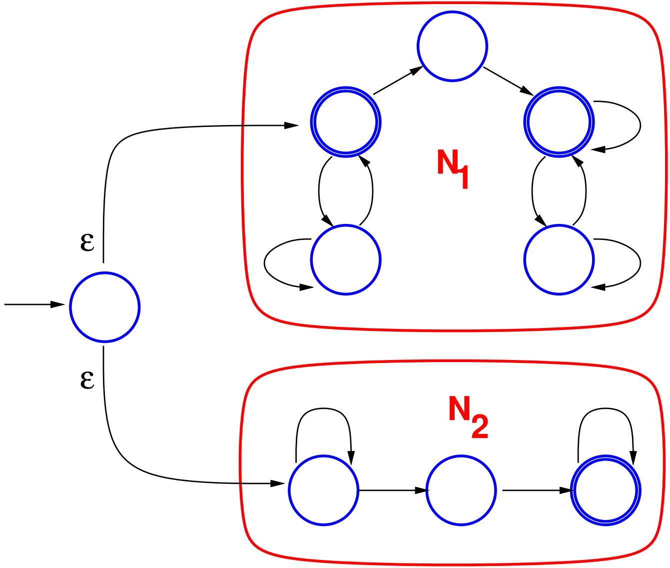
This is an alternative way of characterizing regular languages.

We will now use the equivalence to show that regular languages are closed under the regular operations (union, concatenation, star).

Closure under union (alternative proof)



Closure under union, cont.



Closure under union cont..

- NFA $N_1 = (Q_1, \Sigma, \delta_1, S_1, F_1)$ accept $\mathcal{L}_1$, and
- NFA $N_2 = (Q_2, \Sigma, \delta_2, S_2, F_2)$ accept $\mathcal{L}_2$.

Wlg. that $Q_1 \cap Q_2 = \emptyset$.(?)

Define NFA $N = (Q = \{q_0\} \cup Q_1 \cup Q_2, \Sigma, \delta, S = \{q_0\}, F = F_1 \cup F_2)$,

for $\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ S_1 \cup S_2 & q = q_0 \text{ and } a = \varepsilon \end{cases}$

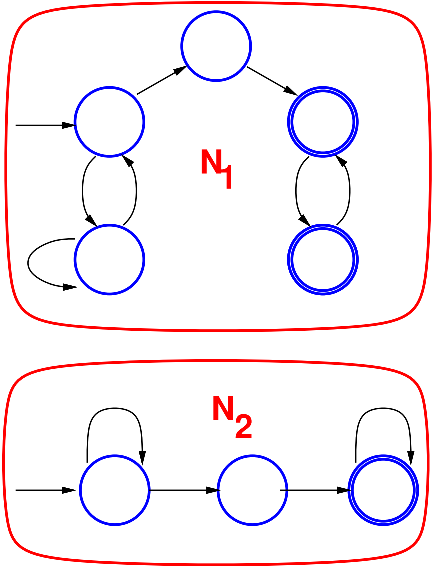
Alternatively, let $S = S_1 \cup S_2$ and omit the last line of δ.

Claim 20

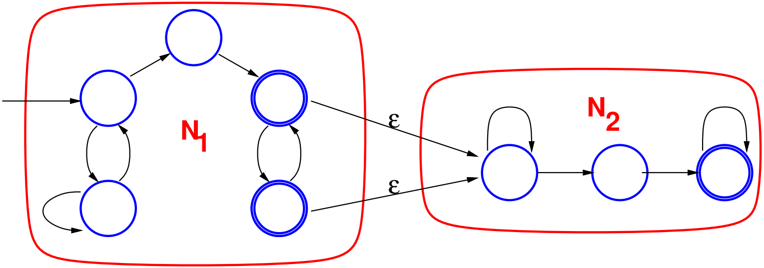
$\mathcal{L}(N) = \mathcal{L}(N_1) \cup \mathcal{L}(N_2)$.

Proof: ?

Closure under concatenation



Closure under concatenation, cont.



Remark: Final states are exactly those of N_2.

Closure under concatenation, cont..

- NFA $N_1 = (Q_1, \Sigma, \delta_1, S_1, F_1)$ accept $\mathcal{L}_1$
- NFA $N_2 = (Q_2, \Sigma, \delta_2, S_2, F_2)$ accept $\mathcal{L}_2$

Define NFA $N = (Q_1 \cup Q_2, \Sigma, \delta, S_1, F_2)$:

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \wedge\ a \neq \varepsilon \\ \delta_1(q,a) & (q \in Q_1 \setminus F_1) \wedge\ a = \varepsilon \\ \delta_1(q,a) \cup S_2 & q \in F_1 \wedge\ a = \varepsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

Claim 21

$\mathcal{L}(N) = \mathcal{L}_1 \| \mathcal{L}_2$.

Proof: Need to prove $w \in \mathcal{L}(N) \iff w \in \mathcal{L}_1 \| \mathcal{L}_2$.

Proving $w \in \mathcal{L}(N) \iff \mathcal{L}_1 \| \mathcal{L}_2$

Wlg. N_1 and N_2 have no ε move.(?)

Assume $w \in \mathcal{L}_1 \| \mathcal{L}_2$:

$\implies$ $\exists w^1 \in \mathcal{L}_1$ and $w^2 \in \mathcal{L}_2$, s.t. $w = w^1 w^2$

$\implies$ $\exists\ r^1_0, \ldots r^1_{|w^1|}$ and $r^2_0, \ldots r^2_{|w^2|}$, such that for both $j \in \{1, 2\}$:

(1) $r^j_0 \in S_j$ (2) $r^j_{|w^j|} \in F_j$ (3) $\forall\ 0 \le i < |w^j|$: $r^j_{i+1} \in \delta_j(r^j_i, w^j_{i+1})$.

$\implies$ (details...) $r^1_0, \ldots r^1_{|w^1|}, r^2_0, \ldots r^2_{|w^2|}$ proves that $r^2_{|w^2|} \in \widehat{\delta}(S_1, w)$

$\implies$ $w \in \mathcal{L}(N)$

Assume $w \in \mathcal{L}(N)$:

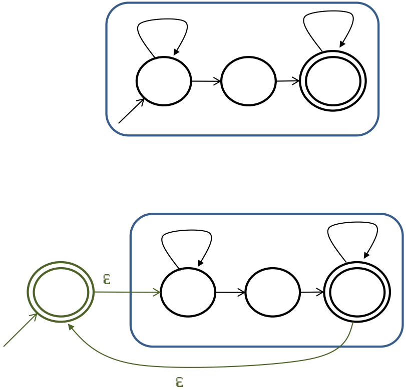
$\implies$ $\exists$ parsing $w = a_1 a_2 \ldots a_k \in (\Sigma_\varepsilon)^k$ and $r_0 \ldots r_k$ such that:

(1) $r_0 \in S_1$ (2) $r_k \in F_2$ (3) $\forall\ 0 \le i < k$: $r_{i+1} \in \delta(r_i, a_{i+1})$.

- Let j be the last index such that $r_j \in Q_1$
- By construction (details...) (1) $a_{j+1} = \varepsilon$ (2) $r_0 \ldots r_j$ proves that $a_1, \ldots, a_j \in \mathcal{L}_1$ (3) $r_{j+1} \ldots r_k$ proves that $a_{j+2}, \ldots, a_k \in \mathcal{L}_2$

$\implies$ $w = a_1, \ldots, a_j, \varepsilon, a_{j+2}, \ldots, a_k \in \mathcal{L}_1 \| \mathcal{L}_2$ $\square$

Closure under Star



Closure under star, cont.

Let $N = (Q, \Sigma, \delta, S, F)$ accepting $\mathcal{L}$, assuming wlg. that $q_0 \notin Q$.

Define $N' = (Q' = Q \cup \{q_0\}, \Sigma, \delta', S' = \{q_0\}, F' = \{q_0\})$:

$$\delta'(q, a) = \begin{cases} \delta(q, a) & q \in Q \wedge a \neq \varepsilon \\ \delta(q, \varepsilon) & q \notin F \wedge a = \varepsilon \\ \delta(q, \varepsilon) \cup \{q_0\} & q \in F \wedge a = \varepsilon \\ S & q = q_0 \wedge a = \varepsilon \end{cases}$$

Claim 22

$\mathcal{L}(N') = \mathcal{L}(N)^*$.

Proof: ?

Summary

- Regular languages are closed under
  - union
  - concatenation
  - star
- Non-deterministic finite automata
  - are equivalent to deterministic finite automata
  - but much easier to use in some proofs and constructions.

Part III

Regular Expressions

Regular expressions

Notation for building up languages by describing them as expressions, e.g., $(0 \cup 1)0^*$.

- 0 and 1 are shorthand for the set (languages) $\{0\}$ and $\{1\}$
- so $0 \cup 1 = \{0, 1\}$.
- 0^* is shorthand for $\{0\}^*$.
- Concatenation, is implicit. So 0^*10^* stands for $\{w \in \{0,1\} \colon\ w$ havs exactly a single $1\}$.
- Just like in arithmetic, operations have precedence:
  - star first
  - concatenation next
  - union last
  - parentheses used to change default order i.e., $ab^* \neq (ab)^*$

Q.: What does $(0 \cup 1)0^*$ stand for?

Remark: Regular expressions are often used in text editors or shell scripts.

Regular expressions – formal definition

Definition 23

A string R is a regular expression over Σ, if R is of form

- a for some $a \in \Sigma$
- ε
- $\emptyset$
- $(R_1 \cup R_2)$ for regular expressions R_1 and R_2
- $(R_1 \| R_2)$ for regular expressions R_1 and R_2
- (R_1^*) for regular expression R_1

$R(\Sigma)$ denotes all (finite) regular expression over Σ.

Parenthesis and $\|$ are omitted when their role is clear from the context.

Formal Definition, cont.

Definition 24

The language $\mathcal{L}(R)$ of regular expression R, is defined by

R	$\mathcal{L}(R)$
a	$\{a\}$
ε	$\{\varepsilon\}$
$\emptyset$	$\emptyset$
$(R_1 \cup R_2)$	$\mathcal{L}(R_1) \cup \mathcal{L}(R_2)$
$(R_1 R_2)$	$\mathcal{L}(R_1) \| \mathcal{L}(R_2)$
(R_1^*)	$\mathcal{L}(R_1)^*$

Isn't this definition circular?

Examples of regular expressions

For $\Sigma = \{0, 1\}$, write regular expression for the following languages:

- The third letter from the end is 1

  $(0 \cup 1)^* 1 (0 \cup 1)^2$

- The number of 1's is even

  $(0^* \cup 10^* 1)^*$

- The number of 1's is odd

  $(0^* \cup 10^* 1)^* 10^*$

Part IV

Regular Expressions and Regular Languages

Remarkable Fact

Theorem 25

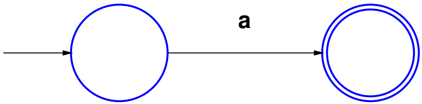
A language is described by a regular expression iff it is regular.

$\Longleftarrow$: Given a regular language, construct a regular expression describing it.

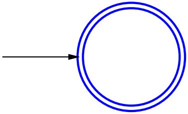
$\Longrightarrow$: Given a regular expression, construct an NFA accepting its language.

Given RE R, build NFA Accepting it ($\Longrightarrow$)

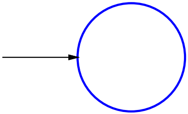
1. $R = a$, for some $a \in \Sigma$



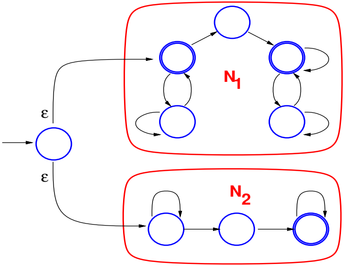
2. $R = \varepsilon$



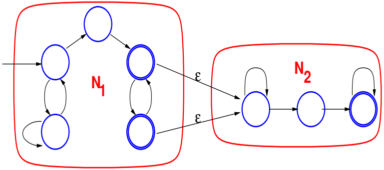
3. $R = \emptyset$



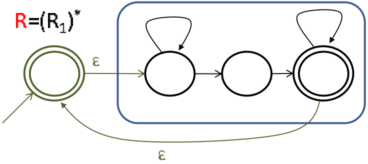
Given R, Build NFA Accepting It ($\Longrightarrow$), cont.



$R = (R_1 \cup R_2)$

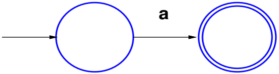


$R = (R_1 \| R_2)$

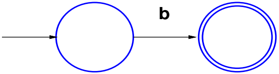


Example

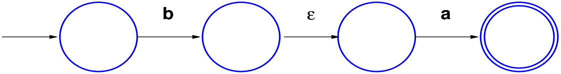
a



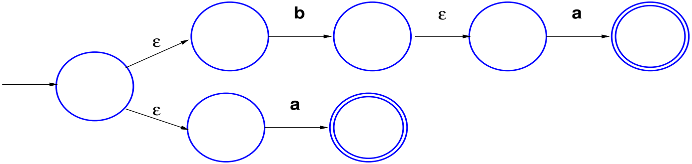
b



ba



ba ∪ a



Formal proof by induction on the length of the regular expression

Regular expression from a DFA ($\Longleftarrow$)

Easy for “non-circular" DFA (board), but more complicated for general DFA’s.

NFA:

- Each transition is labeled with a symbol or ε.
- Reads zero or one symbols.
- Takes matching transition, if any.

Generalized non-deterministic finite automata (GNFA):

- Each transition is labeled with a regular expression.
- Reads zero or more symbols.
- Takes matching regular expression, if any.

Example (board).

GNFAs are natural generalization of NFAs.

GNFA – Formal Definition

Let $\mathcal{R}(\Sigma)$ be the set of regular expressions over Σ.

Definition 26

A generalized deterministic finite automaton (GNFA) is $(Q, \Sigma, \delta, q_s, q_a)$

- Q is a finite set of states
- Σ is the alphabet
- $\delta : (Q \setminus \{q_a\}) \times (Q \setminus \{q_s\}) \mapsto \mathcal{R}(\Sigma)$ is the transition function.
- $q_s \in Q$ is the start state
- $q_a \in Q$ is the unique accept state

It is a special type of GNFA, but still it is easy to transform any DFA/NFA into this form.

GNFA – Model of Computation

Definition 27

A GNFA $G = (Q, \Sigma, \delta, q_s, q_a)$ accepts a string $w \in \Sigma^*$, if there exists

- parsing $w = a_1 a_2 \cdots a_k \in (\Sigma^*)^k$, and
- $r_0, \ldots, r_k \in Q$,

such that

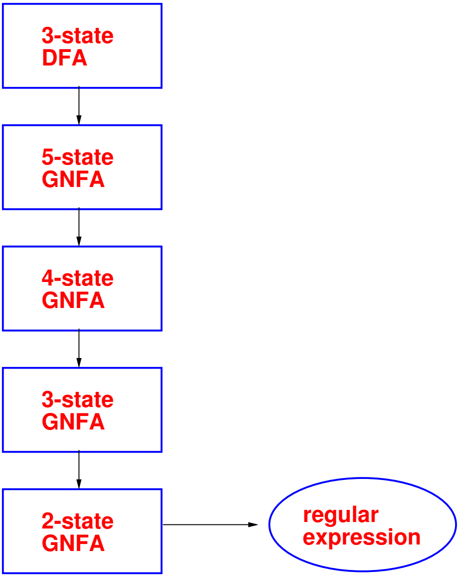
- $r_0 = q_s$
- $r_k = q_a$
- $a_i \in \mathcal{L}(\delta(r_{i-1}, r_i))$, for every $0 < i \leq k$.

The Transformation: DFA $\rightarrow$ Regular Expression

Strategy – sequence of equivalent transformations

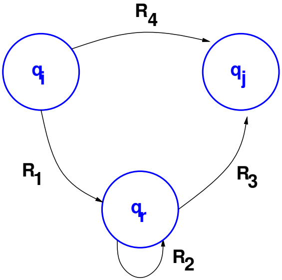
- Given a k-state DFA
- Transform into $(k+2)$-state GNFA (how?)
- While GNFA has more than 2 states, transform it into equivalent GNFA with one fewer state
- Eventually reach 2-state GNFA (states are just start and accept).
- Label of single transition is the desired regular expression. ♠

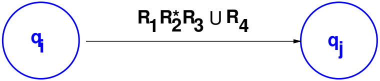
Converting strategy ($\Longleftarrow$)



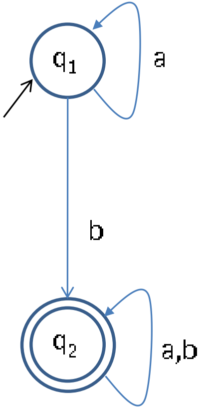
Removing a state

We remove one state q_r, and then repair the machine by altering regular expression of other transitions.

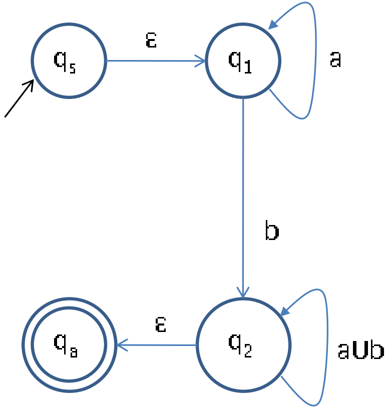




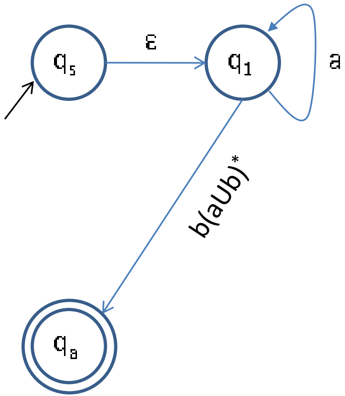
Conversion - Example



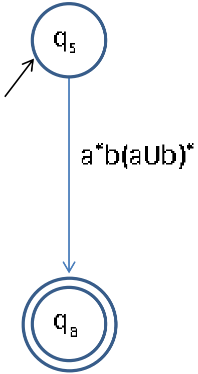
Conversion - Example



Conversion - Example



Conversion - Example



The StateReduce and Convert algorithms

Algorithm 28 (StateReduce)

Input: a $(k > 2)$-state GNFA $G = (Q, \Sigma, \delta, q_s, q_a)$.

- Select any state $q_r \in Q \setminus \{q_s, q_a\}$.
- Let $Q' = Q \setminus \{q_r\}$.
- For any $q_i \in Q' \setminus \{q_a\}$ and $q_j \in Q' \setminus \{q_s\}$, let
  - $R_1 = \delta(q_i, q_r)$, $R_2 = \delta(q_r, q_r)$,
  - $R_3 = \delta(q_r, q_j)$, and $R_4 = \delta(q_i, q_j)$.
- Define $\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4)$.
- Return the resulting $(k-1)$-state GNFA $G' = (Q', \Sigma, \delta', q_s, q_a)$.

Algorithm 29 (Convert)

Input: a $(k \geq 2)$-state GNFA G.

- If $k = 2$, return the regular expression labeling the only arrow of G.
- Otherwise, return Convert(StateReduce(G)).

Correctness proof

Claim 30

G and Convert(G) accept the same language.

Proof: By induction on k – the number of states of G.

Basis. $k = 2$: Immediate by the definition of GFNA.

Induction step: Assume claim for $(k-1)$-state GNFA, where $k > 2$, prove for k-state GNFA.

Let $G' = \text{StateReduce}(G)$ (note that G' has $k-1$ states), and let q_r be the removed state.
We prove (in a very high level) that $\mathcal{L}(G) = \mathcal{L}(G')$ (i.e., G and G' accept the same language). We show

- $w \in \mathcal{L}(G) \implies w \in \mathcal{L}(G')$
- $w \in \mathcal{L}(G') \implies w \in \mathcal{L}(G)$

$w \in \mathcal{L}(G) \implies w \in \mathcal{L}(G')$

Let $w \in \mathcal{L}(G)$ and let $p = q_s, q_1, \ldots, q_\ell, q_a$ be (a possible) "path of states" traversed by G on w.

- If $q_r \notin p$, then G' accepts w (the new regular expression on each edge of G' contains the old regular expression in the "union part".)
- If $p = q_S, \ldots, q_i, q_r, q_j, \ldots, q_a$, the regular expression $(R_{i,r})(R_{r,r})^*(R_{r,j})$ linking q_i and q_j in G', causes G' to accept w.

Hence, $w \in \mathcal{L}(G')$.

$w \in \mathcal{L}(G') \implies w \in \mathcal{L}(G)$

Let $w \in \Sigma^*$ and let $p = q_0, \ldots, q_n$ be be (a possible) "path of states" traversed by G' on w.

There exists parsing $w = w_1, \ldots, w_n$ such that $w_i \in \mathcal{L}(R'_i)$ for $R'_i = \delta_{G'}(q_{i-1}, q_i)$.

For $q_i, q_j \in Q$, let $R_{i,j} = \delta_G(q_i, q_j)$.

Hence, $\delta_{G'}(q_{i-1}, q_i) = \left(R_{i-1,r} R^*_{r,r} R_{r,i}\right) \cup R_{i-1,i}$.

- If $w_i \in \mathcal{L}(R_{i-1,i})$, then $w_i \in \delta_G(q_{i-1}, q_i)$.
- If $w_i \in \mathcal{L}(R_{i-1,r} R^*_{r,r} R_{r,i})$, then $w_i = u_1, \ldots u_\ell$ such that
  - $u_1 \in \mathcal{L}(R_{i-1,r})$
  - $u_\ell \in \mathcal{L}(R_{r,i})$
  - $u_j \in \mathcal{L}(R_{r,r})$ for $2 \leq j \leq \ell - 1$.

In both cases, w_i corresponds to possible traverse from q_{i-1} to q_i in G.

Hence, $w \in \mathcal{L}(G') \implies w \in \mathcal{L}(G)$.

Summing it up

- We proved $\mathcal{L}(G) = \mathcal{L}(G')$.
- Hence, G and (the regular expression) Convert(G) accept the same language.
- Thus, we proved: Every regular language can be described by a regular expression.

Summary

- Non-Deterministic Automata (with ε-moves)
  - Equivalence to DFA
- Closure properties
  - union
  - concatenation
  - star
- Regular expressions.
  - Equivalence to DFA