3 Finite Automata

Finite automata correspond to a computer with a fixed finite amount of memory[4]. We will introduce *deterministic finite automata* (DFA) first and then move to *nondeterministic finite automata* (NFA). An automaton will accept certain words (sequences of symbols of a given alphabet Σ) and reject others. The set of accepted words is called the language of the automaton. We will show that the class of languages that are accepted by DFAs and NFAs is the same.

3.1 Deterministic finite automata

3.1.1 What is a DFA?

A *deterministic finite automaton* (DFA) $A = (Q, \Sigma, \delta, q_0, F)$ is given by:

1. A finite set of *states* Q
2. A finite set of input symbols, the *alphabet*, Σ
3. A *transition function* $\delta \in Q \times \Sigma \to Q$
4. An *initial state* $q_0 \in Q$
5. A set of *final states* $F \subseteq Q$

The initial states are sometimes called *start states*, and the final states are sometimes called *accepting states*.

As an example consider the following automaton

$$D = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta_D, q_0, \{q_2\})$$

where

$$\delta_D = \{((q_0, 0), q_1), ((q_0, 1), q_0), ((q_1, 0), q_1), ((q_1, 1), q_2), ((q_2, 0), q_2), ((q_2, 1), q_2)\}$$

if we view a function as a set of argument-result pairs. Alternatively, we can define it case by case:

$$\begin{aligned}
\delta_D(q_0, 0) &= q_1 \\
\delta_D(q_0, 1) &= q_0 \\
\delta_D(q_1, 0) &= q_1 \\
\delta_D(q_1, 1) &= q_2 \\
\delta_D(q_2, 0) &= q_2 \\
\delta_D(q_2, 1) &= q_2
\end{aligned}$$

A DFA may be more conveniently represented by a *transition table*. The transition table for the DFA D is:

δ_D	0	1
$\to q_0$	q_1	q_0
q_1	q_1	q_2
$* q_2$	q_2	q_2

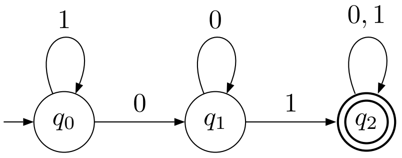
[4]However, that does not mean that finite automata are a good model of general purpose computers. A computer with n bits of memory has 2^n possible states. That is an absolutely enormous number even for very modest memory sizes, say 1024 bits or more, meaning that describing a computer using finite automata quickly becomes infeasible. We will encounter a better model of computers later, the *Turing Machines*.

A transition table represents the transition function δ of a DFA; i.e., the value of $\delta(q, x)$ is given by the row labelled q in the column labelled x. In addition, the initial state is identified by putting an arrow $\rightarrow$ to the left of it, and all final states are similarly identified by a star $*$. The inclusion of this additional information makes a transition table a self-contained representation of a DFA.

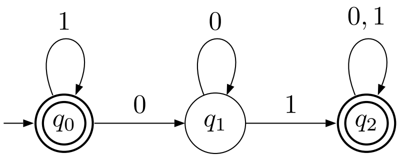
Note that the initial state also can be final (accepting). For example, for a variation D' of the DFA D where q_0 also is final:

	$\delta_{D'}$	0	1
$\rightarrow *$	q_0	q_1	q_0
	q_1	q_1	q_2
$*$	q_2	q_2	q_2

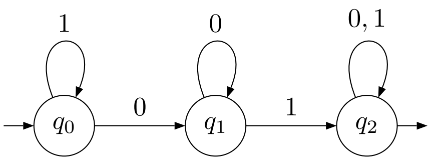
Another way to represent a DFA is through a *transition diagram*. The transition diagram for the DFA D is:



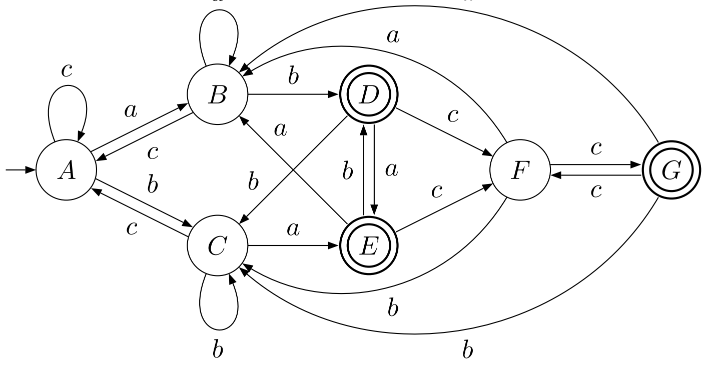
The initial state is identified by an incoming arrow. Final states are drawn with a double outline. If $\delta(q, x) = q'$ then there is an arrow from state q to q' that is labelled x. For another example, here is the transition diagram for the DFA D':



An alternative to the double outline for a final state is to use an outgoing arrow. Using that convention, the transition diagram for the DFA D is:



Here is an example of a larger DFA over the alphabet $\Sigma = \{a, b, c\}$ represented by a transition diagram:



The states are named by capital letters this time for a bit of variation: $Q = \{A, B, C, D, E, F, G\}$. While it is common to use symbols q_i, $i \in \mathbb{N}$ to name states, we can pick any names we like. Another common choice is to use natural numbers; i.e., $Q \subset \mathbb{N} \wedge Q$ is finite.

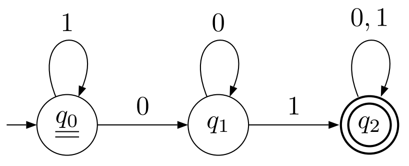
The representation of the above DFA as a transition table is:

	δ	a	b	c
$\rightarrow$	A	B	C	A
	B	B	D	A
	C	E	C	A
$*$	D	E	C	F
$*$	E	B	D	F
	F	B	C	G
$*$	G	B	C	F

3.1.2 The language of a DFA

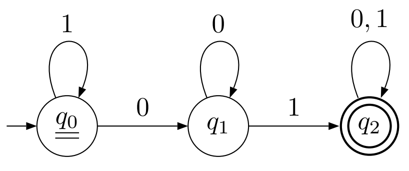
We will now discuss how a DFA accepts or rejects words over its alphabet of input symbols. The set of words accepted by a DFA A is called the *language* $L(A)$ of the DFA. Thus, for a DFA A with alphabet Σ, $L(A) \subseteq \Sigma^*$.

To determine whether a word $w \in L(A)$, the machine starts in its initial state. Taking the DFA D above as an example, it would start in state q_0. We indicate the state of a DFA by underlining the state name:

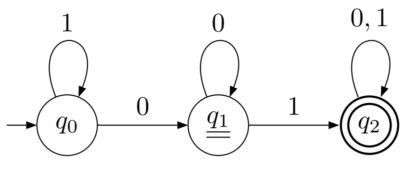


Then, whenever an input symbol is read from w, the machine transitions to a new state by following the edge labelled with this symbol. Once all symbols in the input word w have been read, the word is *accepted* if the state is final, meaning $w \in L(A)$, otherwise the word is *rejected*, meaning $w \notin L(A)$.

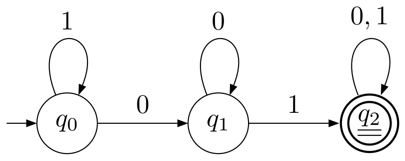
To continue with the example, suppose $w = 101$. The machine D would thus first read 1 and transition to a new state by following the edge labelled 1. As that edge in this case forms a loop back to state q_0, the machine D would transition back into state q_0:



The machine would then read 0 and transition to state q_1 by following the edge labelled 0. We indicate this by moving the mark along that edge to q_1:



Finally, the machine would read the last 1 in the input word, moving to q_2:



As q_2 is a final state, the DFA D accepts the word $w = 101$, meaning $101 \in L(D)$. In the same way, we can determine that $0 \notin L(D)$, $110 \notin L(D)$, but $011 \in L(D)$. Verify this. Indeed, a little bit of thought reveals that

$$L(D) = \{w \mid w \text{ contains the substring } 01\}$$

To make the notion of the language of a DFA precise, we now give a formal definition of $L(A)$. First we define the *extended transition function* $\hat{\delta} \in Q \times \Sigma^* \to Q$. Intuitively, $\hat{\delta}(q, w) = q'$ if the machine starting from state q ends up in state q' when reading the word w. Formally, $\hat{\delta}$ is defined by primitive recursion:

$$\hat{\delta}(q, \epsilon) = q \tag{1}$$
$$\hat{\delta}(q, xw) = \hat{\delta}(\delta(q, x), w) \tag{2}$$

where $x \in \Sigma$ and $w \in \Sigma^*$. Thus, xw stands for a nonempty word the first symbol of which is x and the rest of which is w. For example, if $xw = 010$, then $x = 0$ and $w = 10$. Note that w may be empty; e.g., if $xw = 0$, then $x = 0$ and $w = \epsilon$.

As an example, we calculate $\hat{\delta}_D(q_0, 101) = q_1$:

$$\begin{array}{rcll}
\hat{\delta}_D(q_0, 101) & = & \hat{\delta}_D(\delta_D(q_0, 1), 01) & \text{by } (2) \\
& = & \hat{\delta}_D(q_0, 01) & \text{because } \delta_D(q_0, 1) = q_0 \\
& = & \hat{\delta}_D(\delta_D(q_0, 0), 1) & \text{by } (2) \\
& = & \hat{\delta}_D(q_1, 1) & \text{because } \delta_D(q_0, 0) = q_1 \\
& = & \hat{\delta}_D(\delta_D(q_1, 1), \epsilon) & \text{by } (2) \\
& = & \hat{\delta}_D(q_2, \epsilon) & \text{because } \delta_D(q_1, 1) = q_2 \\
& = & q_2 & \text{by } (1)
\end{array}$$

Using the extended transition function $\hat{\delta}$, we define the language $L(A)$ of a DFA A formally:

$$L(A) = \{w \mid \hat{\delta}(q_0, w) \in F\}$$

Returning to our example, we thus have that $101 \in L(D)$ because $\hat{\delta}_D(q_0, 101) = q_2$ and $q_2 \in F_D$.

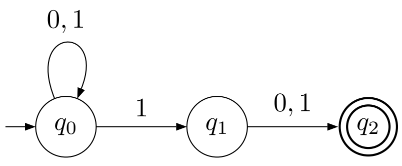
3.2 Nondeterministic finite automata

3.2.1 What is an NFA?

Nondeterministic finite automata (NFA) have transition functions that map a given state and an input symbol to zero or more successor states. We can think of this as the machine having a "choice" whenever there are two or more possible transitions from a state on an input symbol. In this presentation, we will further allow an NFA to have more than one initial state[5]. Again, we can think of this as the machine having a "choice" of where to start. An NFA accepts a word w if there is at least one *possible* way to get from one of the initial states to one of the final states along edges labelled with the symbols of w in order.

It is important to note that although an NFA has a nondetermistic transition function, it can always be determined whether or not a word belongs to its language. Indeed, we shall see that every NFA can be translated into an DFA that accepts the same language.

Here is an example of an NFA C that accepts all words over $\Sigma = \{0, 1\}$ such that the symbol before the last is 1:



A *nondeterministic finite automaton* (NFA) $A = (Q, \Sigma, \delta, S, F)$ is given by:

1. A finite set of *states* Q,
2. A finite set of input symbols, the *alphabet*, Σ,
3. A *transition function* $\delta \in Q \times \Sigma \to \mathcal{P}(Q)$,
4. A set of *initial states* $S \subseteq Q$,
5. A set of *final* (or *accepting*) states $F \subseteq Q$.

Thus, in contrast to a DFA, an NFA may have many initial states, not just one, and its transition function maps a state and an input symbol to a set of possible successor states, not just a single state. As an example we have that

$$C = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta_C, \{q_0\}, \{q2\})$$

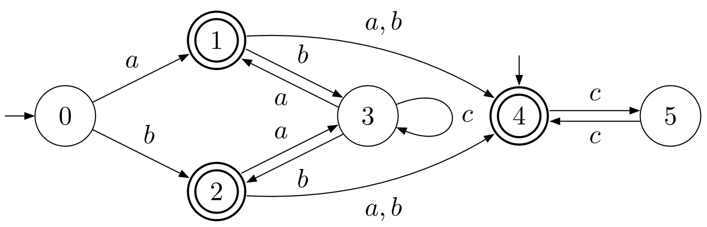
where δ_C is given by

δ_C	0	1
$\to\ q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
$*\ q_2$	$\emptyset$	$\emptyset$

Note that the entries in the table are *sets* of states, and that these sets may be empty ($\emptyset$), here exemplified by the entries for state q_2. Again, the (in this case only) initial state has been marked with $\to$ and the (in this case only) final state marked with $*$ to make this a self-contained representation of the NFA.

Here is another example of an NFA, this time over the alphabet $\Sigma = \{a, b, c\}$ and with states $Q = \{0, 1, 2, 3, 4, 5\} \subset \mathbb{N}$:

[5] Note that we diverge slightly from the definition in the book [HMU01], which uses a single initial state instead of a set of initial states. Permitting more than one initial state allows us to avoid introducing ϵ-NFAs (see [HMU01], section 2.5).



The transition table for this NFA is:

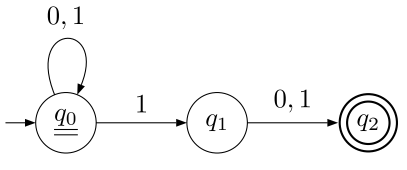
δ	a	b	c
$\rightarrow$ 0	$\{1\}$	$\{2\}$	$\emptyset$
$*$ 1	$\{4\}$	$\{3,4\}$	$\emptyset$
$*$ 2	$\{3,4\}$	$\{4\}$	$\emptyset$
3	$\{1\}$	$\{2\}$	$\{3\}$
$\rightarrow *$ 4	$\emptyset$	$\emptyset$	$\{5\}$
5	$\emptyset$	$\emptyset$	$\{4\}$

Note that this NFA has multiple initial states, multiple final states, one initial state that also is final, and that there in some cases are no possible successor states and in other cases more than one.

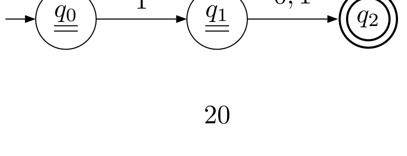
3.2.2 The language accepted by an NFA

To see whether a word $w \in \Sigma^*$ is accepted by an NFA A, we have to consider *all* possible states the machine could be in after having read a sequence of input symbols. Initially, an NFA can be in any of its initial states. Each time an input symbol is read, all successor states on the read symbol for each current possible state become the new possible states. After having read a complete word w, if at least one of the possible states is final (accepting), then that word is accepted, meaning $w \in L(A)$, otherwise it is rejected, meaning $w \notin L(A)$.

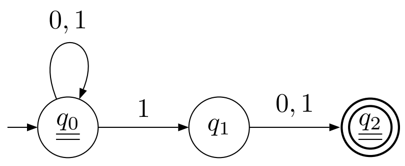
We will illustrate by showing how the NFA C rejects the word 100. We will again mark the current states of the NFA by underlining the state names, but this time there may be more than one marked state at once. Initially, as q_0 is the only initial state, we would have:



Each time when we read a symbol we look at all the marked states. We remove the old markers and put markers at all the states that are *reachable* via an arrow marked with the current input symbol. This may include one or more states that were marked previously. It may also be the case that no states are reachable, in which case all marks are removed and the word rejected (as it no longer is possible to reach any final states). In our example, after reading 1, there would be two marked states as there are two arrows from q_0 labelled 1:

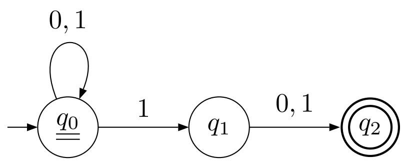


After reading 0, the next symbol in the word 100, there would still be two marked states as the machine on input 0 can reach q_0 from q_0 and q_2 from q_1:



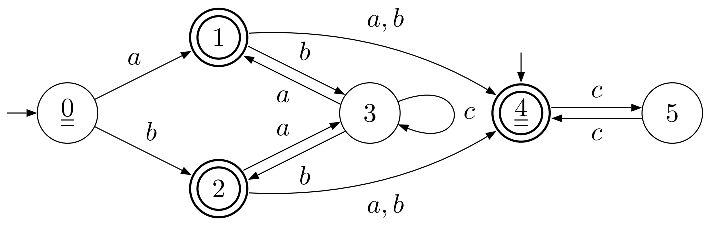
Note that one of the marked states is a final (accepting) state, meaning the word read so far (10) is accepted by the NFA.

However, there is one symbol left in our example word 100, and after having read the final 0, the final state would no longer be marked because it cannot be reached from any of the marked states:

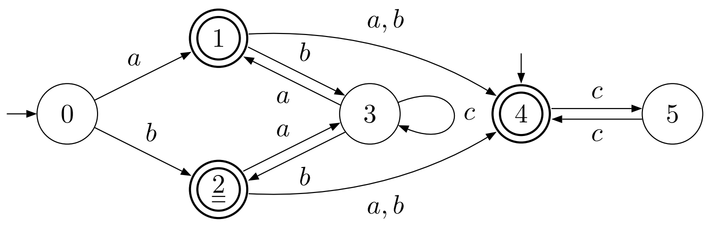


The NFA C thus rejects the word 100.

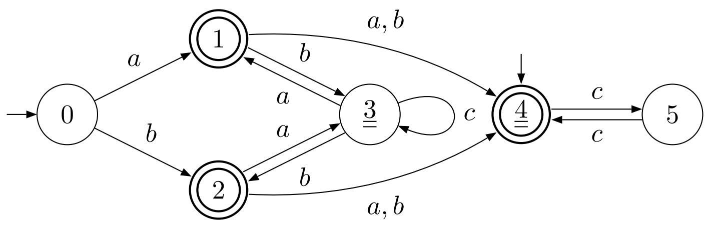
For another example, consider the NFA at the end of section 3.2.1. Convince yourself that you understand how this NFA accepts the words ϵ, *abcc*, *abcca*, and rejects *abccaac*. We illustrate by tracing its operation on the word *bacac*. We start by marking all initial states. Then it is just a matter of systematically exploring all possibilities:



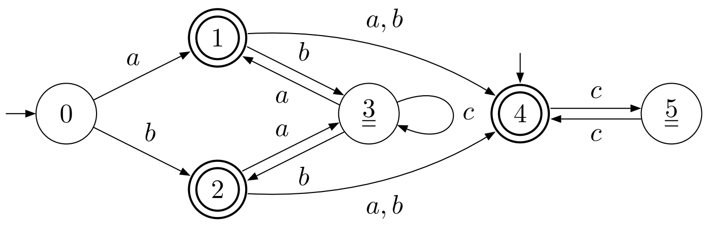
After reading b:



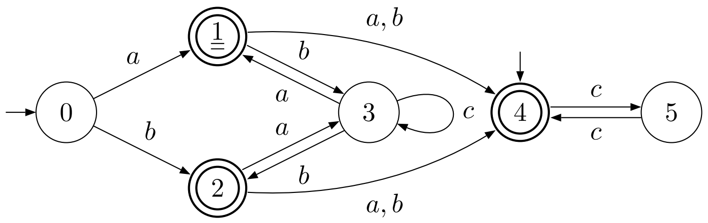
After reading a:



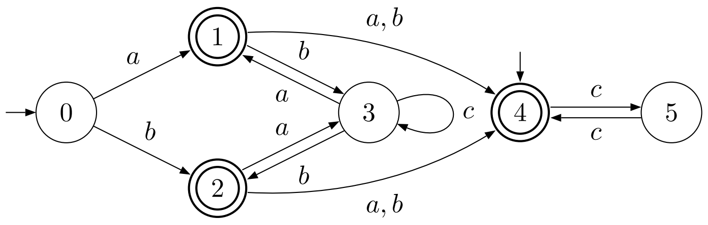
After reading c:



After reading a:



After reading c:



The machine thus rejects *bacac* as no final state is marked. In fact, as there are *no* marked states left at all, this shows that this NFA will reject *all* words that start *bacac*. Can you find other such prefixes?

To define the *extended transition function* $\hat{\delta}$ for NFAs we use a generalisation of the union operation $\cup$ on sets over a (finite) set of sets:

$$\bigcup\{A_1, A_2, \ldots A_n\} = A_1 \cup A_2 \cup \cdots \cup A_n$$

In the special cases of the empty set of sets and a one element set of sets:

$$\bigcup \emptyset = \emptyset \qquad \bigcup\{A\} = A$$

As an example

$$\bigcup\{\{1\}, \{2,3\}, \{1,3\}\} = \{1\} \cup \{2,3\} \cup \{1,3\} = \{1,2,3\}$$

Alternatively, we can define $\bigcup$ by comprehension, which also extends the operation to infinite sets of sets (although we don't need this here):

$$\bigcup B = \{x \mid \exists A \in B . x \in A\}$$

We define $\hat{\delta} \in \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$ such that $\hat{\delta}(P, w)$ is the set of states that are reachable from one of the states in P on the word w:

$$\begin{aligned} \hat{\delta}(P, \epsilon) &= P && (3) \\ \hat{\delta}(P, xw) &= \hat{\delta}(\bigcup\{\delta(q, x) \mid q \in P\}, w) && (4) \end{aligned}$$

where $x \in \Sigma$ and $w \in \Sigma^*$. Intuitively, if P are the possible states, then $\hat{\delta}(P, w)$ are the possible states after having read a word w.

To illustrate, we calculate $\hat{\delta}_C(q_0, 100)$:

$$\begin{aligned} \hat{\delta}_C(\{q_0\}, 100) &= \hat{\delta}_C(\bigcup\{\delta_C(q, 1) \mid q \in \{q_0\}\}, 00) && \text{by (4)} \\ &= \hat{\delta}_C(\delta_C(q_0, 1), 00) \\ &= \hat{\delta}_C(\{q_0, q_1\}, 00) \\ &= \hat{\delta}_C(\bigcup\{\delta_C(q, 0) \mid q \in \{q_0, q_1\}\}, 0) && \text{by (4)} \\ &= \hat{\delta}_C(\delta_C(q_0, 0) \cup \delta_C(q_1, 0), 0) \\ &= \hat{\delta}_C(\{q_0\} \cup \{q_2\}, 0) \\ &= \hat{\delta}_C(\{q_0, q_2\}, 0) \\ &= \hat{\delta}_C(\bigcup\{\delta_C(q, 0) \mid q \in \{q_0, q_2\}\}, \epsilon) && \text{by (4)} \\ &= \hat{\delta}_C(\delta_C(q_0, 0) \cup \delta_C(q_2, 0), 0) \\ &= \hat{\delta}_C(\{q_0\} \cup \emptyset, \epsilon) \\ &= \{q_0\} && \text{by (3)} \end{aligned}$$

Of course, we already knew this from the worked example above illustrating how the NFA C rejects 100. Make sure you see how the marked states after each step coincides with the set of possible states in the calculation.

The language of an NFA can now be defined using $\hat{\delta}$:

$$L(A) = \{w \mid \hat{\delta}(S, w) \cap F \neq \emptyset\}$$

Thus, $100 \notin L(C)$ because

$$\hat{\delta}_C(S_C, 100) \cap F_C = \hat{\delta}_C(\{q_0\}, 100) \cap \{q_2\} = \{q_0\} \cap \{q_2\} = \emptyset$$

3.2.3 The subset construction

DFAs can be viewed as a special case of NFAs; i.e., those for which the there is precisely one start state ($S = \{q_0\}$) and for which the transition function always returns singleton (one-element) sets ($\delta(q, x) = \{q'\}$ for all $q \in Q$ and $x \in \Sigma$).

The opposite is also true, however: NFAs are really just DFAs "in disguise". We show this by for a given NFA systematically constructing an *equivalent* DFA; i.e., a DFA that accepts the same language as the given NFA. NFAs are thus no more powerful than DFAs; i.e., NFAs cannot describe more languages than DFAs. However, in some cases, NFAs need a lot fewer states than the corresponding DFA, and they may be easier to construct in the first place.

The subset construction: Given an NFA $A = (Q, \Sigma, \delta, S, F)$ we construct the equivalet DFA:

$$D(A) = (\mathcal{P}(Q), \Sigma, \delta_{D(A)}, S, F_{D(A)})$$

where

$$\begin{aligned} \delta_{D(A)}(P, x) &= \bigcup\{\delta(q, x) \mid q \in P\} && (5) \\ F_{D(A)} &= \{P \in \mathcal{P}(Q) \mid P \cap F \neq \emptyset\} && (6) \end{aligned}$$

The basic idea of this construction is to define a DFA whose states are *sets of NFA states*. A set of possible NFA states thus becomes a single DFA state. The DFA transition function is given by considering all reachable NFA states for each of the current possible NFA states for each input symbol. The resulting set of possible NFA states is again just a single DFA state. A DFA state is final if that set that contains at least one final NFA state.

As an example, let us construct a DFA $D(C)$ equivalent to C above:

$$D(C) = (\mathcal{P}(\{q_0, q_1, q_2\}, \{0, 1\}, \delta_{D(C)}, \{q_0\}, F_{D(C)})$$

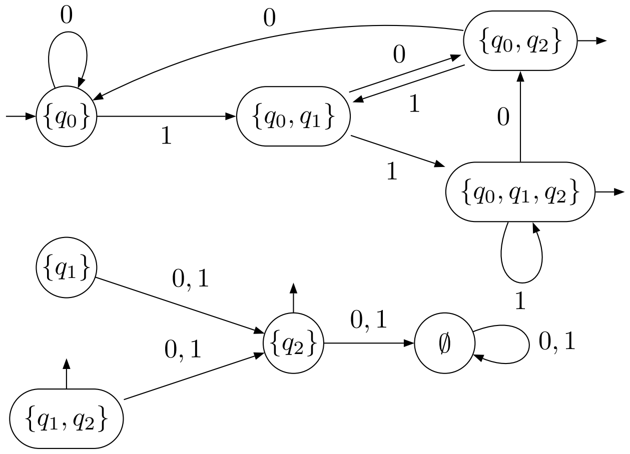
where $\delta_{D(C)}$ is given by:

	$\delta_{D(C)}$	0	1
	$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow$	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
	$\{q_1\}$	$\{q_2\}$	$\{q_2\}$
$*$	$\{q_2\}$	$\emptyset$	$\emptyset$
	$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$*$	$\{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$
$*$	$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$
$*$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

and $F_{D(C)}$ (all the states marked with $*$ above) by:

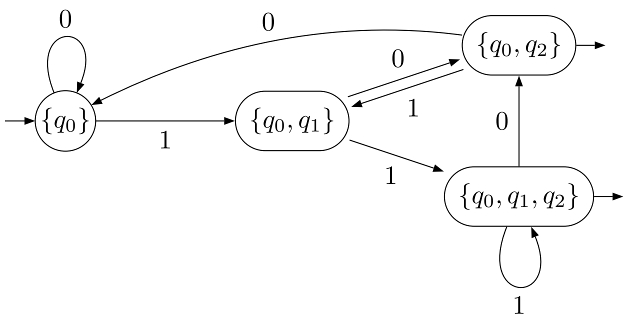
$$F_{D(C)} = \{\{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$$

The transition diagram is:



Accepting states have been marked by outgoing arrows.

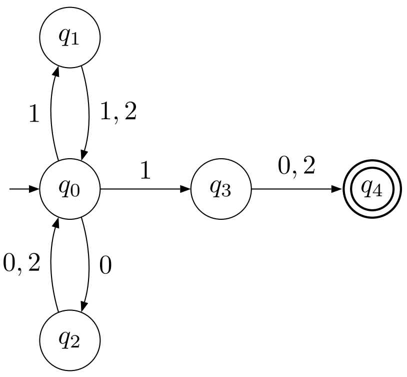
Note that some of the states ($\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}$) cannot be reached from the initial state. This means that they can be omitted without changing the language. We thus obtain the following automaton:



It is possible to avoid having to perform calculations for states that cannot be reached by carrying out the subset construction in a “demand driven” way. The idea is to start from the initial DFA state, which is just the set of initial NFA states S, and then only consider the DFA states (subsets of NFA states) that appear during the course of the calculations. We illustrate this approach by an example. Consider the following NFA N:

$$N = (Q_N = \{q_0, q_1.q_2, q_3, q_4\},\ \Sigma_N = \{0,\ 1,\ 2\},\ \delta_N,\ S_N = \{q_0\},\ F_N = \{q_4\})$$

where δ_N is given by the transition diagram:



Note that N has 5 states which means that the DFA $D(N)$ has $|\mathcal{P}(Q_N)| = 2^5 =$ 32 states. However, as we will see, only a handful of those 32 states can actually be reached from the initial state S_N of $D(N)$. We would thus waste quite a bit of effort if we were to tabulate all of them.

We start from $S_N = \{q_0\}$, the set of start states of N, and we we compute $\bigcup\{\delta(q, x) \mid q \in S_N\}$ for each $x \in \Sigma_N$ (equation (5)). In this case we get:

$\delta_{D(N)}$	0	1	2
$\rightarrow\ \{q_0\}$	$\{q_2\}$	$\{q_1, q_3\}$	$\emptyset$

Whenever we encounter§q a state $P \subseteq Q$ of $D(N)$ that has not been considered before, we add P to the table, marking any final states as such. In this case, three new DFA states emerge ($\{q_2\}$, $\{q_1, q_3\}$, and $\emptyset$), none of which is final:

$\delta_{D(N)}$	0	1	2
$\rightarrow\ \{q_0\}$	$\{q_2\}$	$\{q_1, q_3\}$	$\emptyset$
$\{q_2\}$			
$\{q_1, q_3\}$			
$\emptyset$			

We then proceed to tabulate $\delta_{D(N)}$ for each of the new states for each $x \in \Sigma$, adding any further new states to the table:

$\delta_{D(N)}$	0	1	2
$\rightarrow$ $\{q_0\}$	$\{q_2\}$	$\{q_1, q_3\}$	$\emptyset$
$\{q_2\}$	$\{q_0\}$	$\emptyset$	$\{q_0\}$
$\{q_1, q_3\}$	$\emptyset \cup \{q_4\} = \{q_4\}$	$\{q_0\} \cup \emptyset = \{q_0\}$	$\{q_0\} \cup \{q_4\} = \{q_0, q_4\}$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

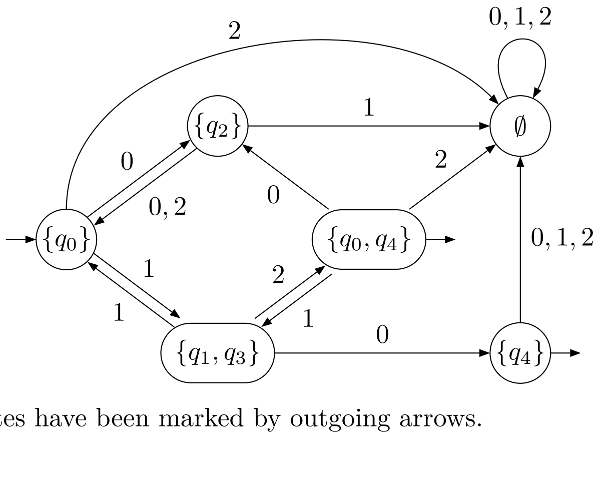
Here, two new states emerge ($\{q_4\}$ and $\{q_0, q_4\}$), both final (because $\{q_4\} \cap F_N \neq \emptyset$ and $\{q_0, q_4\} \cap F_N \neq \emptyset$):

$\delta_{D(N)}$	0	1	2
$\rightarrow$ $\{q_0\}$	$\{q_2\}$	$\{q_1, q_3\}$	$\emptyset$
$\{q_2\}$	$\{q_0\}$	$\emptyset$	$\{q_0\}$
$\{q_1, q_3\}$	$\emptyset \cup \{q_4\} = \{q_4\}$	$\{q_0\} \cup \emptyset = \{q_0\}$	$\{q_0\} \cup \{q_4\} = \{q_0, q_4\}$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$*$ $\{q_4\}$			
$*$ $\{q_0, q_4\}$			

This process is repeated until no new states emerges. Tabulating for the last two new states reveals that no further states emerge in this case and we are thus done, having only had to tabulate for 6 reachable out of the 32 DFA states:

$\delta_{D(N)}$	0	1	2
$\rightarrow$ $\{q_0\}$	$\{q_2\}$	$\{q_1, q_3\}$	$\emptyset$
$\{q_2\}$	$\{q_0\}$	$\emptyset$	$\{q_0\}$
$\{q_1, q_3\}$	$\emptyset \cup \{q_4\} = \{q_4\}$	$\{q_0\} \cup \emptyset = \{q_0\}$	$\{q_0\} \cup \{q_4\} = \{q_0, q_4\}$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$*$ $\{q_4\}$	$\emptyset$	$\emptyset$	$\emptyset$
$*$ $\{q_0, q_4\}$	$\{q_2\} \cup \emptyset = \{q_2\}$	$\{q_1, q_3\} \cup \emptyset = \{q_1, q_3\}$	$\emptyset \cup \emptyset = \emptyset$

After double checking that we have not forgotten to mark any final states, we can draw the transition diagram for $D(N)$:



Accepting states have been marked by outgoing arrows.

3.2.4 Correctness of the subset construction

We still have to convince ourselves that the subset construction actually works; i.e., that for a given NFA A it really is the case that $L(A) = L(D(A))$. We start by proving the following lemma, which says that the extended transition functions coincide:

Lemma 3.1

$$\hat{\delta}_{D(A)}(P, w) = \hat{\delta}_A(P, w)$$

The result of both functions is a set of states of the NFA A: for the left-hand side because the extended transition function on NFAs returns a set of states, and for the right-hand side because the states of $D(A)$ are sets of states of A.

Proof. We show this by *induction* over the length of the word w, $|w|$.

$|w| = 0$ Then $w = \epsilon$ and we have

$$\begin{array}{rcll} \hat{\delta}_{D(A)}(P, \epsilon) & = & P & \text{by (1)} \\ & = & \hat{\delta}_A(P, \epsilon) & \text{by (3)} \end{array}$$

$|w| = n + 1$ Then $w = xv$ with $|v| = n$.

$$\begin{array}{rcll} \hat{\delta}_{D(A)}(P, xv) & = & \hat{\delta}_{D(A)}(\delta_{D(A)}(P, x), v) & \text{by (2)} \\ & = & \hat{\delta}_A(\delta_{D(A)}(P, x), v) & \text{ind.hyp.} \\ & = & \hat{\delta}_A(\bigcup\{\delta_A(q, x) \mid q \in P\}, v) & \text{by (5)} \\ & = & \hat{\delta}_A(P, xv) & \text{by (4)} \end{array}$$

□

We can now use the lemma to show

Theorem 3.2

$$L(A) = L(D(A))$$

Proof.

$$\begin{array}{c} w \in L(A) \\ \iff \quad \text{Definition of } L(A) \text{ for NFAs} \\ \hat{\delta}_A(S, w) \cap F \neq \emptyset \\ \iff \quad \text{Lemma 3.1} \\ \hat{\delta}_{D(A)}(S, w) \cap F \neq \emptyset \\ \iff \quad \text{Definition of } F_{D(A)} \\ \hat{\delta}_{D(A)}(S, w) \in F_{D(A)} \\ \iff \quad \text{Definition of } L(A) \text{ for DFAs} \\ w \in L_{D(A)} \end{array}$$

□

Corollary 3.3 *NFAs and DFAs recognise the same class of languages.*

Proof. We have noticed that DFAs are just a special case of NFAs. On the other hand the subset construction introduced above shows that for every NFA we can find a DFA that recognises the same language. □