Finite state machines

16/10/2008 (2.1)
(continued)

- Finite automata:

  - simplest model of computation
  - describes so called "regular languages"
  - works as follows:
    - is always in one of finitely-many states
    - starts in some state
    - changes state in response to input
    - accepts input by ending in an accepting (or final) st.

Example: F.A. scanning HTML documents for a list of football-game results

Observations:

- Σ = HTML tags $\cup$ ASCII characters
- each result stored in the form:

  team1 ␣–␣ team2 ␣–␣ n:m

- list represented as HTML list:

  <OL> ... ordered list

  <UL> ... unordered list

  <LI> .. list item

- accepts when it finds end of list

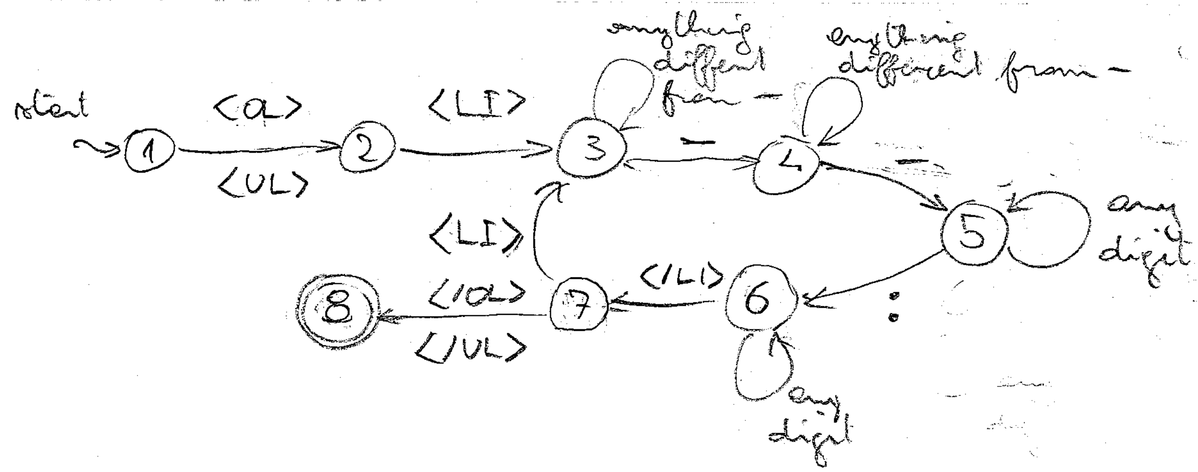
Example:

<OL>

<LI> Roma – Lazio – 2:0 </LI>

<LI> Inter – Juve – 10:2 </LI>

</OL>



<u>Notation</u> in the state transition diagram:

- state (5)
- start state → (1) (or initial state)
- final state ((8)) (or accepting state)
- transition

$$3 \xrightarrow{\_} 4$$

meaning: when the F.A. is in state (3) and it sees a '_' in the input, it moves to state (4) and advances on the input

<u>Example</u>: - describe using a set-former the language of all binary strings that contain the pattern 01.
- construct a F.A. that accepts the language

Solution: $\Sigma = \{0, 1\}$

$L = \{ w \in \Sigma^* \mid w \text{ has substring } 01 \} =$

$= \{ x01y \mid x, y \in \Sigma^* \}$

q_0 ... waiting for first 0

q_1 ... seen 0, waiting for 1

q_2 ... seen 01, waiting for rest of input

Note: • FA scans input from left to right (cannot go back) making transitions

• accepts if it is in an accepting state when it reaches the end of the input

Language accepted by a FA A: $\mathcal{L}(A) = \{w \in \Sigma^* \mid A \text{ accepts } w\}$

What we have seen are called Deterministic Finite Automata (DFAs)

Definition: a DFA is a quintuple

$$A = (Q, \Sigma, \delta, q_0, F)$$

- Q ... finite nonempty set of states e.g. $Q = \{q_0, q_1, q_2\}$
- Σ ... input alphabet e.g. $\Sigma = \{0, 1\}$
- q_0 ... initial (or start state)

  $q_0 \in Q$
- F ... set of final (or accepting) states

  $F \subseteq Q$ e.g. $F = \{q_2\}$
- δ ... total function $\delta : Q \times \Sigma \to Q$

  called state transition function

  Can be represented - as a diagram
  - as a transition table

e.g.

δ	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

means:

$\delta(q_0, 0) = q_1$ $\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_1$ $\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_2$ $\delta(q_2, 1) = q_2$

Note: we have still not defined formally what the language accepted by a DFA is

Extended transition function:

- we want to extend δ to multiple transitions

$$\delta : Q \times \Sigma \to Q$$
$$\hat{\delta} : Q \times \Sigma^* \to Q$$

meaning: $\hat{\delta}(q, x) = p$

denotes that starting at state q, portion x of input string will take DFA to state p

In other words: if $x = a_1 \cdots a_n$ and

$\delta(q, a_1) = p_1 \quad \delta(p_1, a_2) = p_2 \cdots \delta(p_{n-1}, a_n) = p$

then $\hat{\delta}(q, a_1 \cdots a_n) = p$

We can define $\hat{\delta}$ formally by induction:

$\forall q \in Q, \ \forall a \in \Sigma, \ \forall x \in \Sigma^*$

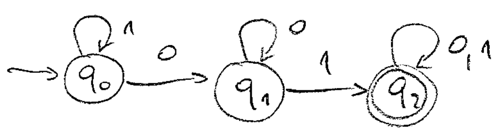
Basis: $\hat{\delta}(q, \varepsilon) = q$

Induction: $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

Note: we exploit the fact that strings are defined inductively

- ε is a string
- if x is a string and $a \in \Sigma$ then xa is a string
- nothing else is a string

Example:



$\hat{\delta}(q_0, \varepsilon) = q_0$

$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \varepsilon), 1) = \delta(q_0, 1) = q_0$

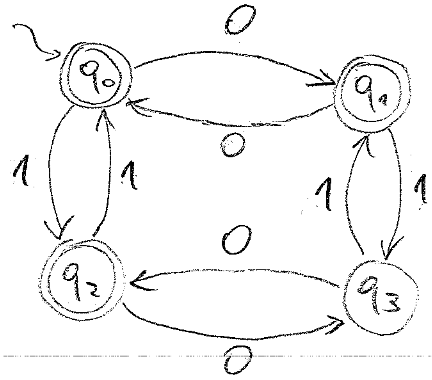
$\hat{\delta}(q_0, 10) = \delta(\hat{\delta}(q_0, 1), 0) = \delta(q_0, 0) = q_1$

$\hat{\delta}(q_0, 101) = \delta(\hat{\delta}(q_0, 10), 1) = \delta(q_1, 1) = q_2$

Language accepted by a DFA $A = (Q, \Sigma, \delta, q_0, F)$

Definition: $\mathcal{L}(A) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F \}$

Example:



What is $L(A)$?

Strings over $\Sigma = \{0, 1\}$ that contain an even number of 0's or an even number of 1's

This DFA partitions the strings over $\Sigma = \{0,1\}$ in 4 equivalence classes, depending on the parity of the numbers of 0's and 1's.

This is a general property:

- each DFA partitions the strings into a finite number of equivalence classes, and conversely
- each partition of strings into a finite number of equivalence classes corresponds to a DFA

Fact: $\forall a \in \Sigma, \quad \forall q \in Q$

$$\hat{\delta}(q, a) = \delta(q, a)$$

Proof: $\hat{\delta}(q, a) = \delta(\hat{\delta}(q, \epsilon), a) = \delta(q, a)$

Consequence: δ and $\hat{\delta}$ agree on strings of length 1

Also, δ is defined only for strings of length 1,

Hence we can adopt the convention to call $\hat{\delta}$ as δ.

Exercise 2.2.2: Prove that $\forall q \in Q$, $\forall x, y \in \Sigma^*$

$$\hat{\delta}(q, x \cdot y) = \hat{\delta}(\hat{\delta}(q, x), y)$$

Hint: use induction on $|y|$

Exercise 2.2.5: Give DFA's that accept the set of all strings over $\Sigma = \{0, 1\}$ such that:

a) each consecutive block of 5 symbols contains at least two 0's

b) the 10th symbol from the right is a 1
(don't try to write down the whole DFA!)

c) the string either begins or ends (or both) with 01

d) the number of 0's is divisible by 5, and the number of 1's is divisible by 3

Optional exercises:

Exercise 2.2.8: Let A be a DFA such that for some $a \in \Sigma$ and all $q \in Q$ we have $\delta(q, a) = q$

a) Show that for all $n > 0$, $\hat{\delta}(q, a^n) = q$

b) Show that either $\{a\}^* \subseteq \mathcal{L}(A)$ or $\{a\}^* \cap \mathcal{L}(A) = \emptyset$

Exercise 2.2.9: Let $A = (Q, \Sigma, \delta, q_0, \{q_f\})$ be a DFA such that for all $a \in \Sigma$ we have $\delta(q_0, a) = \delta(q_f, a)$

a) Show that for all $w \neq \varepsilon$, we have $\hat{\delta}(q_0, w) = \hat{\delta}(q_f, w)$

b) Show that for all $x \in \mathcal{L}(A)$ with $x \neq \varepsilon$, we have $x^k \in \mathcal{L}(A)$ for all $k > 0$.

Non-determinism

- Deterministic F A : $\delta(q,a)$ is a unique state

  $\Rightarrow$ for each $w \in \Sigma^*$, the execution is completely determined

- Nondeterministic F.A. (NFA) : $\delta(q,a)$ is a set of states
  - may be the empty set
  - may contain several states

  $\Rightarrow$ multiple choices allow NFA to "guess" the right move.
  Accepts a string w if there is a sequence of guesses that leads to a final state.

Definition: an NFA is a quintuple $A_N = (Q, \Sigma, \delta_N, q_0, F)$

where : Q, Σ, q_0, F are as for a DFA

δ_N is a total function

$$\delta_N : Q \times \Sigma \to 2^Q$$

powerset of Q (i.e. the set of all subsets of Q)

i.e. $\delta(q,a)$ is a subset of Q

Note : $\delta(q,a)$ may be the empty set
i.e. the NFA makes no transition on that input

Definition: the extended transition function of an NFA A_N is the function $\hat{\delta}_N : Q \times \Sigma^* \to 2^Q$ defined as follows :

$\forall q \in Q, \ \forall a \in \Sigma, \ \forall x \in \Sigma^*$

- $\hat{\delta}_N(q, \varepsilon) = \{q\}$
- $\hat{\delta}_N(q, x \cdot a) = \bigcup_{p \in \hat{\delta}_N(q,x)} \delta_N(p, a)$

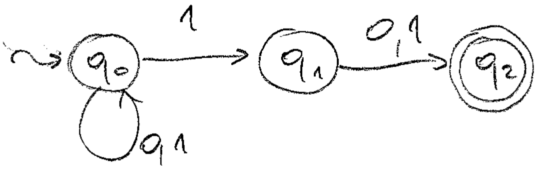
$\boxed{\text{i.e. if } \hat{\delta}_N(q,x) = \{p_1, \dots, p_k\} \text{ and } \delta_N(p_i, a) = S_i \text{ for } i \in \{1, \dots, k\} \text{ then } \hat{\delta}_N(q, xa) = S_1 \cup \dots \cup S_k}$

Definition: the language accepted by an NFA A_N is

$$\mathcal{L}(A_N) = \{ w \in \Sigma^* \mid \hat{\delta}_N(q_0, w) \cap F \neq \emptyset \}$$

Example: $L_{ex} = \{ w \mid w\text{'s one but last symbol is } 1 \}$

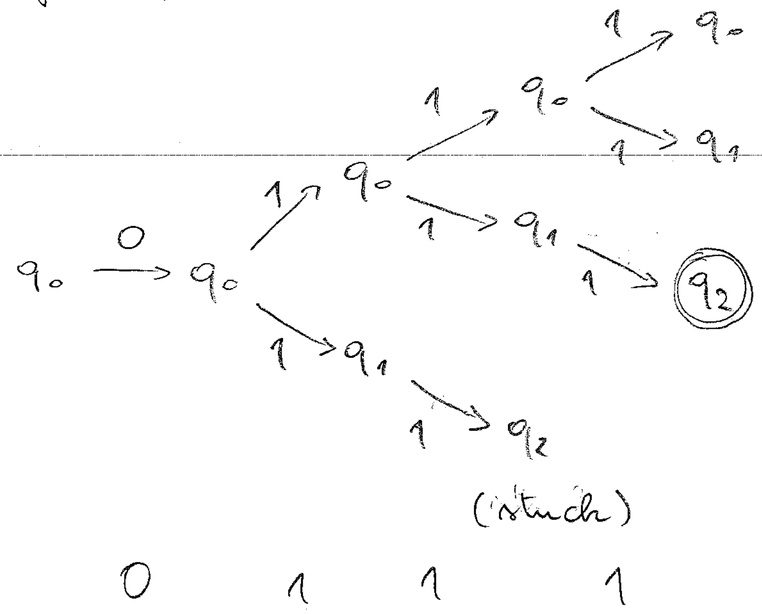
Idea: NFA "guesses" the end of input using nondeterminism and looks for 10 or 11



(note: transitions from q_2 are all to $\emptyset$)

Given an input string w, we can represent the computation of A_N on w as a tree of possible executions (instead of a trace in a state-space)

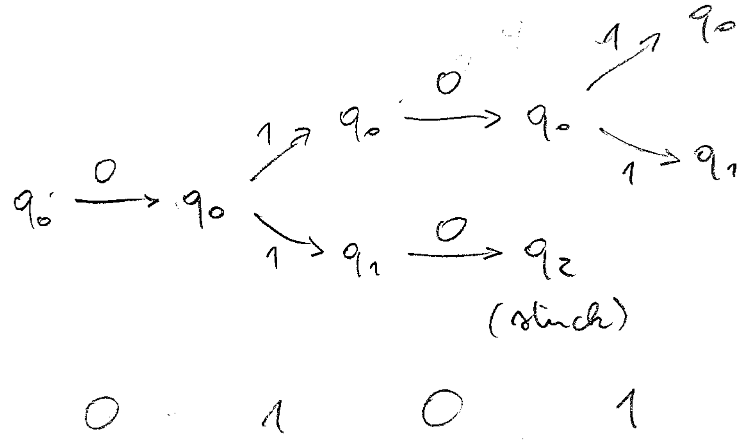
e.g. for input 0111



The string 0111 is accepted, because $\hat{\delta}_N(q_0, 0111)$ contains at least one final state.

I.e., there is at least one execution path that ends in a final state

for input 0101



The string 0101 is not accepted.

All execution paths either get stuck, or end in a non-final state

Different views of non-determinism:

1) The NFA always makes the right choices to ensure acceptance (if possible at all)

2) The NFA spawns off multiple copies at each non-deterministic choice point

3) The NFA explores multiple paths in parallel

Note: The various paths / computations evolve completely independently from each other

(different e.g. from parallel computations which may synchronize at a certain point)

Exercise E2.1 Give NFA's for the languages in Exercise 2.2.5

6/10/2008

Relationship between DFAs and NFAs

Let $\mathcal{L}$(DFA) be the class of languages accepted by some DFA.
— " — $\mathcal{L}$(NFA) — " — NFA

What is the relationship between $\mathcal{L}$(DFA) and $\mathcal{L}$(NFA)?

We show now that $\mathcal{L}$(DFA) = $\mathcal{L}$(NFA), i.e. DFA's and NFAs have the same expressive power.

We show the two directions separately.

<u>Theorem</u>: $\mathcal{L}(DFA) \subseteq \mathcal{L}(NFA)$

i.e., for every DFA A_D there is an NFA A_N such that
$$\mathcal{L}(A_N) = \mathcal{L}(A_D)$$

<u>Proof</u>: Easy. Let $A_D = (Q, \Sigma, \delta_D, q_0, F)$ be a DFA.

We define an NFA $A_N = (Q, \Sigma, \delta_N, q_0, F)$, with δ_N defined by the rule:

if $\delta_D(q, a) = p$ then $\delta_N(q, a) = \{p\}$

(Intuitively: we view the DFA as an NFA)

We can show by induction on $|w|$ that if $\hat{\delta}_D(q_0, w) = p$ then $\hat{\delta}_N(q_0, w) = \{p\}$. $\boxed{\text{Exercise 2.3.5 (optional)}}$

Since A_D and A_N coincide in the initial and final states, we get that $\mathcal{L}(A_D) = \mathcal{L}(A_N)$. q.e.d.

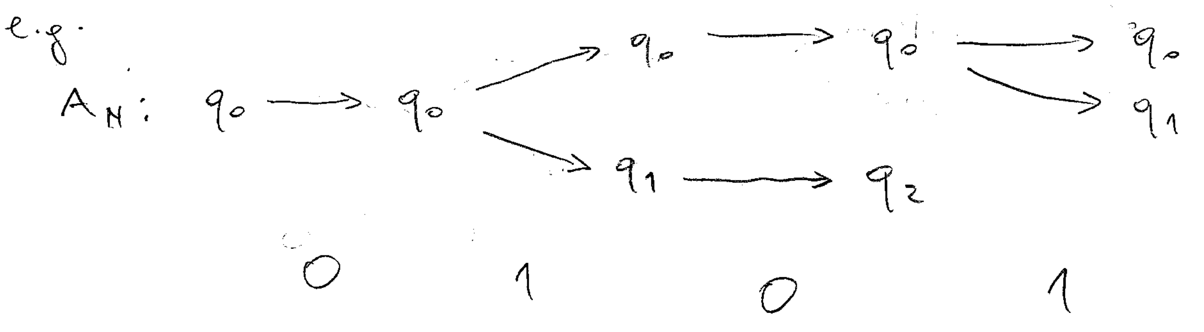
<u>Theorem</u>: $\mathcal{L}(NFA) \subseteq \mathcal{L}(DFA)$

i.e. for every NFA A_N there is a DFA A_D such that
$$\mathcal{L}(A_D) = \mathcal{L}(A_N)$$

Idea for the construction of A_D:

A_D simulates the entire execution tree of A_N in one exec.

e.g.



A_D: $\{q_0\} \longrightarrow \{q_0\} \longrightarrow \{q_0, q_1\} \longrightarrow \{q_0, q_2\} \longrightarrow \{q_0, q_1\}$

$\Rightarrow$ a state in A_D corresponds to a subset of A_N's states.

Subset construction:

given $A_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$

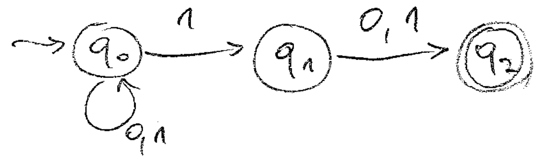
define $A_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ with

- $Q_D = 2^{Q_N}$
- $F_D = \{ S \subseteq Q_N \mid S \cap F_N \neq \emptyset \}$
- $\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$

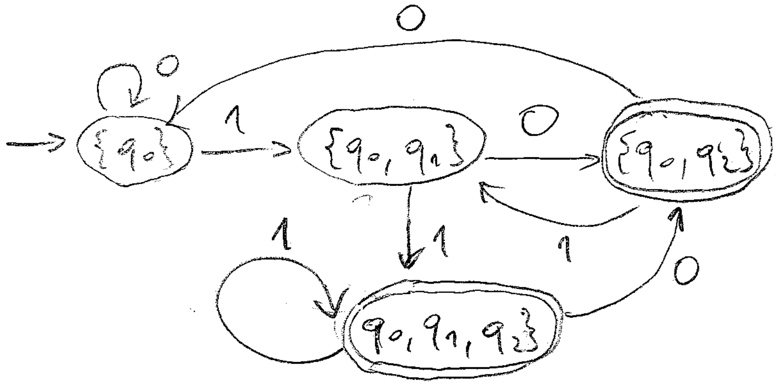
  i.e. $\delta_D(S, a)$ is the set of states of A_N reachable in A_N via a from some state in S.

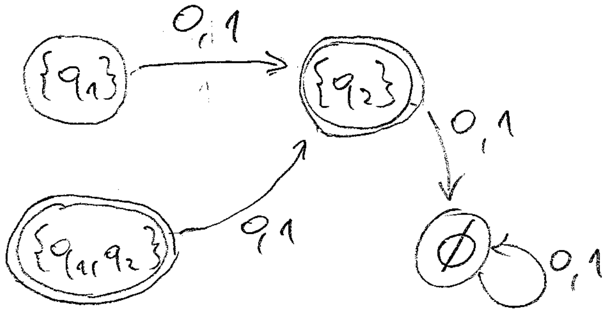
Example:

A_N :



A_D





$\emptyset$ is a dead state: we cannot leave it (the computation is stuck)

Note: Some states cannot be reached from the start state $\Rightarrow$ can be eliminated

We still have to show that for the DFA A_D constructed from A_N via the subset construction, we have $\mathcal{L}(A_D) = \mathcal{L}(A_N)$

<u>Optional part</u>

<u>Lemma</u>: $\forall q \in Q_N$, $\forall w \in \Sigma^*$

$$\hat{\delta}_D(\{q\}, w) = \hat{\delta}_N(q, w)$$

Proof: by induction on $|w|$

- basis: $|w| = 0$, i.e. $w = \varepsilon$

$$\hat{\delta}_D(\{q\}, \varepsilon) = \{q\} = \hat{\delta}_N(q, \varepsilon)$$

[def. of $\hat{\delta}_D$, base case] [def. of $\hat{\delta}_N$, base case]

- induction: assume claim holds for $|w| = n$, show for $|w| = n+1$

Let $w = x \cdot a$, with $|x| = n$, $|w| = n+1$

By inductive hyp. we have $\hat{\delta}_D(\{q\}, x) = \hat{\delta}_N(q, x)$

$\hat{\delta}_D(\{q\}, w) =$ [$w = x \cdot a$]

$= \hat{\delta}_D(\{q\}, x \cdot a) =$ [def. of $\hat{\delta}_D$]

$= \delta_D(\hat{\delta}_D(\{q\}, x), a) =$ [I.H.]

$= \delta_D(\hat{\delta}_N(q, x), a) =$ [def. of δ_D]

$= \bigcup_{p \in \hat{\delta}_N(q, x)} \delta_N(p, a) =$ [def. of $\hat{\delta}_N$]

$= \hat{\delta}_N(q, x \cdot a) =$ [$w = x \cdot a$]

$= \hat{\delta}_N(q, w)$

We can finish now the proof that $\mathcal{L}(A_D) = \mathcal{L}(A_N)$ (2.13)

$\mathcal{L}(A_D) = \{w \in \Sigma^* \mid \hat{\delta}_D(\{q_0\}, w) \in F_D\} =$ [def. of F_D]

$= \{w \in \Sigma^* \mid \hat{\delta}_D(\{q_0\}, w) \cap F_N \neq \emptyset\} =$ [Lemma]

$= \{w \in \Sigma^* \mid \hat{\delta}_N(q_0, w) \cap F_N \neq \emptyset\} =$ [def. of $\mathcal{L}(A_N)$]

$= \mathcal{L}(A_N)$

q.e.d.

↑ End of optional part

Note: the DFA A_D obtained from an NFA A_N has in general a number of states that is exponential in the number of states of A_N.

Can we do better? NO!

There are languages accepted by an NFA of n states, and for which the minimum size DFA has $O(2^n)$ states

Exercise E2.2 For $k \geq 1$, define an NFA A_N^k such that

$\mathcal{L}(A_N^k) = \{w \in \{0,1\}^* \mid$ the k-th last symbol of w is a $1\}$

Try to construct a DFA A_D^k s.t. $\mathcal{L}(A_D^k) = \mathcal{L}(A_N^k)$ by applying the subset construction

What are the numbers of states of A_N^k and A_D^k?

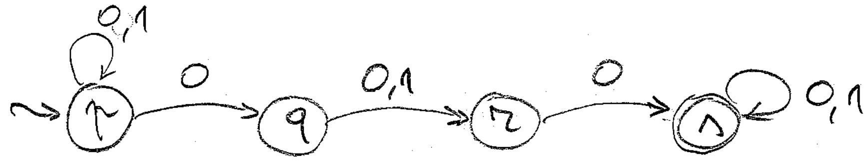
Exercise E2.3 For $\Sigma_k = \{a_1, \ldots, a_k\}$ construct an NFA A_N^k such that

$\mathcal{L}(A_N^k) = \{w \in \Sigma_k^* \mid w$ does not contain at least one of the symbols $a_1, \ldots, a_k\}$

Try to construct an equivalent DFA A_D^k

What are the numbers of states of A_N^k and A_D^k?

$\boxed{\text{Exercise 2.3.1}}$ Convert the following NFA to a DFA



$\boxed{\text{Exercise 2.3.4}}$ Give NFA's that accept the following languages

a) The set of strings over $\{0, \dots, 5\}$ s.t. the final digit has appeared before

b) The set of strings over $\{0, \dots, 5\}$ s.t. the final digit has not appeared before

Finite automata with ε-transitions

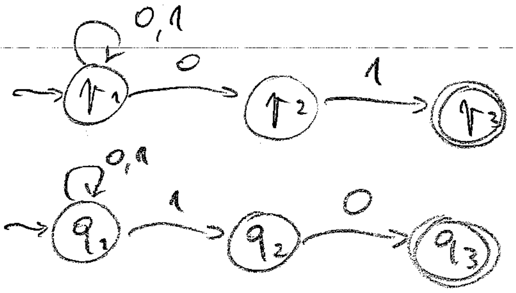
We add to NFA's ε-moves

meaning: the automaton can do a transition without consuming an input symbol

ε-NFA is as an NFA, but allowing also ε-moves

Example:

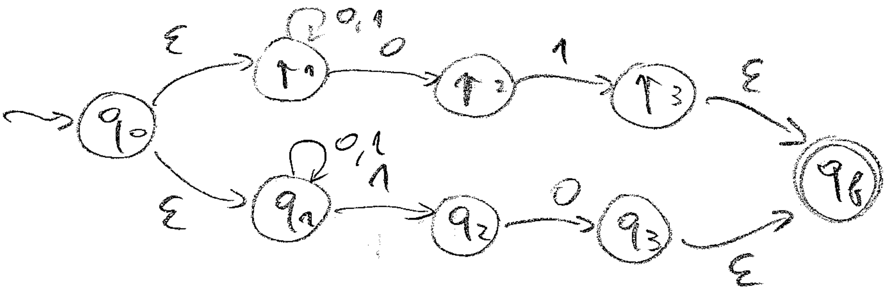
A_{01} / A_{10}



strings that end in 01

— " — 10

We want an automaton accepting all strings that end either in 01 or in 10



Note: ε-moves are another form of non-determinism: the automaton can non-deterministically choose to change state

Why are they useful?

- useful descriptive tool (for specifications), to take into account "external" events
- useful for composing NFA's
- conversion to DFA's is still possible

Definition: An ε-NFA is a quintuple $A_\varepsilon = (Q, \Sigma, \delta, q_0, F)$

where Q, Σ, q_0, F are as for an NFA

and $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow 2^Q$

e.g. we may have : $\delta(q_1, \varepsilon) = \{q_2, q_3, q_4\}$

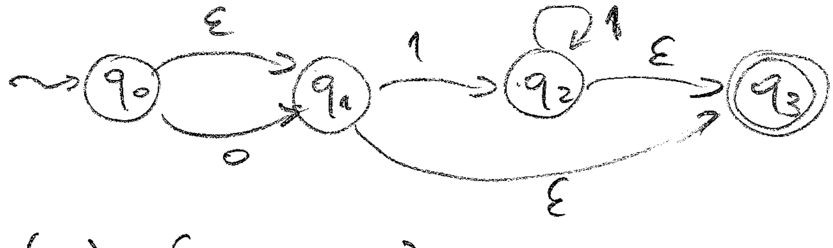
ε-closure: for $q \in Q$, $Eclose(q)$ is the set of all states reachable from q using a sequence of ε-moves (including the empty sequence).

Can be defined inductively:

- $q \in Eclose(q)$
- if $p \in Eclose(q)$ and $p' \in \delta(p, \varepsilon)$, then $p' \in Eclose(q)$
- nothing else is in $Eclose(q)$

Note: always $q \in Eclose(q)$

Example:



$Eclose(q_0) = \{q_0, q_1, q_3\}$ $\qquad$ $Eclose(q_1) = \{q_1, q_3\}$

We can extend Eclose to sets of states: $Eclose(S) = \bigcup_{p_i \in S} Eclose(p_i)$

To define $\hat{\delta}$, we have to take into account Eclose:

- basis: $\hat{\delta}(q, \varepsilon) = Eclose(q)$
- induction: $\hat{\delta}(q, x \cdot a) = Eclose\left(\bigcup_{p_i \in \hat{\delta}(q, x)} \delta(p_i, a)\right) =$

$$= \bigcup_{p_i \in \hat{\delta}(q, x)} Eclose(\delta(p_i, a))$$

In more detail:

- let $\hat{\delta}(q, x) = \{p_1, \dots, p_k\}$
- let $\bigcup_{p_i \in \hat{\delta}(q,x)} \delta(p_i, e) = \delta(p_1, e) \cup \dots \cup \delta(p_k, e) =$

$$= \{r_1, \dots, r_m\}$$

then $\hat{\delta}(q, x \cdot e) = \text{Eclose}(\{r_1, \dots, r_m\})$

In other words: $\hat{\delta}(q, w)$ is the set of all states reachable from q along paths whose labels on arcs, apart from ε, yield w

Note:
- $q \in \hat{\delta}(q, \varepsilon)$
- $\delta(q, e) \neq \hat{\delta}(q, e)$ (different from DFA/NFA)

In fact $\hat{\delta}(q, e) = \text{Eclose}\left(\bigcup_{p_i \in \text{Eclose}(q)} \delta(p_i, e)\right)$

Example (previous ε-NFA)

$\hat{\delta}(q_0, \varepsilon) = \{q_0, q_1, q_3\}$ $\qquad$ $\delta(q_0, \varepsilon) = \{q_1\}$

$\hat{\delta}(q_0, 1) = \text{Eclose}\left(\bigcup_{p_i \in \{q_0, q_1, q_3\}} \delta(p_i, 1)\right) = \text{Eclose}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$
$= \text{Eclose}(\emptyset \cup \{q_2\} \cup \{q_2\}) = \{q_2, q_3\}$

Definition: language accepted by an ε-NFA A_ε

$$\mathcal{L}(A_\varepsilon) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

13/11/2008

Theorem: For each ε-NFA A_ε there exists an NFA A_N such that $\mathcal{L}(A_\varepsilon) = \mathcal{L}(A_N)$

Idea: equivalent NFA has (almost) the same Q, q_0, and F. Only δ_N is changed by removing ε-moves and adding new moves instead

Formally: Let $A_\varepsilon = (Q, \Sigma, \delta_\varepsilon, q_0, F)$ be an ε-NFA.

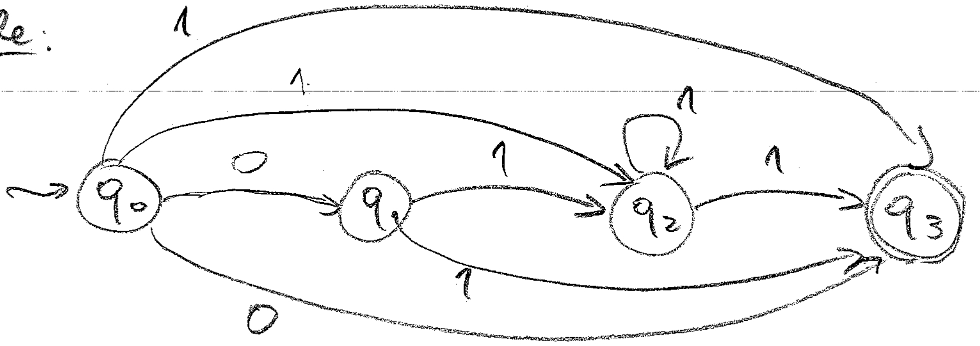
We construct the NFA $A_N = (Q, \Sigma, \delta_N, q_0, F)$ with

$\forall q \in Q, \ \forall a \in \Sigma$

$$\delta_N(q, a) = \hat{\delta}_\varepsilon(q, a) = \text{Eclose}\left(\bigcup_{p_i \in \text{Eclose}(q)} \delta_\varepsilon(p_i, a)\right)$$

Note: $\delta_N(q, \varepsilon)$ is not defined (and it should not be)

Example:



Question: Do we have that $\mathcal{L}(A_N) = \mathcal{L}(A_\varepsilon)$?

Yes, except possibly for ε.

In A_ε, we have that $\varepsilon \in \mathcal{L}(A_\varepsilon)$ if $\text{Eclose}(q_0) \cap F \neq \emptyset$

In A_N — " — if $q_0 \in F$

We have to adjust for that:

make q_0 a final state of A_N, if in A_ε $\text{Eclose}(q_0) \cap F \neq \emptyset$

$\boxed{\text{Exercise E2.4}}$ Prove that $\mathcal{L}(A_N) = \mathcal{L}(A_\varepsilon)$

Note: Combining the elimination of ε-transition with the subset construction, we can convert an ε-NFA to a DFA.

(Textbook provides a direct construction)