

Technical Definitions for Automata and a Relationship

Deterministic Finite Automata (DFA) and Nondeterministic Finite Automata (NFA) should first be encountered in a casual way, by means of diagrams, exercises in processing strings, discovering the languages of small automata, constructing small automata that have a prescribed language, and so forth.

However, once this has been accomplished, it is important to "bite the bullet" and carefully examine some very technical (especially for the NFA!) mathematical definitions for these structures. We have already done so in class for the DFA. I will review this here, and will then lay out the corresponding technical details for an NFA. Make sure you follow the details, although this will take a very concerted effort in the case of NFAs. As I said, wait until you have an intuitive grasp of an NFA before sweating its technical definition.

Definition of a DFA. A DFA is an ordered 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set whose elements are called "states"
- Σ is a finite set whose elements are called "symbols"
- q_0 is an element of Q , called the "initial state"
- F is a subset of Q whose elements are called "accepting states"
- δ is a function called the "transition function" whose domain is $Q \times \Sigma$, and whose codomain is Q .

Definition of a DFA's extended transition function. Given such a DFA $M = (Q, \Sigma, \delta, q_0, F)$, it is handy to define another function $\hat{\delta}$ called the *extended transition function*. Its domain is $Q \times \Sigma^*$, where you should recall that Σ^* is the set of strings formed from the symbols in Σ . The codomain of $\hat{\delta}$ is Q , and this function is defined recursively as follows, where q denotes an arbitrary element of Q :

- With ϵ denoting the empty string, define $\hat{\delta}(q, \epsilon) = q$
- For a symbol $\sigma \in \Sigma$, define $\hat{\delta}(q, \sigma) = \delta(q, \sigma)$
- For a string $\sigma \in \Sigma^*$ and a symbol $\tau \in \Sigma$, define $\hat{\delta}(q, \sigma\tau) = \delta(\hat{\delta}(q, \sigma), \tau)$

Definition of the language of a DFA. The *language* of a DFA $M = (Q, \Sigma, \delta, q_0, F)$ is the set of strings

$$\{\sigma \in \Sigma^* \mid \hat{\delta}(q_0, \sigma) \in F\}.$$

The strings in this set are said to be *accepted* by M .

Now let's turn to the corresponding definitions for a NFA. As I already warned, this is quite a bit more complicated. This is especially true due to the fact that we will follow Sipser's lead and really define a somewhat more complicated thing that is most people call an ϵ -NFA, but which Sipser just calls an NFA. Here goes.

Definition of an NFA. A NFA is an ordered 5-tuple $N = (Q, \Sigma, \delta, q_0, F)$, where

- Q is a finite set whose elements are called "states"
- Σ is a finite set whose elements are called "symbols"
- q_0 is an element of Q , called the "initial state"
- F is a subset of Q whose elements are called "accepting states"
- δ is a function called the "transition function" whose domain is $Q \times \Sigma_\epsilon$, where Σ_ϵ just means the set $\Sigma \cup \{\epsilon\}$. The codomain of δ is the power set $\wp(Q)$.

(continue on back)

Definition of an NFA's extended transition function. Given such an NFA $N = (Q, \Sigma, \delta, q_0, F)$, we can define an *extended transition function* $\hat{\delta}$, analogous to a DFA's extended transition function. However, it is technically more difficult to do so, as we will now see. Again the function is defined recursively, and q will denote an arbitrary element of Q .

- $\hat{\delta}(q, \epsilon)$ is defined to be the smallest possible subset of Q with these properties:
 - ◊ $q \in \hat{\delta}(q, \epsilon)$
 - ◊ given any $q' \in \hat{\delta}(q, \epsilon)$ and any $q'' \in \hat{\delta}(q', \epsilon)$, it is required that $q'' \in \hat{\delta}(q, \epsilon)$
- For a symbol $\sigma \in \Sigma$, define

$$\hat{\delta}(q, \sigma) = \bigcup_{q'' \in \bigcup_{q' \in \hat{\delta}(q, \epsilon)} \delta(q', \sigma)} \hat{\delta}(q'', \epsilon)$$

- For a string $\sigma \in \Sigma^*$ and a symbol $\tau \in \Sigma$, define

$$\hat{\delta}(q, \sigma\tau) = \bigcup_{q' \in \hat{\delta}(q, \sigma)} \hat{\delta}(q', \tau)$$

Definition of the language of an NFA. Well at least the definition of the *language* of an NFA is now easy. It is just the set

$$\{\sigma \in \Sigma^* \mid \hat{\delta}(q_0, \sigma) \cap F \neq \phi\}.$$

A couple more definitions. We say that a language (*i.e.* a subset of Σ^*) is *regular* if it is the language of some DFA. We say that two automata (DFA or NFA) are *equivalent* if they have the same language.

Sipser's Theorem 1.19. Every NFA is equivalent to some DFA. Therefore, every NFA's language is regular.

Here is the basis for a constructive proof of this theorem. Consider any NFA $N = (Q, \Sigma, \delta, q_0, F)$. Define $Q' = \wp(Q)$. Remember that this means that an element of Q' is a subset of Q . Also define $q'_0 = \{q_0\} \in Q'$. Also define $F' = \{S \subseteq Q \mid S \cap F \neq \phi\}$. Also define a function δ' from $Q' \times \Sigma$ to Q' via

$$\delta'(S, \sigma) = \bigcup_{q \in S} \hat{\delta}(q, \sigma),$$

where $S \in Q'$ and $\sigma \in \Sigma$. Define $M = (Q', \Sigma, \delta', q'_0, F')$. Now carefully notice that M is a DFA. Also notice that

$$\hat{\delta}'(S, \sigma) = \bigcup_{q \in S} \hat{\delta}(q, \sigma)$$

for all $S \in Q'$ and all strings $\sigma \in \Sigma^*$. This requires some careful thinking. Given the definitions of q'_0 and F' , and the definitions of languages, it can now be checked that M and N have the same languages, *i.e.* accept the same strings. So M and N are equivalent. (Note: building M based on N is often called the "subset construction".)