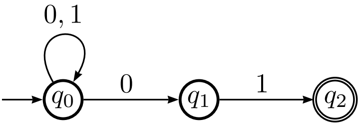
Non-deterministic finite automata

A **non-deterministic finite automaton** (**NFA**) has the same definition as a DFA except that δ returns a set of states instead of one state.

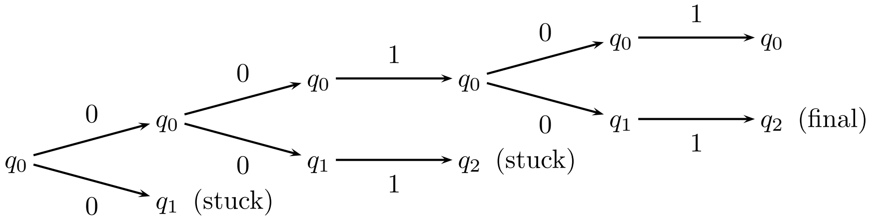
Consider



There are two out-going edges from state q_0 which are labeled 0, hence two states can be reached when 0 is input: q_0 (loop) and q_1.
This NFA recognises the language of words on the binary alphabet whose suffix is 01.

Non-deterministic finite automata (cont)

Before describing formally what is a recognisable language by a NFA, let us consider as an example the previous NFA and the input 00101.
Let us represent each transition for this input by an edge in a tree where nodes are states of the NFA.



NFA/Formal definitions

A NFA is represented essentially like a DFA: $\mathcal{N} = (Q_N, \Sigma, \delta_N, q_0, F_N)$ where the names have the same interpretation as for DFA, except δ_N which returns a subset of Q — not an element of Q.

For example, the NFA whose transition diagram is page 141 can be specified formally as

$$\mathcal{N} = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta_N, q_0, \{q_2\})$$

where the transition function δ_N is given by the transition table:

$\mathcal{N}$	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$\# q_2$	$\varnothing$	$\varnothing$

NFA/Formal definitions (cont)

Note that, in the transition table of a NFA, all the cells are filled: there is no transition between two states if and only if the corresponding cell contains $\varnothing$.

In case of a DFA, the cell would remain empty.

It is common also to set that in case of the empty word input, ε, both for the DFA and NFA, the state remains the same:

- for DFA: $\forall q \in Q.\delta_D(q, \varepsilon) = q$
- for NFA: $\forall q \in Q.\delta_N(q, \varepsilon) = \{q\}$

NFA/Formal definitions (cont)

As we did for the DFAs, we can *extend the transition function* δ_N to accept words and not just letters (labels). The extended function is noted $\hat{\delta}_N$ and defined as

- for all state $q \in Q$, let $\hat{\delta}_N(q, \varepsilon) = \{q\}$
- for all state $q \ \in Q$, all words $w \in \Sigma^*$, all input $a \in \Sigma$, let

$$\hat{\delta}_N(q, wa) = \bigcup_{q' \in \hat{\delta}_N(q,w)} \delta_N(q', a)$$

The language $L(\mathcal{N})$ recognised by a NFA $\mathcal{N}$ is defined as

$$L(\mathcal{N}) = \{w \in \Sigma^* \mid \hat{\delta}_N(q_0, w) \cap F \neq \varnothing\}$$

which means that the processing of the input stops successfully as soon as **at least one current state belongs to** F.

NFA/Example

Let us use $\hat{\delta}_N$ to describe the processing of the input 00101 by the NFA page 141:

1. $\hat{\delta}_N(q_0, \varepsilon) = q_0$
2. $\hat{\delta}_N(q_0, 0) = \delta_N(q_0, 0) = \{q_0, q_1\}$
3. $\hat{\delta}_N(q_0, 00) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0) = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$
4. $\hat{\delta}_N(q_0, 001) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
5. $\hat{\delta}_N(q_0, 0010) = \delta_N(q_0, 0) \cup \delta_N(q_2, 0) = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$
6. $\hat{\delta}_N(q_0, 00101) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\} \ni q_2$

Because q_2 is a final state, actually $F = \{q_2\}$, we get $\hat{\delta}_N(q_0, 00101) \cap F \neq \varnothing$ thus the string 00101 is recognised by the NFA.