Lecture 3: More on DFAs

26 January 2010

This lecture continues with material from section 1.1 of Sipser.

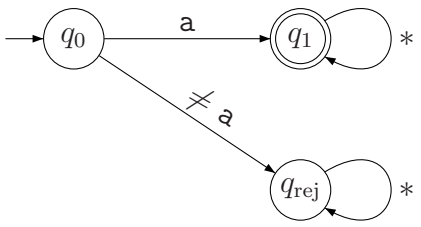
1 JFLAP demo

Go to `http://www.jflap.org`. Run the applet ("Try applet" near the bottom of the menu on the lefthand side). Construct some small DFA and run a few concrete examples through it.

2 State machines

2.1 A simple automata

Here is a simple ***state machine*** (i.e., finite automaton) M that accepts all strings starting with a.



Here $*$ represents any possible character.

Notice key pieces of this machine: three states, q_0 is the start state (arrow coming in), q_1 is the final state (double circle), transition arcs.

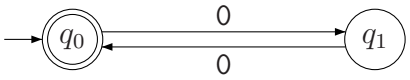
To run the machine, we start at the start state. On each input character, we follow the corresponding arc. When we run out of input characters, we answer "yes" or "no", depending on whether we are in the final state.

The language of a machine M is the set of strings it accepts, written $L(M)$. In this case $L(M) = \{\texttt{a}, \texttt{aa}, \texttt{ab}, \texttt{aaa}, \ldots\}$.

2.2 Another automata

(This section is optional and can be skipped in the lecture.)

Here is a simple ***state machine*** (i.e., finite automaton) M that accepts all ASCII strings ending with `ing`.



Notice key pieces of this machine: four states, q_0 is the start state (arrow coming in), q_3 is the final state (double circle), transition arcs.

To run the machine, we start at the start state. On each input character, we follow the corresponding arc. When we run out of input characters, we answer "yes" or "no", depending on whether we are in the final state.

The language of a machine M is the set of strings it accepts, written $L(M)$. In this case $L(M) = \{\texttt{walking}, \texttt{flying}, \texttt{ing}, \ldots\}$.

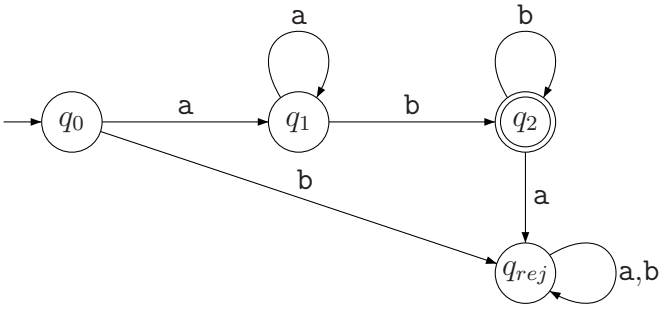
2.3 What automatas are good for?

People use the technology of automatas in real-world applications:

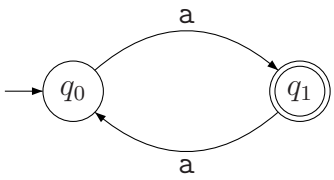
- Find all files containing `-ing` (grep)
- Translate each `-ing` into `-iG` (finite-state transducer)
- How often do words in Chomsky's latest book end in `-ing`?

2.4 DFA - deterministic finite automata

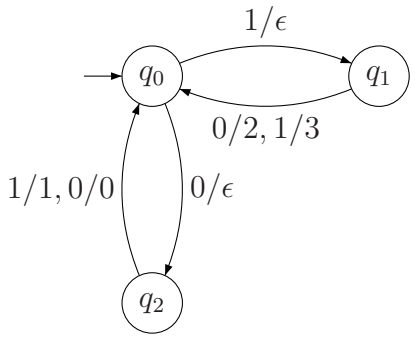
We will start by studying ***deterministic finite automata*** (DFA). Each node in a deterministic machine has exactly one outgoing transition for each character in the alphabet. That is, if the alphabet is $\{\texttt{a}, \texttt{b}\}$, then all nodes need to look like



Both of the following are bad, where $q_1 \neq q_2$ and the right hand machine has no outgoing transition for the input character `b`.



So our `-ing` detector would be redrawn as:

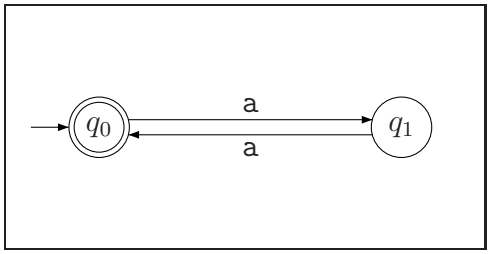


3 More examples of DFAs

3.1 Number of characters is even

Input: $\Sigma = \{0\}$.

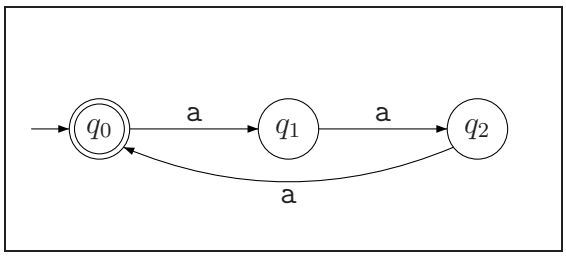
Accept: all strings in which the number of characters is even.



3.2 Number of characters is divisible by 3

Input: $\Sigma = \{0\}$.

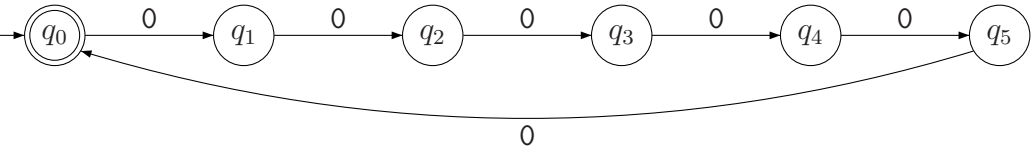
Accept: all strings in which the number of characters is divisible by 3.



3.3 Number of characters is divisible by 6

Input: $\Sigma = \{0\}$.

Accept: all strings in which the number of characters is divisible by 6.

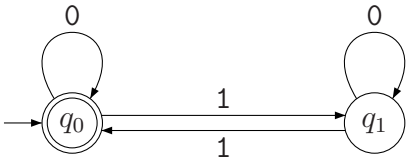


This example is especially interesting, because we can achieve the same purpose, by observing that $n \bmod 6 = 0$ if and only if $n \bmod 2 = 0$ and $n \bmod 3 = 0$ (i.e., to be divisible by 6, a number has to be divisible by 2 and divisible by 3 [a generalization of this idea is known as the Chinese remainder theorem]). So, we could run the two automatas of Section 3.1 and Section 3.2 in parallel (replicating each input character to each one of the two automatas), and accept only if both automatas are in an accept state. This idea would become more useful later in the course, as it provide a building operation to construct complicated automatas from simple automatas.

3.4 Number of ones is even

Input is a string over $\Sigma = \{0, 1\}$.

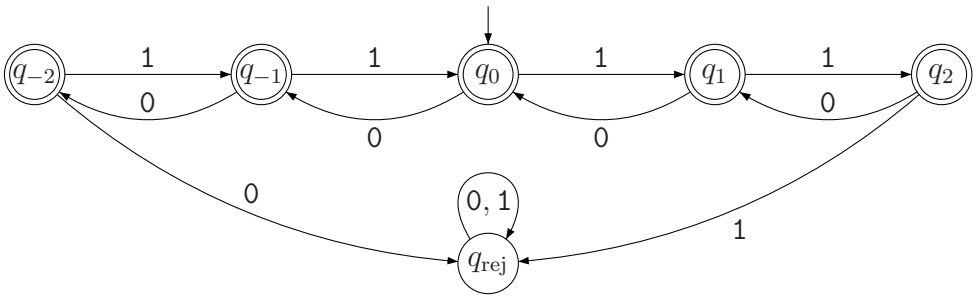
Accept: all strings in which the number of ones is even.



3.5 Number of zero and ones is always within two of each other

Input is a string over $\Sigma = \{0, 1\}$.

Accept: all strings in which the difference between the number of ones and zeros in any prefix of the string is in the range $-2, \ldots, 2$. For example, the language contains ϵ, 0, 001, and 1101. You even have an extended sequence of one character e.g. 001111, but it depends what preceded it. So 111100 isn't in the language.



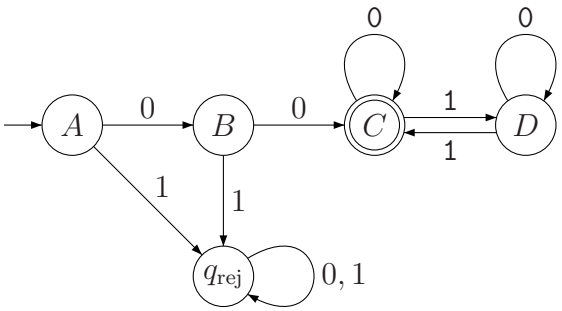
Notice that the names of the states reflect their role in the computation. When you come to analyze these machines formally, good names for states often makes your life much easier. BTW, the language of this DFA is

$$L(M) = \left\{ w \;\middle|\; w \in \{0,1\}^* \text{ and for every } x \text{ that is a prefix of } w, |\#1(x) - \#0(x)| \leq 2 \right\}.$$

3.6 More complex language

The input is strings over $\Sigma = \{0, 1\}$.

Accept: all strings of the form $00w$, where w contains an even number of ones.



You can name states anything you want. Names of the form q_X are often convenient, because they remind you of what's a state. And people often make the initial state q_0. But this isn't obligatory.

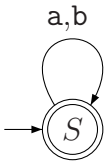
4 The pieces of a DFA

To specify a DFA (***deterministic finite automata***), we need to describe

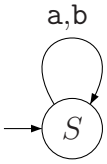
- a (finite) alphabet
- a (finite) set of states
- which state is the start state?
- which states are the final states?
- what is the transition from each state, on each input character?

5 Some special DFAs

For $\Sigma = \{a, b\}$, consider the following DFA that accepts Σ^*:

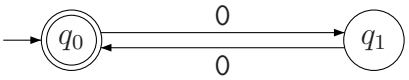


The DFA that accepts nothing, is just



6 Formal definition of a DFA

Consider the following automata, that we saw in the previous lecture:



We saw last class that the following components are needed to specify a DFA:

(i) a (finite) alphabet

(ii) a (finite) set of states

(iii) which state is the start state?

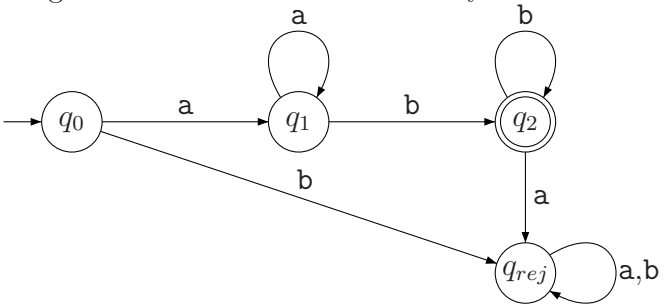
(iv) which states are the final states?

(v) what is the transition from each state, on each input character?

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

- Q: A finite set (the set of ***states***).
- Σ: A finite set (the ***alphabet***)
- $\delta : Q \times \Sigma \to Q$ is the ***transition function***.
- q_0: The ***start*** state (belongs to Q).
- F: The set of ***accepting*** (or ***final***) states, where $F \subseteq Q$.

For example, let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and consider the following DFA M, whose language $L(M)$ contains strings consisting of one or more a's followed by one or more b's.



Then $M = (Q, \Sigma, \delta, q_0, F)$, $Q = \{q_0, q_1, q_2, q_{rej}\}$, and $F = \{q_2\}$. The transition function δ is defined by

δ	a	b
q_0	q_1	q_{rej}
q_1	q_1	q_2
q_2	q_{rej}	q_2
q_{rej}	q_{rej}	q_{rej}

We can also define δ using a formula

$$\delta(q_0, a) = q_1$$

$$\delta(q_1, a) = q_1$$

$$\delta(q_1, b) = q_2$$

$$\delta(q_2, b) = q_2$$

$\delta(q, t) = q_{rej}$ for all other values of q and t.

Tables and state diagrams are most useful for small automata. Formulas are helpful for summarizing a group of transitions that fit a common pattern. They are also helpful for describing algorithms that modify automatas.

7 Formal definition of acceptance

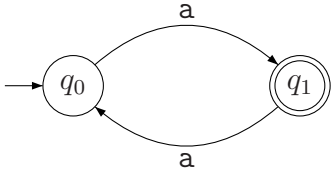
We've also seen informally how to run a DFA. Let us turn that into a formal definition. Suppose $M = (Q, \Sigma, \delta, q_0, F)$ is a given DFA and $w = w_1 w_2 \dots w_k \in \Sigma^*$ is the input string. Then M ***accepts*** w iff there exists a sequence of states $r_0, r_1, \dots r_k$ in Q, such that

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \dots, k-1$.
3. $r_k \in F$.

The ***language recognized*** by M, denoted by $L(M)$, is the set $\left\{ w \;\middle|\; M \text{ accepts } w \right\}$.

For example, when our automaton above accepts the string aabb, it uses the state sequence $q_0 q_1 q_1 q_2 q_2$. (Draw a picture of the transitions.) That is $r_0 = q_0$, $r_1 = q_1$, $r_2 = q_1$, $r_3 = q_2$, and $r_4 = q_2$.

Note that the states do not have to occur in numerical order in this sequence, e.g. the following DFA accepts aaa using the state sequence $q_0 q_1 q_0 q_1$.



A language (i.e. set of strings) is ***regular*** if it is recognized by some DFA.