Introduction to the Theory of Computation

XU Ming

School of Software Engineering, East China Normal University

February 25, 2025

OUTLINE

- Deterministic Finite Automata
- Nondeterministic Finite Automata

Deterministic Finite Automata

Definition of DFA

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

- Q is a finite set of states,
- Σ is a finite set of input symbols (i.e. an alphabet),
- δ is a transition function from $Q \times \Sigma$ to Q,
- $q_0 \in Q$ is a start state,
- $F \subseteq Q$ is a set of final or accepting states.

Example

Let $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, q_0 is the start state, and $F = \{q_1\}$. If transition function $\delta : Q \times \Sigma \to Q$ is defined by

$$\begin{aligned} \delta(q_0, 0) &= q_2, \quad \delta(q_1, 0) = q_1, \quad \delta(q_2, 0) = q_2, \\ \delta(q_0, 1) &= q_0, \quad \delta(q_1, 1) = q_1, \quad \delta(q_2, 1) = q_1, \end{aligned}$$

then

$$A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

is a DFA.

Intuitional Descriptions for DFA

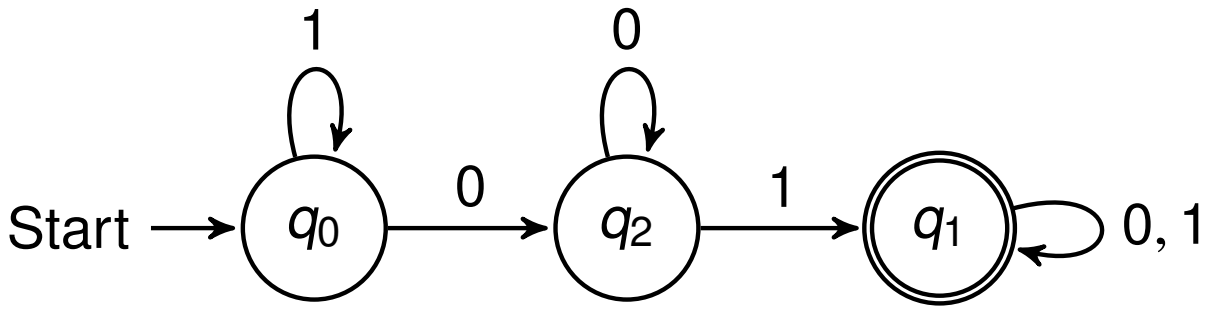
Specifying a DFA as a 5-tuple with a detailed description of the transition function δ is both tedious and hard to read. There are two preferrable descriptions:

Transition diagram a graph in which the nodes are the states, and arcs are labeled by input symbols, indicating the transitions of that automaton.

Transition table a tabular listing of the transition function δ, which by implication tells us the set of states and the input alphabet.

Transition Diagram

The automaton $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$ as a transition diagram



Transition Table

The automaton $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$ is represented as the transition table:

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

Extended Transition Function

The transition function δ can be extended to $\hat{\delta}$ that operates on states and strings (as opposed to states and symbols) by induction on the length of the input string:

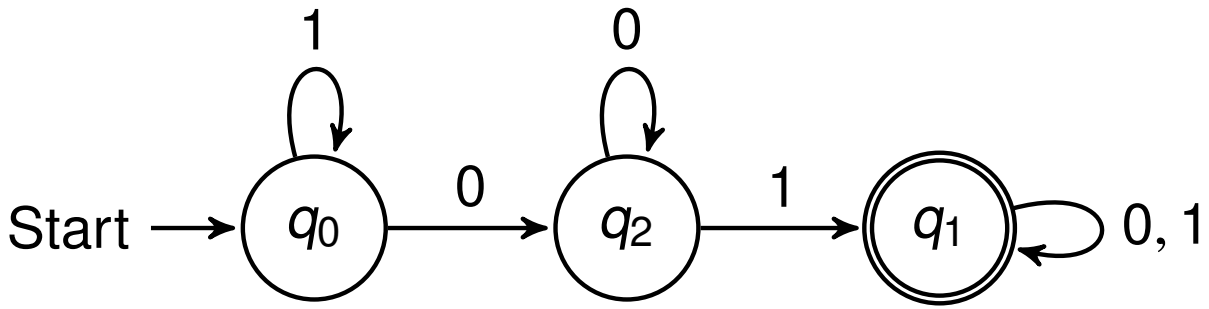
Basis step: $\hat{\delta}(q, \epsilon) = q$.

Inductive step: Suppose $w = xa$, then

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

Example

For $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$, we have $\hat{\delta}(q_0, 1101) = q_1$.

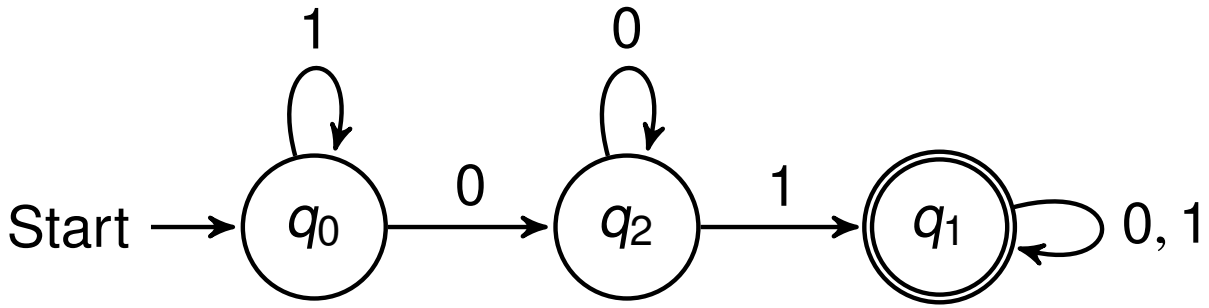


In details,

- $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1) = \delta(q_0, 1) = q_0$,
- $\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_0, 1) = q_0$,
- $\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$,
- $\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_1$.

Example

For $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$, we have $\hat{\delta}(q_0, 1101) = q_1$.



In details,

- $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1) = \delta(q_0, 1) = q_0$,
- $\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_0, 1) = q_0$,
- $\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$,
- $\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_1$.

Obviously, for any state q, and input symbol a,

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

Furthermore, for any strings x and y, one can prove:

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

and further

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

Do them!

Language of DFA

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

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

Note that we require that δ, and consequently $\hat{\delta}$, be *total functions* (at each step, there is a unique move is defined, so that we have justified in calling such an automaton deterministic).

A DFA will process *every* string in Σ^*, which will be either accepted or not.

If L is a language for some DFA A, we say L is a regular language.

To show any language is regular, all we have to do is finding a DFA for it.

Example

For $\Sigma = \{0, 1\}$, let L be the set consisting of all strings with at least one 01. Show L is regular.

Proof The construction of the DFA for this language is not difficult. The solution is just $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$.

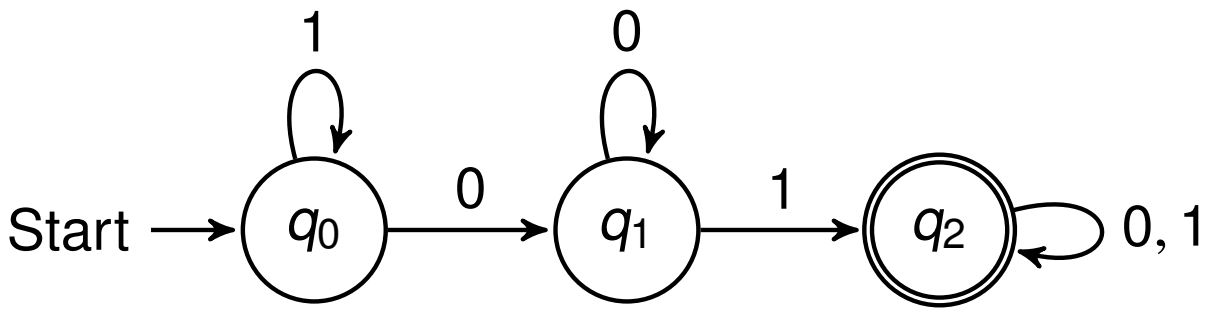
If L is a language for some DFA A, we say L is a regular language.

To show any language is regular, all we have to do is finding a DFA for it.

Example

For $\Sigma = \{0, 1\}$, let L be the set consisting of all strings with at least one 01. Show L is regular.

Proof The construction of the DFA for this language is not difficult. The solution is just $A_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$.



Languages defined by other condition

One may ask what if the acceptance condition

$$\{w \mid \hat{\delta}(q_0, w) \in F\}$$

is changed to:

$$\{w \mid w = uv \wedge \hat{\delta}(q_0, u) \in F\},$$

which means we accept this string w whenever some intermediate state is in F.

We can rephrase it as making all states in F absorbing, since the successive states are irrevalent to accept or reject in the setting.

Languages defined by other condition

DFAs with final states being absorbing are still DFAs, which has the restriction

$$\delta(q, a) = q \quad \text{for each } q \in F \text{ and } a \in \Sigma,$$

so the new languages in the consideration are still in the scope of regular languages.

A more interesting question is whether every regular language over alphabet Σ can be accepted by such a special DFA.

Unfortuniately, the answer is negative.

Languages defined by other condition

But we could partially fill the the gap by: Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$, there is a DFA $B = (Q_B, \Sigma_B, \delta_B, q_0, F_B)$ where

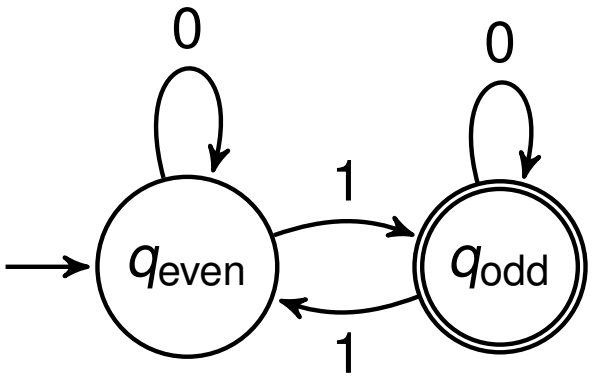
- $Q_B = Q \cup \{good, bad\}$,
- $\Sigma_B = \Sigma \cup \{\#\}$,
- $$\delta_B(q,a) = \begin{cases} \delta(q,a) & \text{if } q \in Q \text{ and } a \in \Sigma, \\ good & \text{if } q \in F \text{ and } a = \#, \\ bad & \text{if } q \in Q \setminus F \text{ and } a = \#, \\ good/bad & \text{if } q = good/bad. \end{cases}$$
- $F = \{good\}$,

so that

$$L(B) = \{u\#v \mid u \in L(A) \wedge v \in \Sigma_B^*\}.$$

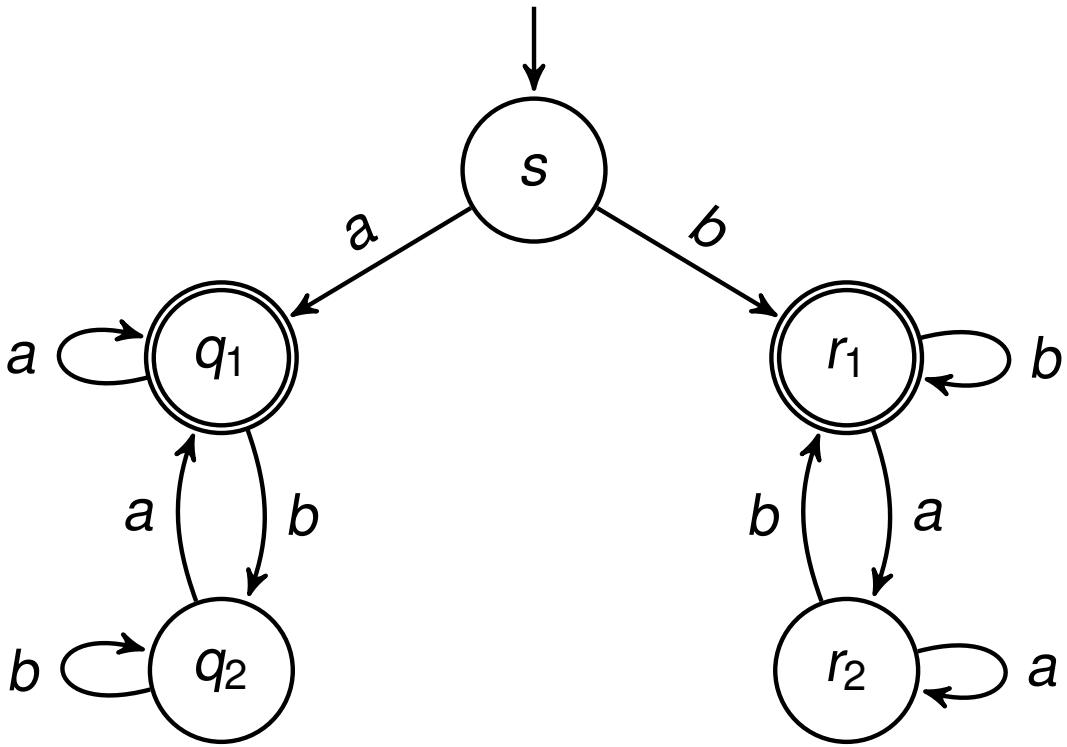
Designing DFA

Whether it is of automaton or artwork, design is a creative process. However, you might find a particular approach helpful when designing various types of automata. For example, suppose $\Sigma = \{0, 1\}$, design a DFA to accept all strings with an odd number of 1's.



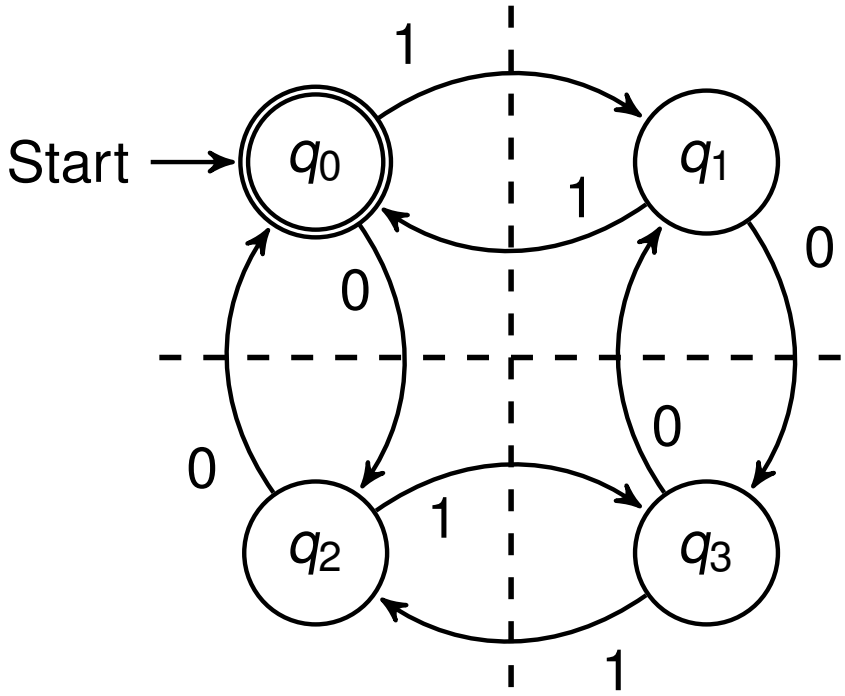
Example

Suppose $\Sigma = \{a, b\}$, design a DFA to accept all strings that start and end with a, or start and end with b.



Example

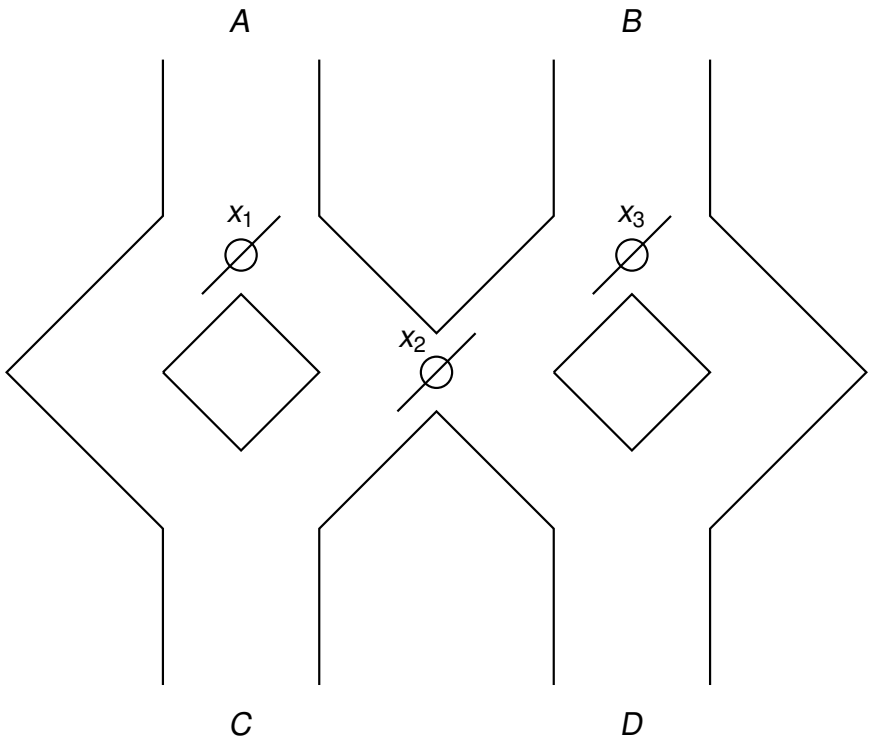
Suppose $\Sigma = \{0, 1\}$, design a DFA to accept all and only strings with an even number of 0's and an even number of 1's.



Question How to design a DFA in order to accept all and only with an even number of 0's and a k-multiple number of 1's ($k = 2, 3, \ldots$).

Example

Marble-rolling toy. A marble is dropped at A and B. Levers x_1, x_2 and x_3 cause the marble to fall either to the left or to the right. Whenever a marble encounters a lever, it causes the lever to reverse after the marble passes, so the next marble will take the opposite branch.

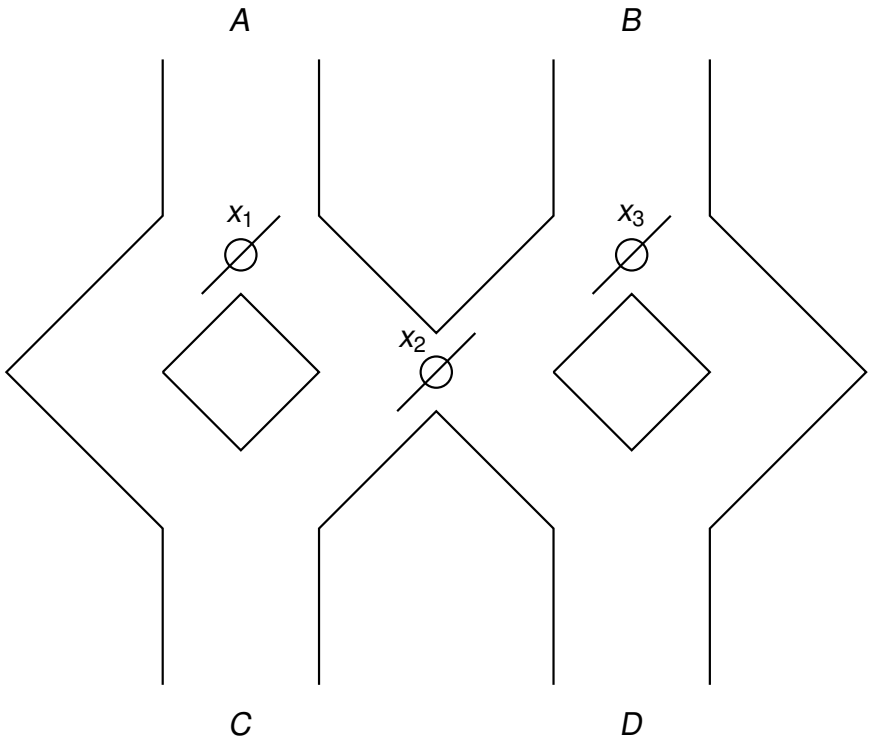


Model this toy by a DFA.

Let the inputs *A* and *B* represent the input into which the marble is dropped. Let acceptance correspond to the marble exiting at *D*; nonacceptance represents a marble exiting at *C*.

A state is represented as sequence of three bits followed by *r* or *a* (previous input *rejected* or *accepted*). For instance, 010*a* means *left*, *right*, *left*, *accepted*.

Tabular representation of DFA for the toy is given next slide.



	A	B
→ 000?	100r	011r
001?	101r	000a
010?	110r	001a
011?	111r	010a
100?	010r	111r
101?	011r	100a
110?	000a	101a
111?	001a	110a

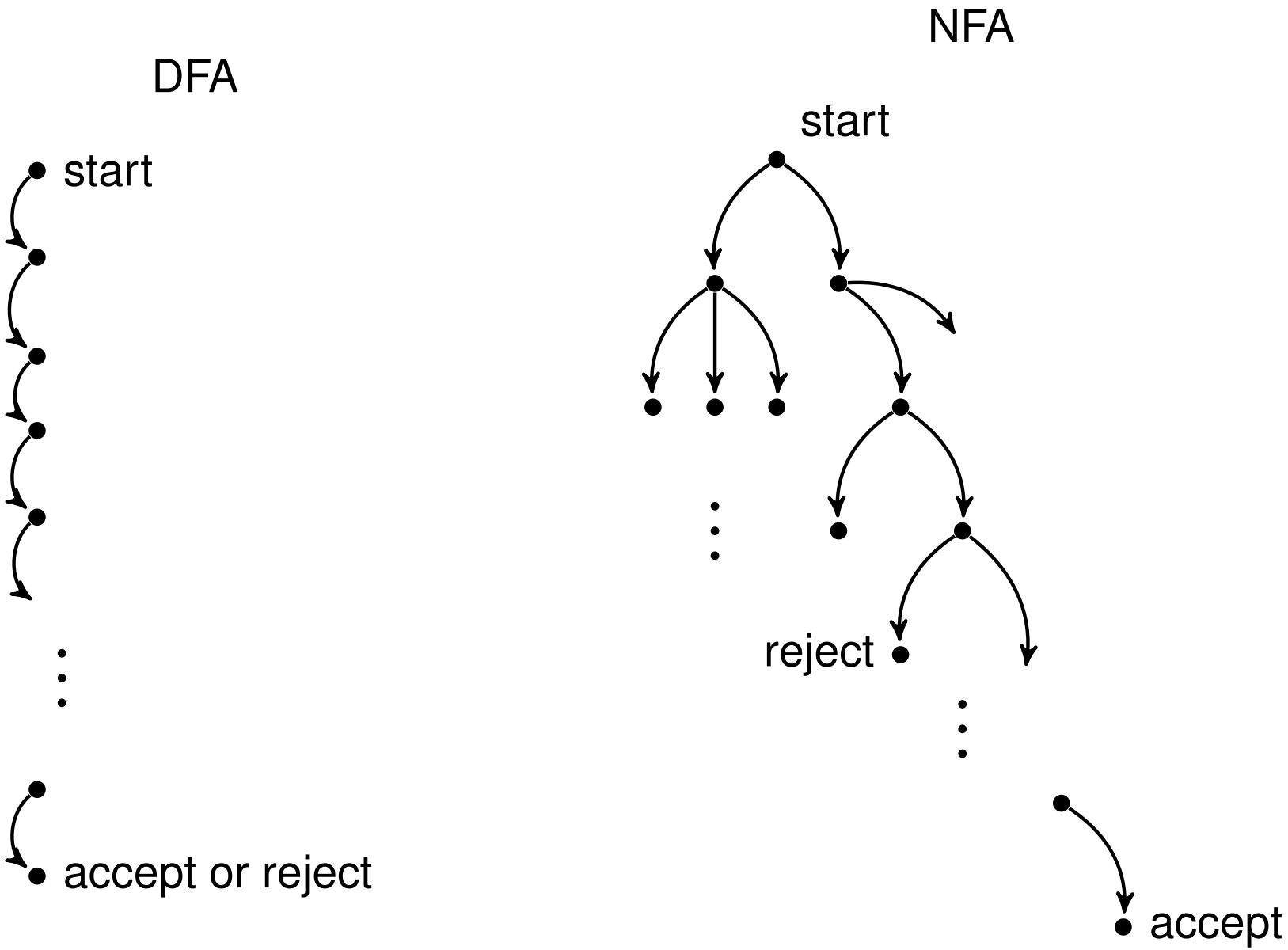
Question Informally describe the language of the automaton.

Nondeterministic Finite Automata

An Informal View of NFA

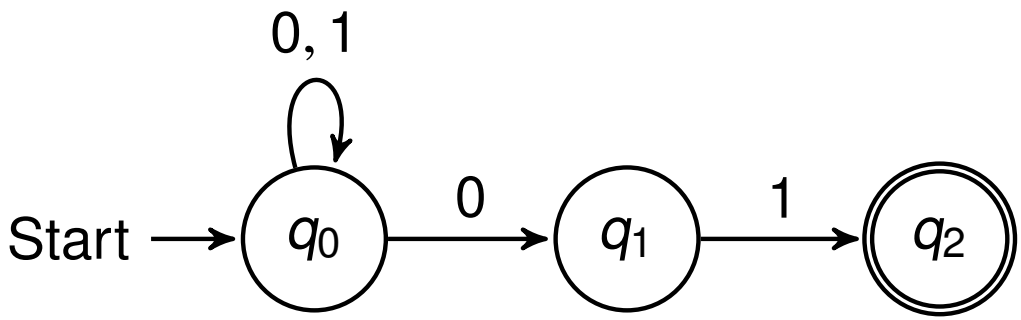
The difference between the DFA and the NFA is in the type of transition function. For the NFA, transition function takes a state and an input symbol as arguments, but returns a set of zero, one, or more states.

That means, an NFA can be in several states at once, or, viewed another way, it can "guess" which state to go to next.


DFA
start
accept or reject
NFA
start
reject
accept

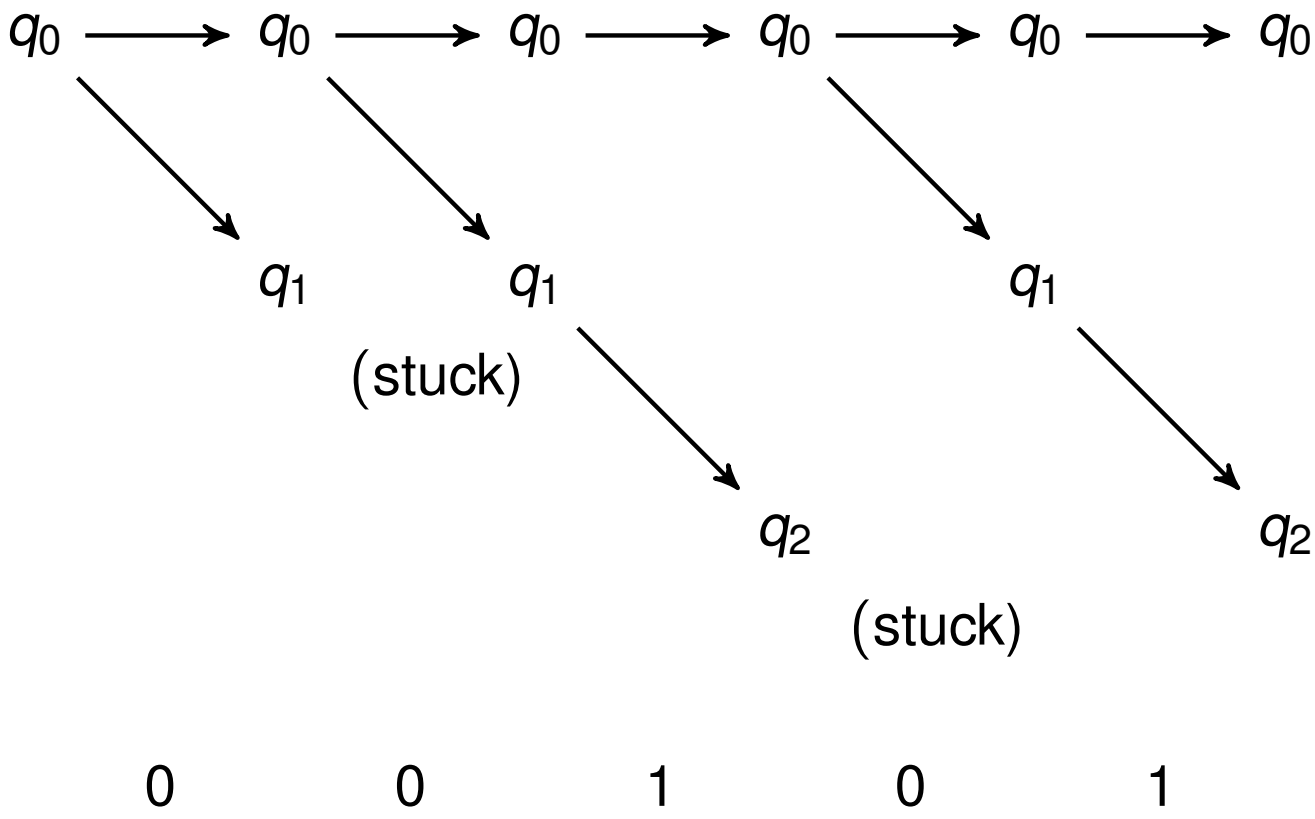
Example

An NFA that accepts all and only strings ending in 01.



Initially, the automaton is in q_0. The next symbol may not be the first one of the desired string 01, even if that symbol is 0. Thus, q_0 has a transition to itself on both 0 and 1. However, if the next symbol is 0, it guesses that final 01 has begun. It has the option of going either to q_0 or to q_1, and in fact it does both. In q_1, it checks that the next symbol is 1, and if so, it goes to q_2 and accepts.

Here is what happens when the NFA processes the input 00101.



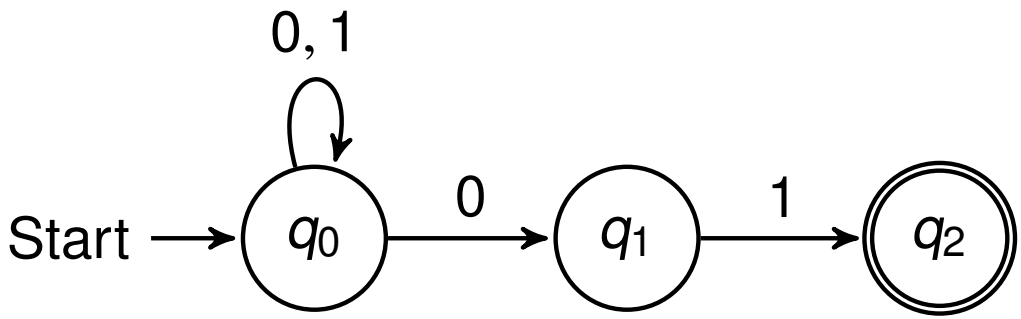
Definition of NFA

A nondeterministic finite automata (NFA) is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set of states,
- Σ is a finite set of input symbols (i.e. an alphabet),
- δ is a transition function from $Q \times \Sigma$ to the powerset 2^Q of Q,
- $q_0 \in Q$ is a start state,
- $F \subseteq Q$ is a set of final or accepting states.

Example

The NFA



can be specified formally as $A_2 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$, where δ is the transition function

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

Extended Transition Function

The transition function δ of an NFA can be extended to $\hat{\delta}$:

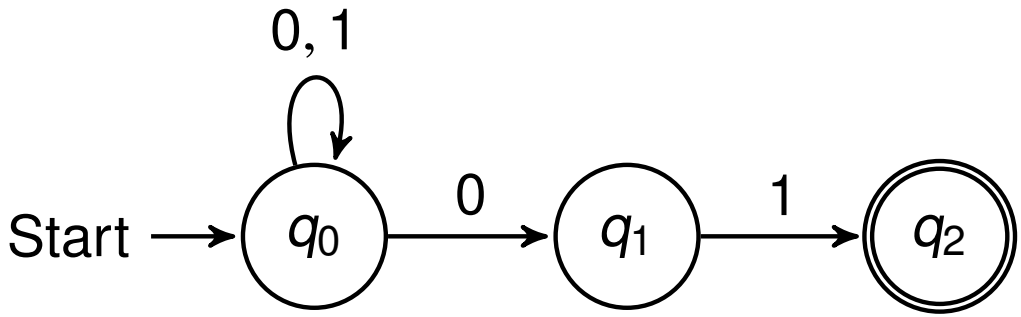
Basis step: $\hat{\delta}(q, \epsilon) = \{q\}$.

Inductive step: Suppose $w = xa$, then

$$\hat{\delta}(q, w) = \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a).$$

Example

Use $\hat{\delta}$ to describe the processing of input 00101 by the NFA A_2



- $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
- $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$
- $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

For any state q and input symbol a,

$$\hat{\delta}(q,a) = \bigcup_{p \in \hat{\delta}(q,\epsilon)} \delta(p,a) = \delta(q,a).$$

Furthermore, for all $q \in Q$ and all $x, y \in \Sigma^*$, one can prove:

$$\hat{\delta}(q,xy) = \bigcup_{p \in \hat{\delta}(q,x)} \hat{\delta}(p,y).$$

Do them!

Language of NFA

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

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

Example

For $A_2 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$, we will prove

$$L(A_2) = \{w \mid w \text{ ends in } 01\}.$$

Proof We'll do a mutual induction on the following three statements:

1. $q_0 \in \hat{\delta}(q_0, w)$ holds if $w \in \Sigma^*$,
2. $q_1 \in \hat{\delta}(q_0, w)$ holds iff $w = x0$,
3. $q_2 \in \hat{\delta}(q_0, w)$ holds iff $w = x01$.

The proof is an induction on the length of w, starting with length 0.

Basis step: If $|w| = 0$, then $w = \epsilon$. Statement (1) follows from the definition. Statements (2) and (3) follow by neither $\hat{\delta}(q_0, \epsilon) = \{q_0\}$, $w = x0$ nor $w = x01$ then.

Inductive step: Assume $w = xa$, where $a \in \{0, 1\}$, $|x| = n$ and statements (1), (2) and (3) hold for x. We will show that the statements hold for w.

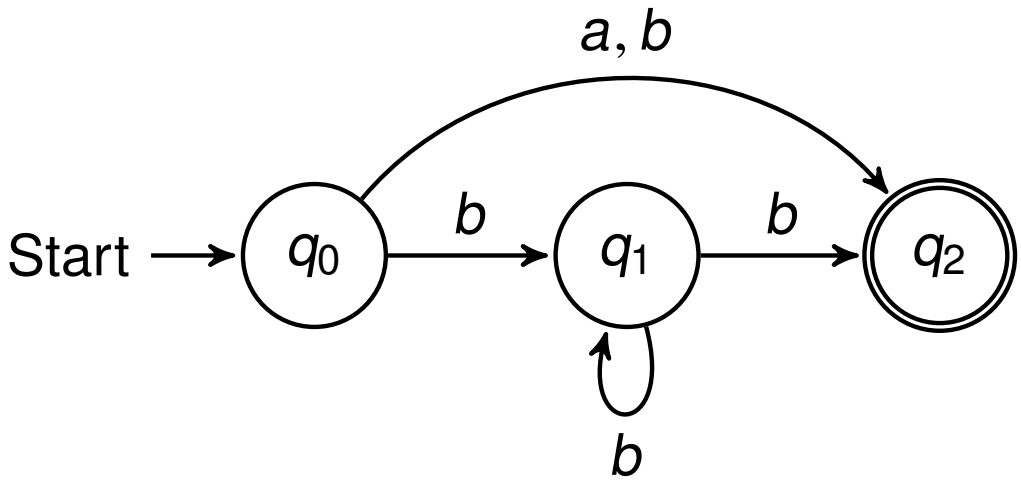
1. We know $q_0 \in \hat{\delta}(q_0, x)$. Since there are transitions on both 0 and 1 from q_0 to itself, it follows that $q_0 \in \hat{\delta}(q_0, w)$, so statement (1) is proved for w.
2. (if) Let $w = x0$. Applying statement (1) to x, we know $q_0 \in \hat{\delta}(q_0, x)$. So $q_1 \in \hat{\delta}(q_0, w)$ as there is a transition from q_0 to q_1 on input 0.
   (only-if) Suppose $q_1 \in \hat{\delta}(q_0, w)$. From the diagram of A_2, we see that the only way to state q_1 is on the input sequence $w = x0$.
3. (if) Let w end in 01. Then if $w = x1$, we know that x ends in 0. Applying statement (2) to x, we know $q_1 \in \hat{\delta}(q_0, x)$. So $q_2 \in \hat{\delta}(q_0, w)$ as there is a transition from q_1 to q_2 on input 1.
   (only-if) Suppose $q_2 \in \hat{\delta}(q_0, w)$. From the diagram of A_2, we see that the only way to state q_2 is on the input sequence $w = x1$, where $q_1 \in \hat{\delta}(q_0, x)$. Applying statement (2) to x, we know that x ends in 0. Thus, w ends in 01, and we have proved statement (3). □

Designing NFA

Example

Find an NFA with a single final state that accepts the set $\{a\} \cup \{b^n \mid n \geq 1\}$.

Solution One solution is



Why do we introduce nondeterminism?

- Many deterministic algorithms require that one make a choice at some stage. A typical example is a game-playing program. Frequently, the best move is not known, but can be found using an exhaustive search with backtracking. A nondeterministic algorithm that can make the best choice would be able to solve the problem without backtracking.
- Nondeterminism is sometimes helpful in solving problems easily.
- There is a technical reason for introducing nondeterminism.

Homework

P. 53: Exercises 2.2.2 & 2.2.4(c)

Please refer to the assignment requirement!