Discussion 3: Non-deterministic finite automatas

February 3, 2009

Purpose: This discussion demonstrates a few simple NFAs, and how to formally define a NFA. We also demonstrate that complementing a NFA is a tricky business.

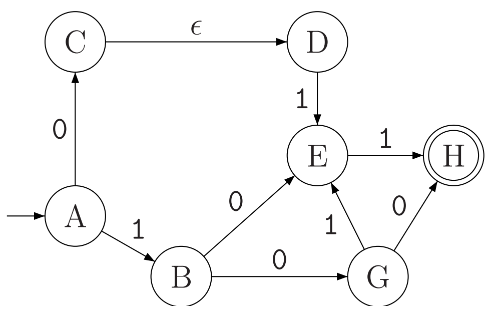
Questions on homework 2?

Any questions? Complaints, etc?

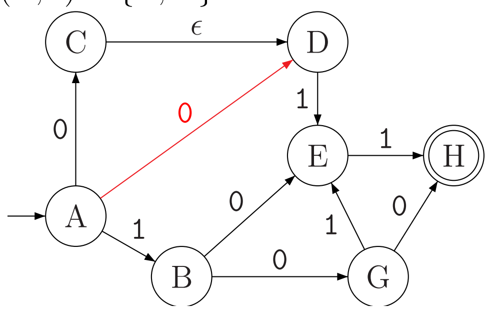
1 Non-determinism with finite number of states

1.1 Formal description of NFA

M_1 :



In the above NFA, we have $\delta(A, 0) = \{C\}$. Despite the ϵ-transition from C to D. As such, $\delta(A, 0) \neq \{C, D\}$. If $\delta(A, 0) = \{C, D\}$ then the NFA is a different NFA:



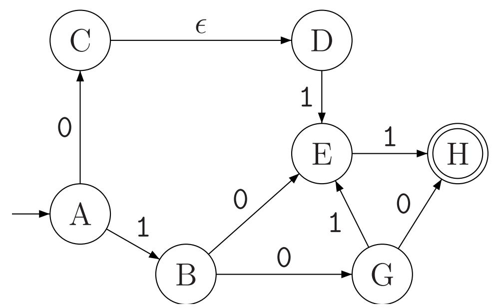
In any case, the NFA M_1 (depicted in first figure) is the 5-tuple $(Q, \Sigma, \delta, A, \mathcal{F})$, where

$$\delta : Q \times \Sigma_\epsilon \rightarrow \mathbb{P}(Q).$$

Here $\Sigma = \{0, 1\}$, $Q = \{A, B, C, D, E, G, H\}$, and $\mathcal{F} = \{H\}$.

δ	0	1	ϵ
A	$\{C\}$	$\{B\}$	$\emptyset$
B	$\{E,G\}$	$\emptyset$	$\emptyset$
C	$\emptyset$	$\emptyset$	$\{D\}$
D	$\emptyset$	$\{E\}$	$\emptyset$
E	$\emptyset$	$\{H\}$	$\emptyset$
G	$\{H\}$	$\{E\}$	$\emptyset$
H	$\emptyset$	$\emptyset$	$\emptyset$

M_1 :



1.2 Concatenating NFAs

We are given two NFAs $M = (Q, \Sigma, \delta, A, F)$ and $M' = (Q', \Sigma, \delta', A', F')$. We would like to build an NFA for the concatenated language $L(M)L(M')$.

First, we can assume that M has a single accepting state f. Indeed, we can create a new accepting state f, add it to Q, make all the states in F non-accepting, but add an ϵ-transition from them to f. Thus, we can now assume that $F = \{f\}$.

Back to our task, of constructing the concatenated NFA, we can just create an ϵ transition from f to A'. Here is the formal construction of the NFA for the concatenated language $N = \left(\mathcal{Q}, \Sigma, \widehat{\delta}, A, F'\right)$, where $\mathcal{Q} = Q \cup Q'$. As for *widehat*δ, we have that

$$\widehat{\delta}(q,x) = \begin{cases} \delta(q,x) & q \in Q \\ \delta'(q,x) & q \in Q' \\ \delta(f,\epsilon) \cup \{A'\} & q = f \text{ and } x = \epsilon. \end{cases}$$

Claim 1.1 *The NFA N accepts a string $w \in \Sigma^*$, if and only if there exists two strings $x, y \in \Sigma^*$, such that $w = xy$ and $x \in L(M)$ and $y \in L(M')$.*

Proof: If $x \in L(M)$ then there is an accepting trace (i.e., a sequence of states and inputs that show that x is being accepted by M, and let the sequence of states be $A = r_0, r_1, \ldots, r_\alpha$, and the corresponding input sequence be $x_1, \ldots, x_\alpha \in \Sigma_\epsilon$. Here $x = x_1 x_2 \ldots x_\alpha$ (note that some of these characters might be ϵ).

Similarly, let $A' = r'_0, r'_1, \ldots, r'_\beta$ be accepting trace of M' accepting y, with the input characters $y_1, y_2, \ldots, y_\beta \in \Sigma_\epsilon$, where $y = y_1 y_2 \ldots y_\beta$.

Note, that by our assumption $r_\alpha = f$. As such, the following is accepting trace of $w = xy$ for N:

$$r_0 \underset{x_1}{\longrightarrow} r_1 \underset{x_2}{\longrightarrow} r_2 \longrightarrow \cdots \underset{x_\alpha}{\longrightarrow} r_\alpha \underset{\epsilon}{\longrightarrow} r'_0 \underset{y_1}{\longrightarrow} r'_1 \underset{y_2}{\longrightarrow} \cdots \underset{y_\beta}{\longrightarrow} r'_\beta.$$

Indeed, its a valid trace, as can be easily verified, and $r'_\beta \in F'$ (otherwise y would not be in $L(M')$.

Similarly, given a word $W \in L(N)$, and accepting trace for it, then we can break this trace into two parts. The first part is trace before using the transition $f \rightarrow_\epsilon A'$, and the other is the rest of the trace. Clearly, if we remove this transition from the given accepting trace, we end up with two accepting traces for M and M', implying that we can break w into two strings x and y, such that $x \in L(M)$ and $y \in L(M')$. ∎

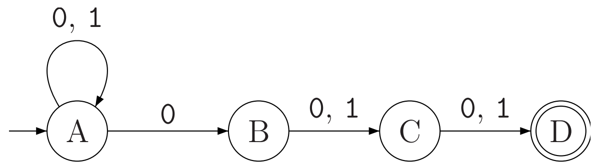
1.3 Sometimes non-determinism keeps the number of states small

Let $\Sigma = \{0, 1\}$. Remember that the smallest DFA that we built for

$$L_3 = \left\{ x \in \Sigma^* \;\middle|\; \text{the third character from the end of } x \text{ is a zero} \right\}.$$

had 8 states. Note that the following NFA does the same job, by guessing position of third character from the end of string.

M_3:



Q: Is there a language L where we have a DFA for L with smaller number of states that of any NFA for L?

A: No. Because any DFA is also a NFA.

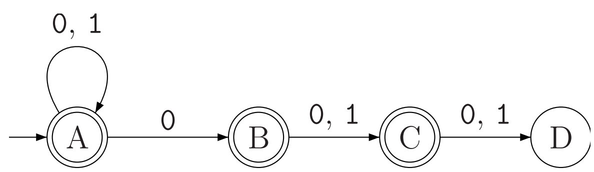
1.4 How to complement an NFA?

Given the NFA above M_3, it is natural to ask how to build a NFA for the complement language

$$\overline{L(M_3)} = \overline{L_3} = \left\{ x \in \Sigma^* \;\middle|\; \text{the third character from the end of } x \text{ is not zero} \right\}.$$

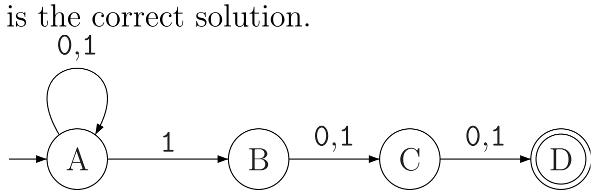
Naively, the easiest thing would be to complement the states of the NFA. We get the following NFA M_4.

M_4:



But this is of course complete and total nonsense. Indeed, the language of $L(M_4) = \Sigma^*$, which is definitely not $\overline{L_3}$. Here is the correct solution.

M_5:



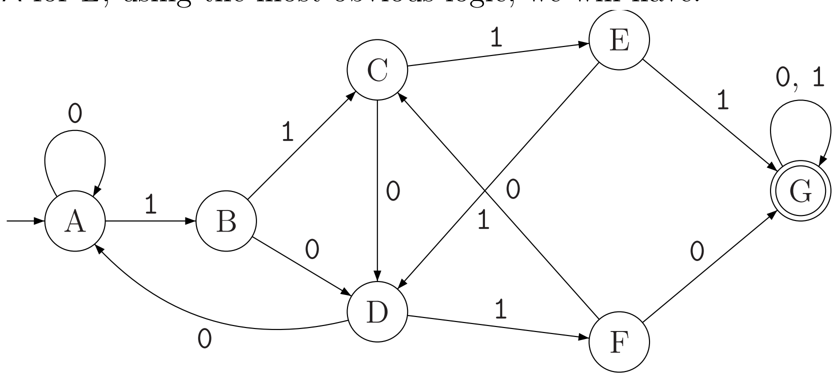
The conclusion of this tragic and sad example is that complementing a NFA is a non-trivial task (unlike DFAs where all you needed to do was to just flip the accepting/non-accepting states). So, for some tasks DFAs are better than DFAs, and vice versa.

1.5 Sometimes non-determinism keeps the design logic simple

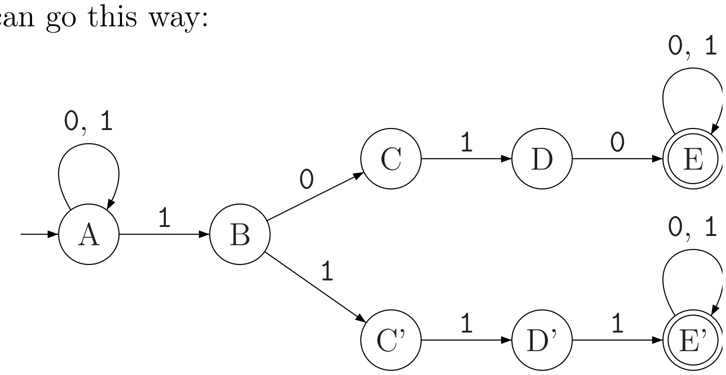
Consider the following language:

$$L = \{x : x \text{ has } 1111 \text{ or } 1010 \text{ as a substring}\}$$

Designing a DFA for L, using the most obvious logic, we will have:



With NFA we can go this way:



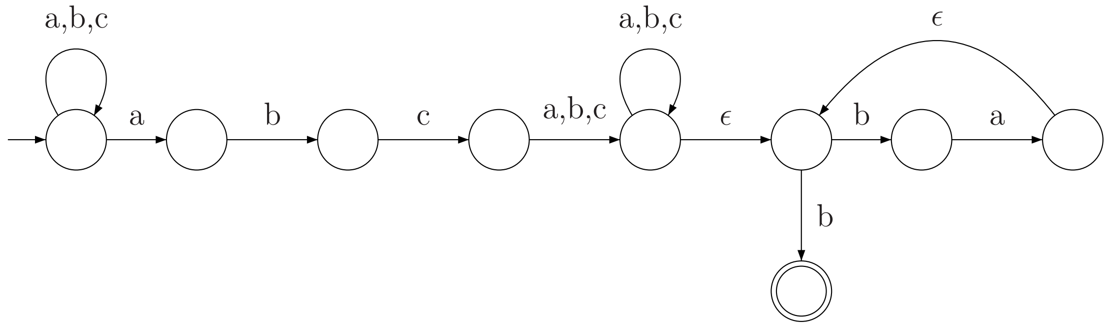
Note that the NFA approach is easily extendable to more than 2 substrings.

2 Pattern Matching

Suppose we wanted to build an NFA for the following pattern.

$$abc?(ba)^*b$$

Where ? represents a substring of length 1 or more and * represents 0 of more of the previous expression. The NFA for this pattern would be



3 Formal definition of acceptance

Recall that a finite automaton M accepts a string w means there is a sequence of states $r_0, r_1, ..., r_n$ in Q where

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, ..., n-1$ and
3. $r_n \in F$

How do we formally show a string w is accepted by M. Lets show that the (automaton on page 1) accepts the string 101.

We show that there must exists states $r_0, r_1, ..., r_3$ statisfying the above three conditions. We claim that the sequence A,B,E,G satisfies the three claims.

1. $A = q_0$
2. $\delta(A, 1) = B$
   $\delta(B, 0) = E$
   $\delta(E, 1) = G$
3. $G \in F$