07 - Non-Deterministic Finite Automata (NFA)

A non-deterministic finite automaton is one that (1) allows multiple arcs with the same label to exit a node, and (2) allows arcs with a λ–label. It is also customary in an NFA to relax the completeness constraint, meaning that missing arcs are assumed to lead to a non-accepting "sink" state.

Formally, the transition function for a DFA is defined as:

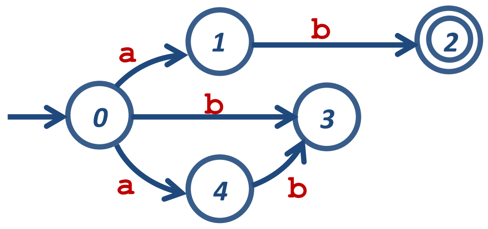
F: S x A → S

Whereas the transition function for a NFA is defined as:

F: S x (A ∪ {λ}) → 𝒫(S)

(recall that **S** is the set of states, **A** is the alphabet, and 𝒫 is the *powerset*).

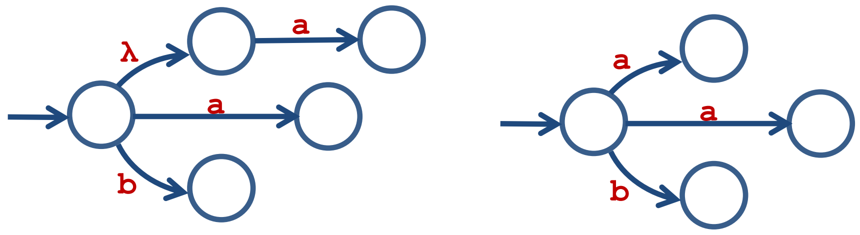
Consider the following NFA:



The string `aa` would not be accepted, because there is no path for aa that leads to an accepting state. In fact, after the first "a", there is no arrow at all for the second "a". Thus the second "a" leads to non-acceptance.

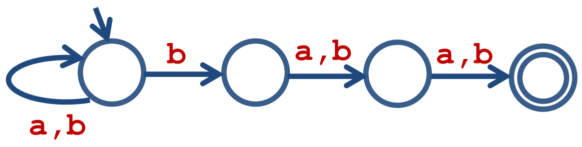
The string `ab` on the other hand has ***two*** possible paths: 0 → 1 → 2 which is accepting, and 0 → 4 → 3 which is non-accepting. Since there ***is*** a path that leads to an accepting state, the NFA accepts the string.

An λ-label on an arc allows for transition without consuming a character:



The two diagrams above are equivalent.

It is usually easier to construct an NFA than a DFA. Here is an NFA that recognizes all strings containing “b” in the third position from the end:

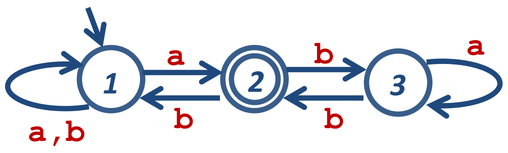


There are a variety of interpretations of the computational model that is represented by an NFA:

- Breadth-first parallel processes produced at each decision point,
- Depth-first trial-and-error with backtracking

More examples:

Given the following NFA:

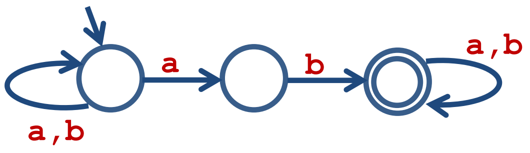


Which of the following strings are accepted, and by what path?

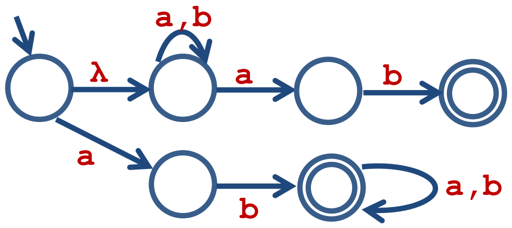
1. **aaaa** *yes – path is 11112*
2. **babbb** *no*
3. **babab** *yes – path is 112332*

Construct NFAs for the following:

- Strings with at least one occurrence of ab (use only 3 states):



- Strings that start with ab and/or end with ab (use only 6 states):



Relationship between NFA and DFA

- Diagrammatically, DFA are a subset of NFA.
- However, they both have the same expressive power.

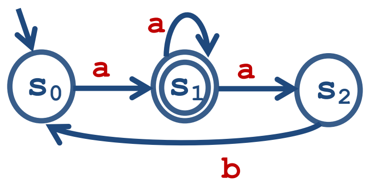
Theorem: For every NFA, there is an equivalent DFA.
Proof: We build a DFA (“D”) out of subsets of the states in an NFA (“N”):

$M_N = (S_N, A, s_0, Y_N, F_N)$
$M_D = (S_D, A, \{s_0\}, Y_D, F_D)$

1. Start building the graph for D by creating a start state $\{s_0\}$.
2. Repeat the following steps until no more edges are missing:
   a. Pick a node $\{s_i, s_j, ...\}$ in D that is missing an edge for symbol *x*.
   b. Build the union of nodes in N from s_i, s_j, etc. with arcs for *x*.
   c. If the resulting union $\{s_k, s_l, \dots\}$ doesn’t exist in D, create it.
   d. Add an edge from $\{s_i, s_j, \dots\}$ to $\{s_k, s_l, \dots\}$ and label it with *x*.
3. Every state in D that contains any state in Y_N is an accept state.
4. If M_N accepts λ, then the start state $\{S_0\}$ is also an accept state.
5. Repeat the process until no transitions are missing.

We defer the handling of λ-transitions until later.

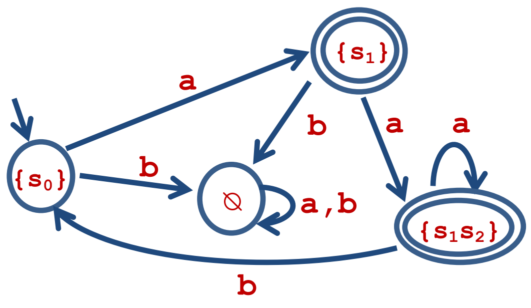
Example:



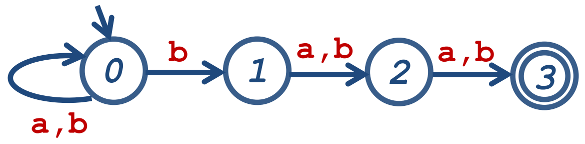
	a	b
s_0	s_1	∅
s_1	$s_1 s_2$	∅
s_2	∅	s_0

The transitions in the above NFA are shown above, to the right.
The constructive steps above results in the DFA as follows:

	a	b
$\{s_0\}$	$\{s_1\}$	∅
$\{s_1\}$	$\{s_1 s_2\}$	∅
$\{s_1 s_2\}$	$\{s_1 s_2\}$	s_0
∅	∅	∅



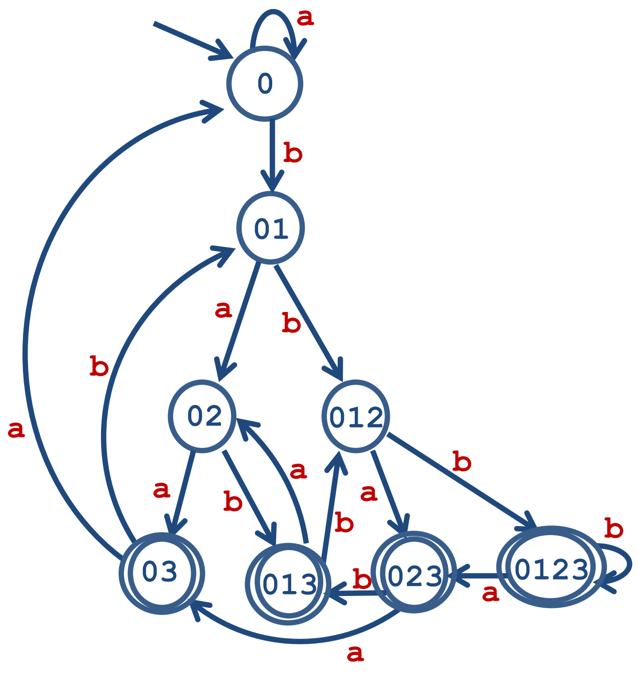
Consider the NFA shown on page 1:



	a	b
0	0	0,1
1	2	2
2	3	3
3	∅	∅

Conversion to a using the constructive steps produces the following DFA:

	a	b
0	0	01
01	02	012
02	03	013
03	0	01
012	023	0123
013	02	012
023	03	013
0123	023	0123



0
01
02
012
03
013
023
0123
a
b
a
b
b
a
a
b
a
a
b
b
b
a
b
a