DFA and NFA

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

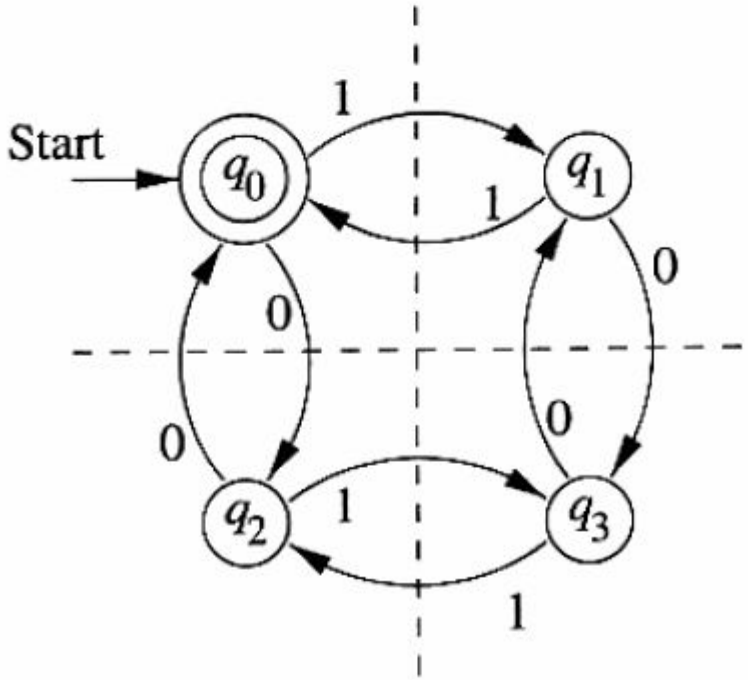
Then: $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$

- 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

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



Example

The check involves computing $\hat{\delta}(q_0, w)$ for each prefix w of 110101, starting at ϵ and going in increasing size. The summary of this calculation is:

- $\hat{\delta}(q_0, \epsilon) = q_0$.
- $\hat{\delta}(q_0, 1) = \delta\big(\hat{\delta}(q_0, \epsilon), 1\big) = \delta(q_0, 1) = q_1$.
- $\hat{\delta}(q_0, 11) = \delta\big(\hat{\delta}(q_0, 1), 1\big) = \delta(q_1, 1) = q_0$.
- $\hat{\delta}(q_0, 110) = \delta\big(\hat{\delta}(q_0, 11), 0\big) = \delta(q_0, 0) = q_2$.
- $\hat{\delta}(q_0, 1101) = \delta\big(\hat{\delta}(q_0, 110), 1\big) = \delta(q_2, 1) = q_3$.
- $\hat{\delta}(q_0, 11010) = \delta\big(\hat{\delta}(q_0, 1101), 0\big) = \delta(q_3, 0) = q_1$.
- $\hat{\delta}(q_0, 110101) = \delta\big(\hat{\delta}(q_0, 11010), 1\big) = \delta(q_1, 1) = q_0$.

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

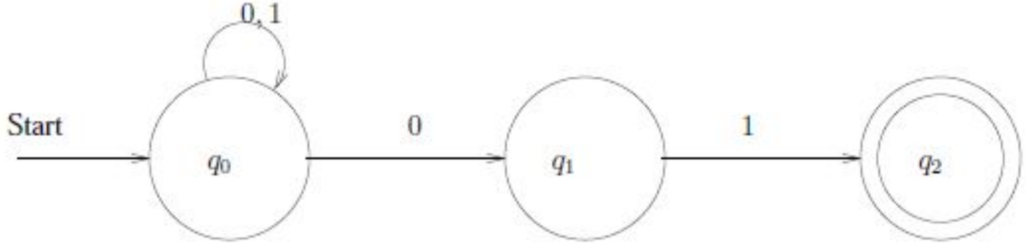
If L is a $L(A)$ from some DFA, then L is a *regular language*

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

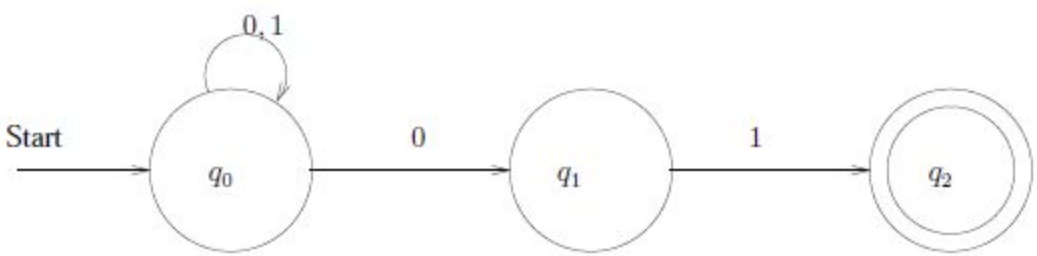
The only difference between a NFA and a DFA is in the type of value that δ returns

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:

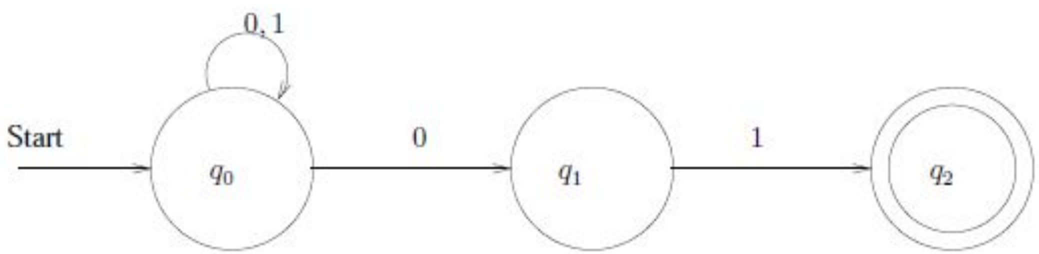
- Suppose w is a string of the form xa; that is a is the last symbol of w, and x is the string consisting of all but the last symbol
- Also suppose that $\hat{\delta}(q, x) = \{p_1, p_2, \dots p_k, \}$
- Let

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

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

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.