Finite Automata

SITE : http://www.info.univ-tours.fr/˜mirian/

Finite State Automata (FSA)

- Deterministic

  On each input there is one and only one state to which the automaton can transition from its current state

- Nondeterministic

  An automaton can be in several states at once

Deterministic finite state automaton

1. A finite set of **states**, often denoted Q
2. A finite set of **input symbols**, often denoted Σ
3. A **transition function** that takes as arguments a state and an input symbol and returns a state.
   The transition function is commonly denoted δ
   If q is a state and a is a symbol, then $\delta(q, a)$ is a state p (and in the graph that represents the automaton there is an arc from q to p labeled a)
4. A **start state**, one of the states in Q
5. A set of **final or accepting** states F ($F \subseteq Q$)

Notation: A DFA A is a tuple

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

Other notations for DFAs

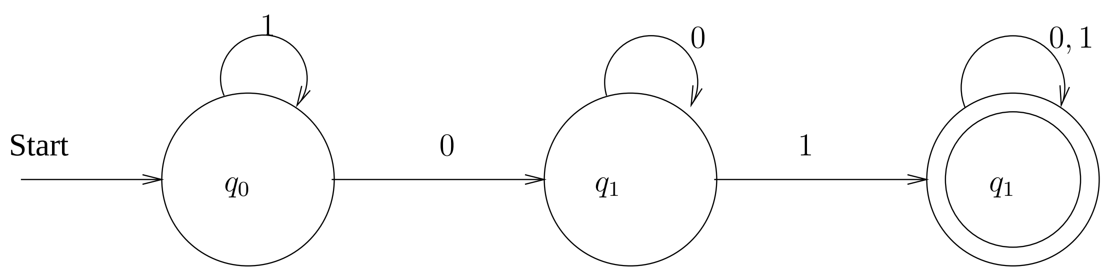
1. Transition diagrams
   - Each state is a node
   - For each state $q \in Q$ and each symbol $a \in \Sigma$, let $\delta(q, a) = p$
     Then the transition diagram has an arc from q to p, labeled a
   - There is an arrow to the start state q_0
   - Nodes corresponding to final states are marked with doubled circle
2. Transition tables
   - Tabular representation of a function
   - The rows correspond to the states and the columns to the inputs
   - The entry for the row corresponding to state q and the column corresponding to input a is the state $\delta(q, a)$

Example: An DFA accepting strings with a substring 01

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

where the transition function δ is given by the table

		0	1
$\rightarrow$	q_0	q_2	q_0
$\star$	q_1	q_1	q_1
	q_2	q_2	q_1



Extending the Transaction Function to Strings

- The DFA define a language: the set of all strings that result in a sequence of state transitions from the start state to an accepting state
- *Extended transition function*
  - Describes what happens when we start in any state and follow any sequence of inputs
  - If δ is our transition function, then the extended transition function is denoted by $\hat{\delta}$
  - The extended transition function is a function that takes a state q and a string w and returns a state p (the state that the automaton reaches when starting in state q and processing the sequence of inputs w)

Formal definition of the extended transition function

Definition by induction on the length of the input string

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

If we are in a state q and read no inputs, then we are still in state q

Induction: Suppose w is a string of the form xa; that is a is the last symbol of w, and x is the string consisting of all but the last symbol

Then: $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$

- To compute $\hat{\delta}(q, w)$, first compute $\hat{\delta}(q, x)$, the state that the automaton is in after processing all but the last symbol of w
- Suppose this state is p, *i.e.*, $\hat{\delta}(q, x) = p$
- Then $\hat{\delta}(q, w)$ is what we get by making a transition from state p on input a - the last symbol of w

Example

Design a DFA to accept the language

$$L = \{w \mid w \text{ has both an even number of } 0 \text{ and an even number of } 1\}$$

The Language of a DFA

The language of a DFA $A = (Q, \Sigma, \delta, q_0, F)$, denoted $L(A)$ is defined by

$$L(A) = \{w \mid \hat{\delta}(q_0, w) \text{ is in } F\}$$

The language of A is the set of strings w that take the start state q_0 to one of the accepting states

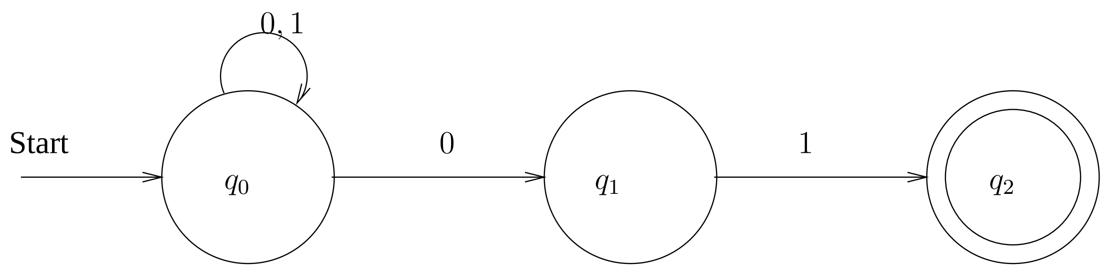
If L is a $L(A)$ from some DFA, then L is a *regular language*

Nondeterministic Finite Automata (NFA)

- A NFA has the power to be in several states at once
- This ability is often expressed as an ability to “guess” something about its input
- Each NFA accepts a language that is also accepted by some DFA
- NFA are often more succinct and easier than DFAs
- We can always convert an NFA to a DFA, but the latter may have exponentially more states than the NFA (a rare case)
- The difference between the DFA and the NFA is the type of transition function δ
    - For a NFA δ is a function that takes a state and input symbol as arguments (like the DFA transition function), but returns a set of zero or more states (rather than returning exactly one state, as the DFA must)

Example: An NFA accepting strings that end in 01

Nondeterministic automaton that accepts all and only the strings of 0s and 1s that end in 01



NFA: Formal definition

A nondeterministic finite automaton (NFA) is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where:

1. Q is a finite set of *states*
2. Σ is a finite set of *input symbols*
3. $q_0 \in Q$ is the *start state*
4. F ($F \subseteq Q$) is the set of *final or accepting* states
5. δ, the *transition function* is a function that takes a state in Q and an input symbol in Δ as arguments and returns a subset of Q

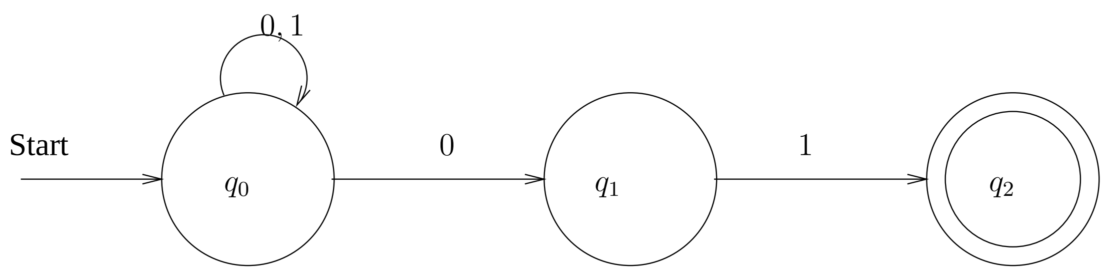
The only difference between a NFA and a DFA is in the type of value that δ returns

Example: An NFA accepting strings that end in 01

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

where the transition function δ is given by the table

		0	1
$\rightarrow$	q_0	$\{q_0, q_1\}$	$\{q_0\}$
	q_1	$\emptyset$	$\{q_2\}$
$\star$	q_2	$\emptyset$	$\emptyset$



The Extended Transition Function

Basis: $\hat{\delta}(q, \epsilon) = \{q\}$
Without reading any input symbols, we are only in the state we began in

Induction:

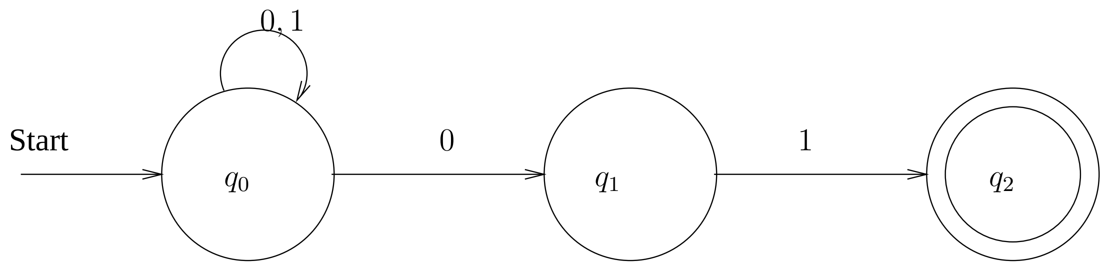
- Suppose w is a string of the form xa; that is a is the last symbol of w, and x is the string consisting of all but the last symbol
- Also suppose that $\hat{\delta}(q, x) = \{p_1, p_2, \ldots p_k, \}$
- Let

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$$

Then: $\hat{\delta}(q, w) = \{r_1, r_2, \ldots, r_m\}$

We compute $\hat{\delta}(q, w)$ by first computing $\hat{\delta}(q, x)$ and by then following any transition from any of these states that is labeled a

Example: An NFA accepting strings that end in 01



Processing $w = 00101$

1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
3. $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
4. $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
5. $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
6. $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

The Language of a NFA

The language of a NFA $A = (Q, \Sigma, \delta, q_0, F)$, denoted $L(A)$ is defined by

$$L(A) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

The language of A is the set of strings $w \in \Sigma^*$ such that $\hat{\delta}(q_0, w)$ contains at least one accepting state

The fact that choosing using the input symbols of w lead to a non-accepting state, or do not lead to any state at all, does not prevent w from being accepted by a NFA as a whole.

Equivalence of Deterministic and Nondeterministic Finite Automata

- Every language that can be described by some NFA can also be described by some DFA.
- The DFA in practice has about as many states as the NFA, although it has more transitions.
- In the worst case, the smallest DFA can have 2^n (for a smallest NFA with n state).

Proof: DFA can do whatever NFA can do

The proof involves an important construction called subset construction because it involves constructing all subsets of the set of stages of NFA.

From NFA to DFA

- We have a NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$
- The goal is the construction of a DFA $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that $L(D) = L(N)$.

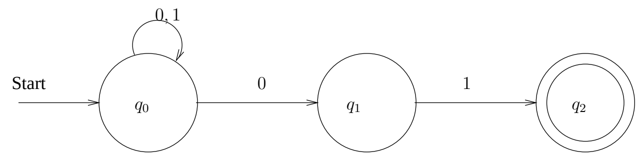
Subset Construction

- Input alphabets are the same.
- The start set in D is the set containing only the start state of N.
- Q_D is the set of subsets of Q_N, *i.e.*, Q_D is the power set of Q_N.
  If Q_N has n states Q_D will have 2^n states. Often, not all of these states are accessible from the start state.
- F_D is the set of subsets S of Q_N such that $S \cap F_N \neq \emptyset$. That is, F_D is all sets of N's states that include at least one accepting state of N.
- For each set $S \subseteq Q_N$ and for each input symbol $a \in \Sigma$

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$$

  To compute $\delta_D(S, a)$, we look at all the states p in S, see what states N goes from p on input a, and take the union of all those states.

Example



$Q_N = \{q_0, q_1, q_2\}$ then $Q_D = \{\emptyset, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\} \ldots\}$, *i.e.*, Q_D has 8 states (each one corresponding to a subset of Q_N)

		0	1
	$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow$	$\{q_0\}$	$\{q_0\ q_1\}$	$\{q_0\}$
	$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\star$	$\{q_2\}$	$\emptyset$	$\emptyset$
	$\{q_0, q_1\}$	$\{q_0\ q_1\}$	$\{q_0, q_2\}$
$\star$	$\{q_0, q_2\}$	$\{q_0\ q_1\}$	$\{q_0\}$
$\star$	$\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$\star$	$\{q_0, q_1, q_2\}$	$\{q_0\ q_1\}$	$\{q_0, q_2\}$

Example: with new names

Note: The states of D correspond to subsets of states of N , but we could have denoted the states of D by, say, A - F just as well.

		0	1
	A	A	A
$\rightarrow$	B	E	B
	C	A	D
$\star$	D	A	A
	E	E	F
$\star$	F	E	B
$\star$	G	A	D
$\star$	H	E	F

Avoiding exponential blow-up

We can often avoid the exponential blow-up by constructing the transition table for D only for accessible states S as follows:

Basis: $S = \{q_0\}$ is accessible in D

Induction: If state S is accessible, so are the states in a $\bigcup_{a \in \Sigma} \delta_D(S, a)$.

Theorem

If $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ is the DFA constructed from NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ by the subset construction, then $L(D) = L(N)$ $\square$

Proof: First we show on an induction on $|w|$ that

$$\hat{\delta}_D((\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

Basis: $w = \epsilon$. The claim follows from def.

Induction: $\hat{\delta}_D(\{q_0\}, x.a) = ...$

Theorem

A language L is accepted by some DFA if and only if L is accepted by some NFA.

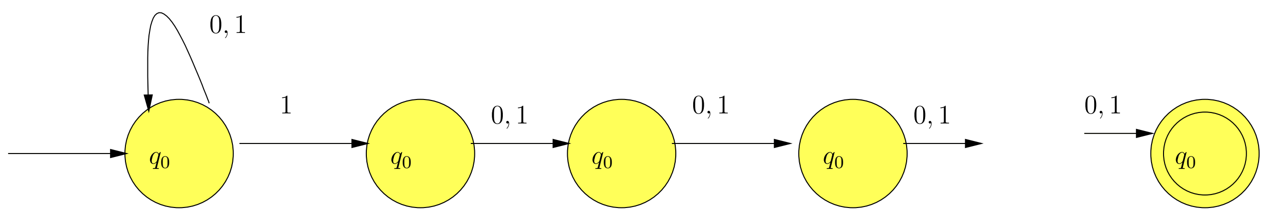
Proof: The "if" part is the theorem before

For the "only if" part we note that any DFA can be converted to an equivalent NFA by modifying the δ_D to δ_N by the rule

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

Exponential Blow-Up

There is an NFA N with $n+1$ states that has no equivalent DFA with fewer than 2^n states.



$L(N)$: the set of all strings of 0 and 1 such that the nth symbol **from the end** is 1.

$$L(N) = \{x1c_2c_3 \dots c_n \mid x \in \{0,1\}^*, c_i \in \{0,1\}\}$$

Suppose an equivalent DFA D with fewer than 2^n states exists.

D must remember the last n symbols it has read.

There are 2^n bitsequences $a_1a_2 \dots a_n$

$$\begin{aligned} \exists q, a_1a_2 \dots a_n, b_1b_2 \dots b_n \mid \quad & q \in \hat{\delta}_N(q0, a_1a_2 \dots a_n), \\ & q \in \hat{\delta}_N(q0, b_1b_2 \dots b_n), \\ & a_1a_2 \dots a_n \neq b_1b_2 \dots b_n \end{aligned}$$

Exponential Blow-Up

Since the sequences are different, they must differ in some position, say $a_i \neq b_i$

Case 1: $i = 1$

$1a_2 \ldots a_n$

$0b_2 \ldots b_n$

Then q has to be both an accepting and a non accepting state.

Case 2: $i > 1$

$a_1 \ldots a_{i-1}\mathbf{1}a_{i+1} \ldots a_n$

$b_1 \ldots b_{i-1}\mathbf{0}b_{i+1} \ldots b_n$

Now $\hat{\delta}_N(q_0, a_1 \ldots a_{i-1}1a_{i+1} \ldots a_n0^{i-1}) = \quad \hat{\delta}_N(q_0, b_1 \ldots b_{i-1}0b_{i+1} \ldots b_n0^{i-1})$

and

$\hat{\delta}_N(q_0, a_1a_{i-1}1a_{i+1}a_n0^{i-1}) \in F_D$

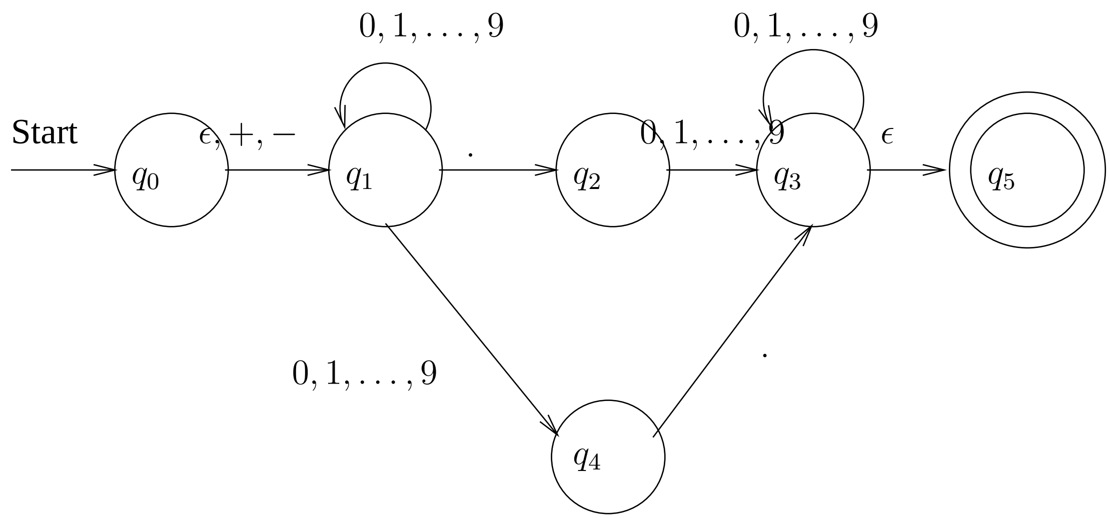
$\hat{\delta}_N(q_0, b_1b_{i-1}0b_{i+1}b_n0^{i-1}) \notin F_D$

Finite Automata with Epsilon-Transition

- We allow now a transition on ϵ, *i.e.*, the empty string
- A NFA is allowed to make a transition spontaneously, without receiving an input symbol
- This capability does not expand the class of languages that can be accepted by finite automata, but it does give us some added programming convenience

Example

NFA that accepts decimal numbers consisting of: an optional $+$ or $-$ sign; a string of digits; a decimal point and another string of digits. Either the first or the second string of bits can be empty, but at least one of the two strings must be nonempty.



Formal Notation

An ϵ-NFA $A = (Q, \Sigma, \delta, q_0, F)$ where all components have their same interpretation as for NFA, except that δ is now a function that takes arguments:

1. A state in Q and
2. A member of $\Sigma \cup \{\epsilon\}$
   We require that ϵ cannot be a member of Σ

Epsilon-Closures

We define ϵ-closure $ECLOSE(q)$ recursively, as follows:

Basis: State q is in $ECLOSE(q)$

Induction: If state p is in $ECLOSE(q)$, and there is a transition from p to r labeled ϵ, then r is in $ECLOSE(q)$

More precisely, if δ is the transition function of the ϵ-NFA involved, and p is in $ECLOSE(q)$, then $ECLOSE(q)$ also contains all the states in $\delta(p, \epsilon)$

Extended Transitions and Languages for ϵ-NFA

The definition of $\hat{\delta}$ is :

Basis $\hat{\delta}(q, \epsilon) = ECLOSE(q)$

If the label of the path is ϵ , then we can follow only ϵ-labeled arcs extending from state q

Induction Suppose w is of the form xa, where $a \in \Sigma$ is the last symbol of w. We compute $\hat{\delta}(q, w)$ as follows:

1. Let $\{p_1, p_2, \ldots p_k\}$ be $\hat{\delta}(q, x)$
   The p_i are all and only the states that we can reach from q following a path labeled x. This path may end with one or more ϵ-transitions and may have other ϵ-transitions.
2. Let $\bigcup_{i=1}^{k} \delta(p_i, a)$ be the set $\{r_1, r_2, \ldots, r_m\}$
   Follow all transitions labeled a from states we can reach from q along paths labeled x. The r_j are some of the states we can reach from q along paths labeled w. The additional states we can reach are found from the r_j by the following ϵ-labeled arcs in the step below
3. Then $\hat{\delta}(q, w) = \bigcup_{j=1}^{m} ECLOSE(r_j)$

The language of an ϵ-NFA

The language of an ϵ-NFA $E = (Q, \Sigma, \delta, q_0, F)$ is

$$L(E) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

Eliminating ϵ-Transitions

Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be an ϵ-NFA. Then the equivalent DFA $D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ is defined as follows:

1. Q_D is the set of subsets of Q_E
2. $q_D = ECLOSE(q_0)$
3. $F_D = \{S \mid S \text{ is in } Q_D \text{ and } S \cap F_E \neq \emptyset\}$
   Those sets of states that contain at least one accepting state of E
4. $\delta_D(S, a)$ is computed, for all a in Σ and sets S in Q_D by:
   - Let $S = \{p_1, p_2, \ldots, p_k\}$
   - Compute $\bigcup_{i=1}^{k} \delta_E(p_i, a)$; let this set be $\{r_1, r_2, \ldots, r_m\}$
   - Then $\delta_D(S, a) = \bigcup_{j=1}^{m} ECLOSE(r_j)$

Theorem

A language L is accepted by some ϵ-NFA iff L is accepted by some DFA.