Deterministic vs. Non-Deterministic FSA

Ling 106

October 1, 2003

1. Formal Definition of A Deterministic Finite State Automaton

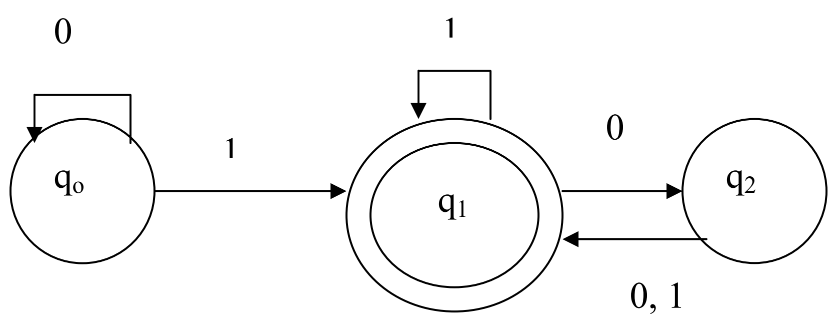
Definition 1.1

A final state automaton is a 5-tuple $\langle Q, \Sigma, \delta, q0, F \rangle$, where:

1. Q is a finite set of states;
2. Σ is a finite set called the alphabet,
3. δ: Q x Σ -> Q is the transition function,
4. q0 $\in$ Q is the start state,
5. $F \subseteq Q$ is the set of accept states.

Questions:
Can finite state machines have more than one accept states?
Can they have zero number of accept states?
Given the definition 1.1 of deterministic FSA, must there be exactly one transition arrow exiting every state for each possible input symbol?

For example, consider the state diagram of the automaton M_1:



We can describe M_1 formally by writing $M = \langle Q, \Sigma, \delta, q0, F \rangle$, where

1. Q={q0, q1, q2},
2. Σ = {0, 1}
3. δ is defined as

	0	*1*
q0	q0	q1
q1	q2	q1
q2	q1	q1

4. q0 is the start state,
5. F={q1}.

Question.
Given the formal description of finite state automaton M_2 below, draw a corresponding state diagram for M_2.

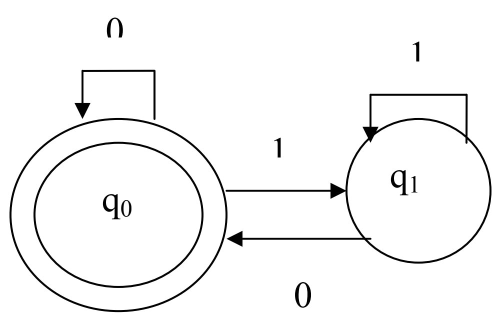
M_2 = <Q, Σ, δ, q0, F>, where
1. Q={q0, q1},
2. Σ = {0, 1}
3. δ is defined as

	0	*1*
q0	q0	q1
q1	q0	q1

4. q0 is the start state,
5. F={q1}.

Which of the following strings are accepted by M_2?
a. 0
b. 1
c. 00
d. 11111
e. 1000000
f. 1010011
g. ε

Question.
Consider the state diagram of finite state automaton M_3, and give a formal description of M_3.



M_2 = <Q, Σ, δ, q0, F>, where
1. Q=
2. Σ =

3. δ is defined as

	0	*1*
q0		
q1		

4. The start state is
5. F=

Which of the following strings are accepted by M_3?

a. 0
b. 1
c. 00
d. 11111
e. 1000000
f. 1010011
g. ε

2. The Language of a FSA

Let M denote a finite state automaton and w denote a string of symbols:

The **language of a machine M,** written as L(M), is the set of all strings that machine M accepts.
For example, $L(M_1)$ ={w | w ends with a 1}
Let A be L(M). We say that M **recognizes A** or that **M accepts A.**
The string w is in the language accepted by M, w∈L(M), if and only if M reads w entirely and halts in a final state of M.
A machine may accept several strings, but it always recognizes only one language.
If the machine accepts no strings, it still recognizes one language, namely, the empty language, written as ∅.

A language is **regular** if and only if there exists a finite state automaton that recognizes it. That is if we claim that a language is a regular language, we must be able to back up our claim by producing a finite state automaton that recognizes the language.

Question.
Show that the following languages are regular. That is, for each of these languages, give a FSA diagram that accepts all and only the strings in that language. Assume Σ={0, 1}.

a. {w: w contains at least three 1s}
b. {w: w begins with 1 and ends with 0}

3. The Regular Operations

Definition 1.2
Let A and B be languages (with the same alphabet Σ). We define the regular operations **union, intersection, concatenation, and star** as follows.

- Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- Intersection: $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
- Concatenation: $A o B = \{xy \mid x \in A \text{ or } x \in B\}$
- Star: $A^* = \{x_1 x_2 x_3 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$
  A^* is the set formed by concatenating members of A together any number of times (including zero) in any order and allowing repetitions.

Examples
Let the alphabet Σ be the standard 26 letters {a, b, c, … z}. If A={good, bad} and B={boy, girl}, then
$A \cup B$={good, bad, girl, boy}
$A \cap B = \emptyset$
AoB={goodboy, goodgirl, badboy, badgirl}
A^*= {ε, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood,goodgoodbad, goodbadgood, badgoodgood, badbadbadbad, badbadbadgood, ….}

Note that the empty string ε is always a member of A^*, no matter what A is!

Question
Let C= {apple, banana, pear} and D={grape, banana}.
$C \cup D$=
$C \cap D$=
CoD=
D^*=

Properties of regular languages

Theorem 1.1
The class of regular languages is closed under the union operation
That is, if A and B are regular languages, then $A \cup B$ is also a regular language

Theorem 1.2
The class of regular languages is closed under the intersection operation
That is, if A and B are regular languages, then $A \cap B$ is also a regular language

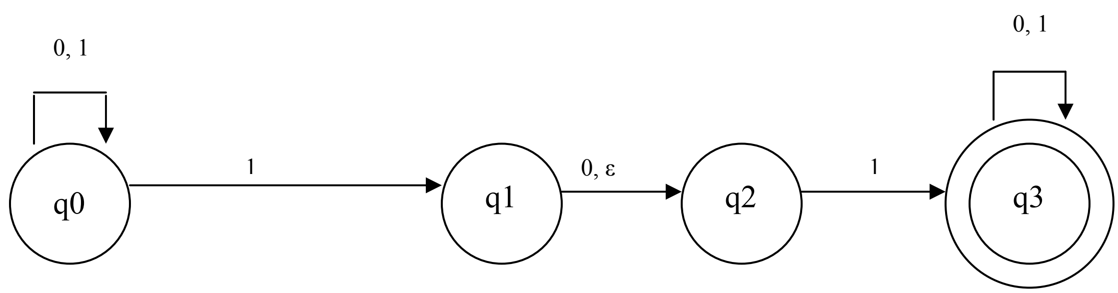
Theorem 1.3
The class of regular languages is closed under the concatenation operation
That is, if A and B are regular languages, then AoB is also a regular language.

Theorem 1.4
The class of regular languages is closed under the star operation
That is, if A is a regular language, then A^* is also a regular language.

4. Characteristics of Nondeterministic Finite State Automata

Every state of a deterministic finite state automaton (DFA) always has exactly one exiting transition arrow for each symbol in the alphabet. That is, in a DFA, the transition function is a true function. This means that there was never any choice about what to do at any step in the computation.
e.g. $\delta(q0, a)=q1$
But in a nondeterministic finite state automaton (NFA), a state may have zero, one, or many exiting arrows for each alphabet symbol. That is, we can think of the transition function of NFA as retuning not a unique state, but a set of states, including the empty set.
e.g. $\delta(q0, a)=\{q0, q1\}$

Consider the machine below called N1.



Things to note

- Because the transition arrow from q1 to q2 is labeled with the empty string and a 0, the machine can move from state q1 to state q2 either by scanning 0 or simply by ‘jumping’ from q1 to q2.
- If the machine read a 1 in a state q0, it can either remain in q0 or it can move to state q2.
- In state q1, there is no exiting arrow for 1, and in state q2, there is no exiting arrow for 0.

Example computation

Does the above machine N1 accept 010110?

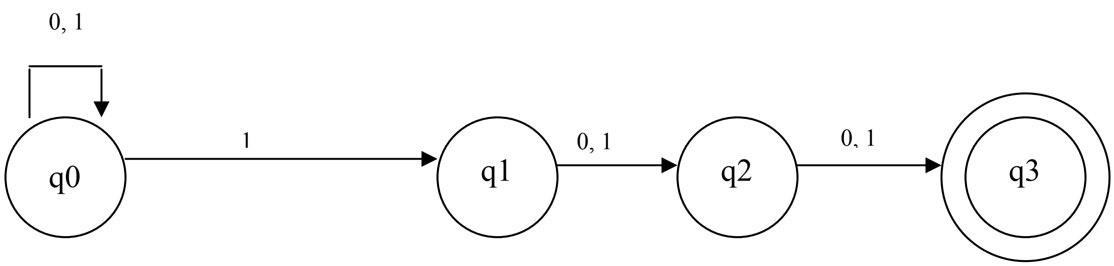
1. We start at q0 and we read the first symbol in the string $\underline{0}.10110$
2. We remain in state q0 and move to the next symbol $0\underline{1}.0110$

3. We are now faced with two options. We ‘copy’ the machine and let each copy carry on the computation.
Copy-1: stays in q0;
Copy-2: moves to q1.
4. An empty string exits state q1, so the machine makes another copy.
Copy-1: stays in q0.
Copy-2: moves to q1.
Copy-3: moves to q2.
5. We move to the next symbol in the input string: 010.110
Copy-1: stays in q0.
Copy-2: moves to q2.
Copy-3: dies.
6. We now scan the fourth symbol in the string: 0101.10
This is ambiguous for copy-1, so it generates a new copy
Copy-1: stays in q0.
Copy-4: moves to q1.
Copy-2: moves to q3.
7. An empty string exits state q1, so copy-4 makes another copy.
Copy-1: stays in q0.
Copy-4: moves to q1.
Copy-5: moves to q2.
Copy-2: moves to q3.
8. Next we look at the fifth symbol in the string: 01011.0
This is ambiguous for copy-1, so it generates a new copy:
Copy-1: stays in q0.
Copy-6: moves to q1.
Copy-4: dies
Copy-5: moves to q3.
Copy-2: stays in q3.
9. An empty string exists state q1, so copy06 makes another copy.
Copy-1: stays in q0.
Copy-6: stays in q1.
Copy-7: moves to q2.
Copy-5: stays in q3.
Copy-2: stays in q3.
10. Since copy-5 and copy-2 are in the same state, the two copies collapse into one.
Copy-1: stays in q0.
Copy-6: stays in q1.
Copy-7: moves to q2.
Copy-2: stays in q3.
11. Now we scan the last symbol in the string: 010110.
Copy-1: stays in q0.
Copy-6: moves to q2.
Copy-7: dies.
Copy-2: stays in q3.

We've reached the end of the string, so we check to see if any of the copies are in an accepting state. Since copy-2 is in q3, which is an accepting state, we can conclude that the machine N1 accepts the string.

Question: Which of the following strings does the above machine N1 accept?
a. 1011
b. 101010
c. 000001
d. 1100101

Questions: What strings does the following NFA accept?



5. Formal Definition of NFA

Definition 1.2
A nondeterministic finite state automaton is a 5-tuple <Q, Σ, δ, q0, F>, where:

1. Q is a finite set of states;
2. Σ is a finite set called the alphabet,
3. $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \wp(Q)$ is the transition function,
4. $q0 \in Q$ is the start state,
5. $F \subseteq Q$ is the set of accept states.

We can describe N1 formally by writing N1=<Q, Σ, δ, q0, F>, where

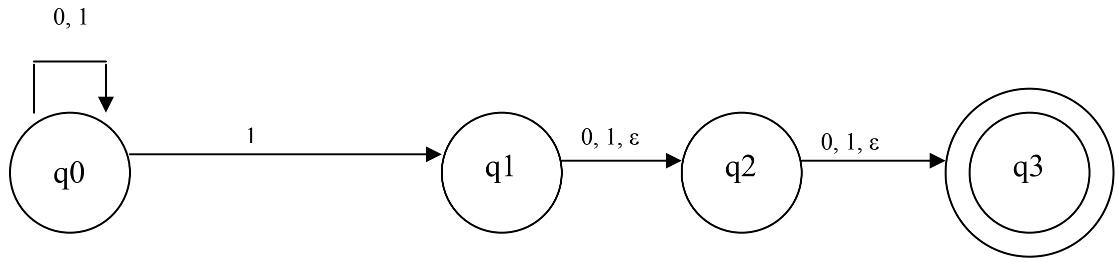
1. Q={q0, q1, q2},
2. Σ = {0, 1}
3. δ is defined as

	0	*1*	*ε*
q0	{q0}	{q0,q1}	∅
q1	{q2}	∅	{q2}
q2	∅	{q3}	∅
q3	{q3}	{q3}	∅

4. q0 is the start state,

5. F={q3}.

Question
Give a formal description of the following machine N2.



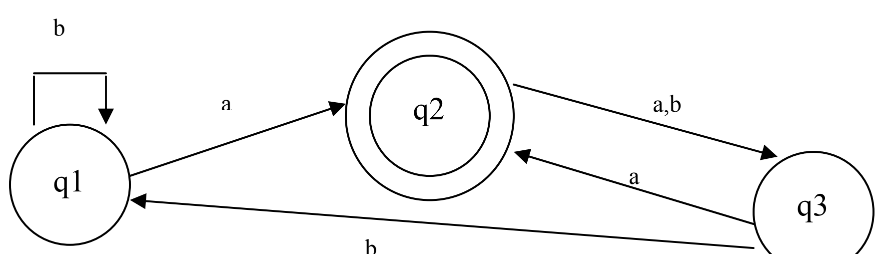
What strings does the machine N2 accept?

5. Equivalence of NFAs and DFAs

We say that two machines are EQUIVALENT if they recognize the same language.
It can be shown that for any NFA there is an equivalent DFA – equivalent in the sense that both accept the same set of strings.
A DFA typically has more states that an NFA to which it is equivalent. The DFA works by keeping track of the set of states that the NFA could be in if followed all possible paths simultaneously on a given input.

Extra exercises.

Exercise 1. Consider the state diagram for deterministic finite state automaton M_4, and give a formal description of M_4.



Which of the following strings does M_4 accept?
a. aabb
b. abaab
c. bbbba
d. baaba
e. aabaa
f. ε

g. ba
h. a
i. aaaaa
j. abaa

Exercise 2.
For each of the following languages, give a FSA that recognizes it. Assume $\Sigma=\{1, 0\}$.
a. {w | w contains exactly three 1s(and any number of 0s)}
b. {w: w has an odd length}
c. {1}
d. {w: w contains the substring 111}
e. {w: the length of w is a multiple of 3}

Exercise 3.

Give a formal description of the following nondeterministic machine N3.