Nondeterministic Operation

So far, every step taken by a finite automaton has been *exactly* determined by the state of the machine and the symbol read. No choices have existed. This mode of operation is called *determinism* and machines of this ilk are known as *deterministic finite automata*. Finite automata need not be so unambiguous. We could have defined them to have some choices of which state to advance to on inputs. Examine figure 1, it provides the state graph of such a machine.

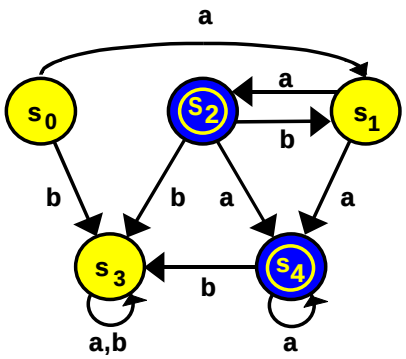


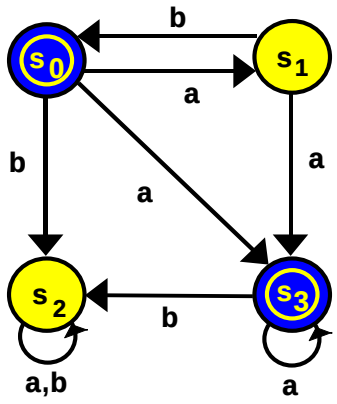
Figure 1 - Nondeterministic Finite Automaton

If the input to this machine begins with the string ab, it will progress from s_0 to s_1 to s_2. Now, if the next input symbol is an a, it has a choice of whether to go back to s_1 or to move to s_4. So, now what does it do? Well, it transfers to the state which will eventually take it into a final state if possible. Simple as that! But, how does it know when it has not seen the remainder of the input yet? Do not worry about that - it always chooses the right state!

The above explanation of the operation of the machine of figure 1 *is* a slight bit mystical. But we can (if we wish) define machines with choices in their transition functions. And we can (if we wish) define acceptance to take place whenever a correct sequence of states under the input will end up in a final state. In other words, *if the machine can get to a final state* in some proper manner, it will accept. Now for a formal definition.

> **Definition.** *A **nondeterministic** finite automaton is the quintuple* $M = (S, I, \delta, S_0, F)$ *where S, I, and F are as before but:*
>
> $S_0 \in S$ *(a set of starting states), and*
> $\delta(s,a) \subseteq S$ *for each* $s \in S$ *and* $a \in I$ *.*

Now instead of having a starting *state* and a *transition function*, we have a starting *state set* and a *set of transition states*. More on this later. For now, note that the only differences in the finite automaton definitions was that the machine now has a choice of states to start in and a choice of transitions under a state-symbol pair. A reduced version of the last nondeterministic machine is presented in figure 2.



State	Input a	Input b	Accept?
0	1,3	2	yes
1	3	0	no
2	2	2	no
3	3	2	yes

Figure 2 - Reduced Nondeterministic Machine

(**N.B.** We must note that the transition indicator δ is not a function any more. To be precise about this we must state that it has become a *relation*. In other words, since δ(s, a) is a *set*, it indicates which states are members of δ(s, a). If that is confusing then forget it and consider δ(s, a) to be a *set*.) Our next step is to define acceptance for nondeterministic finite automata. We could extend the transition indicator so that it will handle strings. Since it provides a set of next states, we will just note the set of states the machine is in after processing a string. Let's look at a picture. Think about the last machine (the one in figure 2). Now imagine what states it might go through if it processed all possible strings of length three. Now, look at figure 3.

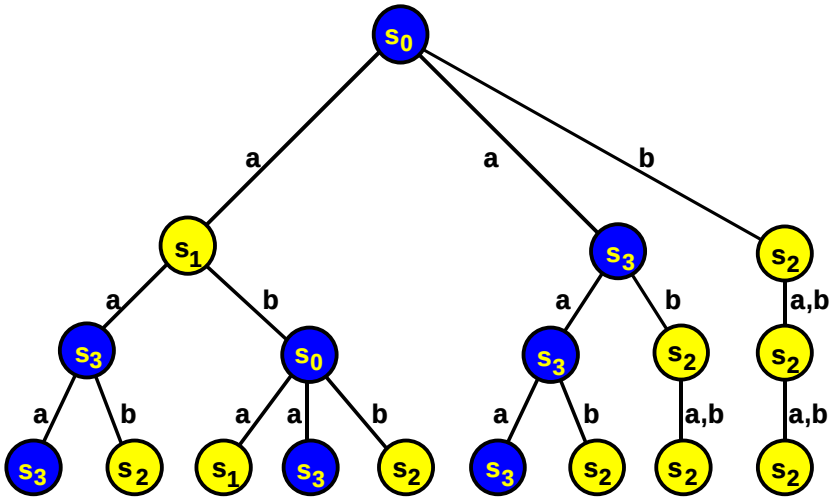


Figure 3- Computation Tree

In processing the string abb, the machine ends up in s_2, but it can get there in two different ways. These are:

$$s_0 \rightarrow s_1 \rightarrow s_0 \rightarrow s_2$$
$$s_0 \rightarrow s_3 \rightarrow s_2 \rightarrow s_2$$

Likewise, the machine can end up in 3 after processing the string aaa. But since the automaton is nondeterministic, it can have the option of ending up in several states after processing an input. For example, after aba, the set of states $\{s_1 , s_2 , s_3\}$ is reached.

In fact, a *set* of states is reached by the machine after processing an input. This set depends upon the choices the automaton was offered along the way. And, if a final state was in the set reached by the machine, we accept. In the above example only the strings aba and aaa can be accepted because they were the only strings which took the automaton into s_3.

This gives a definition of δ^* as the *set of states reached by the automaton* and the tapes accepted as:

$$T(M) = \{ x \mid \delta^*(s_0,x) \cap F \neq \varnothing \}$$

On the other hand, instead of extending δ to strings as with the deterministic case, we shall discuss sequences of state transitions under the state transition indicator δ. The following formal definition of acceptance merely states that a string is accepted by a nondeterministic finite automaton if there is a sequence of states (or path through the computation tree) which leads from a starting

state to a final state under the input string. (Note that we do not discuss how a machine might find this sequence - that does not matter! We just say that the input is accepted if there exists such a sequence of states.)

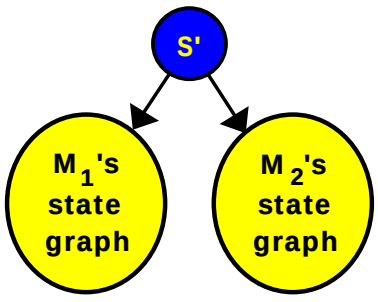
> **Definition.** *The input* $x = x_1 \ldots x_n$ *is* ***accepted*** *by the nondeterministic finite automaton* $M = (S, I, \delta, S_0, F)$ *if and only if there is a sequence of states:* $s_{k_1}, s_{k_2}, \ldots, s_{k_{n+1}}$ *where:*
>
> *a)* $s_{k_1} \in S_0$
>
> *b) for each* $i \leq n$: $s_{k_{i+1}} \in \delta(s_{k_i}, x_i)$
>
> *c)* $s_{k_{n+1}} \in F$.

With this definition of acceptance in mind, we define the set of *tapes accepted* by a nondeterministic finite automaton (denoted T(M) as before) as exactly those inputs for which there is a sequence of states leading from a starting state to an accepting state.

Nondeterminism is useful because it allows us to define very simple machines which perform certain operations. As an example, let's revisit the union closure problem. As usual, suppose we have the two finite automata $M_1 = (S, I, \delta, s_0, F)$ and $M_2 = (Q, I, \gamma, q_0, G)$ and wish to build a machine which accepts the union of the sets they accept. Our new union machine contains the states of both M_1 and M_2 plus a new starting state. This new starting state leads into the states of either M_1 or M_2 in a nondeterministic manner. That is, under the first input symbol, we would advance to an appropriate state in M_1 *or* an appropriate state in M_2. Formally, if $I = \{0,1\}$, the transition indicator of the union machine is ξ, and its starting state s' then:

$$\xi(s', 0) = \{\delta(s_0, 0), \gamma(q_0, 0)\}$$
$$\xi(s', 1) = \{\delta(s_0, 1), \gamma(q_0, 1)\}$$

The rest of the transition relation ξ is just a combination of δ and γ, and the state graph for the new union machine might resemble:



Thus the union machine is $M = (S \cup Q \cup \{s'\}, I, \xi, s', F \cup G)$. Acceptance takes place whenever there is a sequence to an accepting state through either the state graph of M_1 or that of M_2.

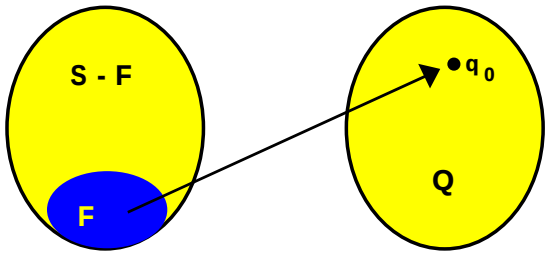
Let's try another closure property and build a nondeterministic machine which realizes it. This will be a string property called *concatenation* or juxtaposition. For strings this is easy, the concatenation of x and y is xy. If we have the sets A and B then the concatenation of them is defined as:

$$AB = \{ xy \mid x \in A \text{ and } y \in B \}.$$

If A and B can be accepted by the deterministic finite automata $M_1 = (S, I, \delta, s_0, F)$ and $M_2 = (Q, I, \gamma, q_0, G)$ we need to try and construct a machine M_3 which will accept AB. Let us look at:

$$M_3 = (S \cup Q, I, \xi, \{s_0\}, G)$$

and define ξ so that M_3 accepts AB. Our strategy will be to start out processing as M_1 might and then when M_1 wishes to accept a part of the input, switch to M_2 and continue on. With any luck we will end up in G, the final state set of M_2. Nondeterminism will be used to make the change from M_1 to M_2. The transition relation ξ will operate just like δ on the state set S and like γ on the state set Q except for the final states $F \subseteq S$. There it will include a transition to the starting state q_0 of M_2. One might picture this as:



and define the transition relation precisely as:

$$\xi(s_i, a) = \{\delta(s_i, a)\} \text{ for } s_i \in S\text{-}F$$
$$\xi(s_i, a) = \{\delta(s_i, a), q_0\} \text{ for } s_i \in F$$
$$\xi(q_i, a) = \{\gamma(q_i, a)\} \text{ for } q_i \in Q$$

By the definition of acceptance for nondeterministic machines, M_3 will accept a string if and only if there is a sequence of states (under the direction of ξ) which leads from s_0 to a state in G. Suppose z is a member of AB. Then:

$$z \in AB \text{ iff } z = xy \text{ where } x \in A \text{ and } y \in B$$
$$\text{iff } x \in T(M_1) \text{ and } y \in T(M_2)$$
$$\text{iff } \delta^*(s_0,x) \in F \text{ and } \gamma^*(q_0,y) \in G$$

This means that there is a sequence of states

$$s_{k_1}, s_{k_2}, \ldots, s_{k_n}$$

in S from $s_0 = s_{k_1}$ to $s_{k_n} \in F$ under δ and x. Also, there is a sequence of states in Q

$$q_{k_1}, q_{k_2}, \ldots, q_{k_m}$$

from $q_0 = q_{k_1}$ to $q_{k_m} \in G$ under γ and y. Since ξ is defined to be just like δ on S and like γ on Q, these sequences of states in S and Q exist under the influence of ξ and x and ξ and y. We now note that instead of going to the last state s_{k_n} in the sequence in S, ξ could have directed transferred control to q_0. Thus there is a sequence:

$$s_0, s_{k_2}, \ldots, s_{k_{n-1}}, q_0, q_{k_2}, \ldots, q_{k_m}$$

under ξ and z = xy which proceeds from s_0 to G. We can now claim that

$$T(M_3) = AB = T(M_1)T(M_2)$$

noting that the only way to get from s_0 to a state in G via ξ is via a string in AB.

A final note on the machine M_3 which was just constructed to accept AB. If A contains the empty word then of course s_0 is a final state and thus q_0 also must be a member of the starting state set. Also, note that if B contains the empty word, then the final states of M_3 must be $F \cup G$ rather than merely G.

Now we shall move along and look at multiple concatenation. Suppose we concatenated the same set together several times. Putting this more formally and in superscript notation, let:

$$A^0 = \{\varepsilon\}$$
$$A^1 = A$$
$$A^2 = AA$$
$$A^3 = AA^2 = AAA$$

and so forth. The general scheme for this is:

$$A^0 = \{\varepsilon\}$$
$$A^{i+1} = A^iA$$

To sum everything up, we consider the union of this infinite sequence of concatenations and call it the *Kleene closure* of the set A. Here is the definition.

> **Definition.** *The **Kleene Closure** (written A^*) of the set A is the union of A^k over all integer values of k.*

This operator is known as the *Kleene Star Operator* and so A^* is pronounced *A star*. One special case of this operator's use needs to be mentioned. If A is the set {a,b} (in other words: an alphabet), then A^* is the set of all strings over the alphabet. This will be handy.

To accept the Kleene closure of a set accepted by a deterministic finite automaton a construction similar to that used above for concatenation works nicely. The strategy we shall use is to allow a reset to the starting state each time a final state is entered. (That is, start over whenever a string from the original set could have ended.) We shall present the construction below and leave the proof of correctness as an exercise.

For a deterministic finite automaton $M = (S, I, \delta, s_0, F)$ we must build a (nondeterministic) machine which accepts $[T(M)]^*$. As we mentioned, our strategy will be to allow the transition relation to reset to s_0 whenever a final state is reached. First we shall introduce a new starting state s' and the following transition relation γ for all a U I:

$$\gamma(s',a) = \{\delta(s_0,a)\}$$
$$\gamma(s_i,a) = \{\delta(s_i,a)\} \text{ for } s_i \in S\text{-}F$$
$$\gamma(s_i,a) = \{ s_0, \delta(s_i,a)\} \text{ for } s_i \in F$$

The machine which accepts $[T(M)]^*$ is $M' = (S\cup\{s'\},I,\gamma,\{s'\},F\cup\{s'\})$. Now that we have seen how useful nondeterministic operation can be in the design of finite automata, it is time to ask whether they have more power than their deterministic relatives.

Theorem 3. *The class of sets accepted by finite automata is exactly the same class as that accepted by nondeterministic finite automata.*

Proof Sketch. Two things need to be shown. First, since deterministic machines are also nondeterministic machines (which do not ever make choices) then the sets they accept are a subclass of those accepted by nondeterministic automata.

To show the other necessary relation we must demonstrate that every set accepted by a nondeterministic machine can be accepted in a deterministic manner. We shall bring forth again our pebble automaton model.

Let $M_n = (S, I, \delta, S_0, F)$ be an arbitrary nondeterministic finite automaton. Consider its state graph. Now, place a pebble on each state in S_0. Then, process an input string under the transition relation δ. As δ calls for transitions from state to state, move the pebbles accordingly. Under an input a, with a pebble on s_i, we pick up the pebble from s_i and place pebbles on every state in $\delta(S_i, a)$. This is done for all states which have pebbles upon them. Indeed, this is parallel processing with a vengeance! (Recall the computation tree of figure 3, we are just crawling over it with pebbles - or processors.) After the input has been processed, we accept if any of the final states have pebbles upon them.

Since we moved pebbles on all of the paths M_n could have taken from a starting state to a final state (and we did not miss any!), we should accept whenever M_n does. And, also, if we accept then there was indeed a path from a starting state to a final state. Intuitively it all seems to work.

But, can we define a deterministic machine which does the job? Let's try. First let us define some states which correspond to the pebbled states of M_n. Consider:

q_1 means a pebble upon s_0
q_2 means a pebble upon s_1
q_3 means pebbles upon s_0 and s_1
q_4 means a pebble upon s_2
q_5 means pebbles upon s_0 and s_2
⋮
$q_{2^{n+1}-1}$ means pebbles upon $s_0, s_1, \ldots, s_n$

This is our state set Q. The starting state is the q_k which means pebbles on all of S_0. Our set of final states (G) includes all q_i which have a pebble on a state of F. All that is left is to define the transition function γ and we have defined a deterministic finite automaton $M_d = (Q, I, \gamma, q_k, G)$ which should accept the same set as that accepted by M_n.

This is not difficult. Suppose q_i means pebbles upon all of the states in $s' \subseteq S$. Now we take the union of $\delta(s_k,a)$ over all $s_k \in s'$ and find the q_j which means pebbles on all of these states. Then we define $\gamma(q_i, a) = q_j$.

The remainder of the proof involves an argument that M_n and M_d accept the same set.

With this theorem and our previous constructions firmly in mind, we now state the following.

Corollary. *The class of sets accepted by finite automata is closed under concatenation and Kleene closure.*

A cautionary remark must be made at this time. It is rather wonderful to be able to use nondeterminism in our machine constructions. But, when these machines are converted to deterministic finite automata via the above construction, there is an *exponential* state explosion! A four state machine would have fifteen states when converted to a deterministic machine. Imagine what a 100 state machine would look like when converted! Fortunately most of these machines can be reduced to a more manageable size.