CSE 322 - Introduction to Formal Methods in Computer Science
Closure Under Regular Operations

Dave Bacon
Department of Computer Science & Engineering, University of Washington

Way back when, we introduced regular operations. Remember those things where we took two languages and constructed a new language. Like, for example, $A \cup B$ and $A \cap B$ and $A \circ B$ and A^*. For the first of these we were able to show that regular languages were closed under this operation. In other words if A and B were regular, then $A \cup B$ is also regular. But when we thought about how to do this for, say $A \circ B$, it wasn't at all obvious how to get his to work. Now, using the fact that the languages recognized by NFAs is exactly the regular languages, we will be able to show that regular languages are closed under each of the above operations by simply showing how to build a NFA which recognizes the appropriate language.

I. UNION

We begin with the regular operation for which we already showed that the class of regular languages is closed under, the union operation. Recall that we did this with the cartesian product construction. Now lets see how to do this using NFAs. Let A and B be two regular languages. Then there exists NFAs M_1 and M_2 which accept A and B respectively. Now lets draw some pictures which show how to construct a NFA which accepts $A \cup B$. As you can see

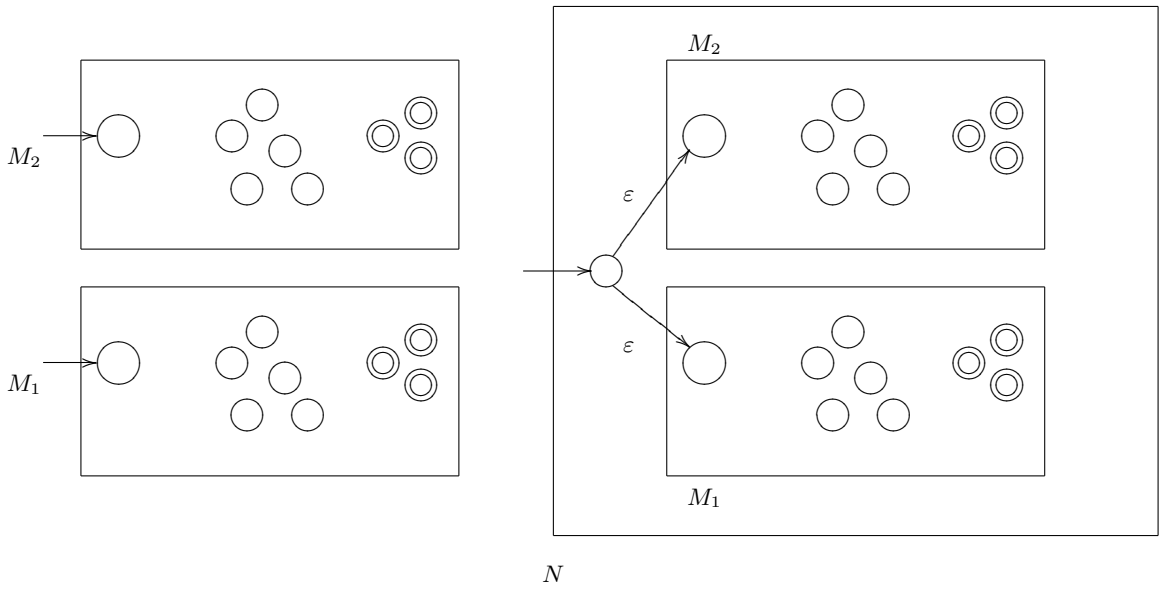

FIG. 1: Drawing this in LaTeX was a real pain.

from the figure, the basic ideas is rather simple. Simply construct a NFA with a start state, wire this start state via ε transitions to the starts states of the original machines. Thus, the NFA starts in the new state, creates two live threads at the start states of the M_1 and M_2 NFAs and then proceeds to evaluate these two machines, exactly in parallel.

Formally: Define machines $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ which recognize regular languages A and B respectively. The NFA constructed from these DFAs which recognizes $A \cup B$ is then $N = (Q, \Sigma, \delta, q_0, F)$ with

1. $Q = \{q_e\} \cup Q_1 \cup Q_2$. i.e. the state is the union of all the states plus one extra state (called q_e.)
2. $q_0 = q_e$, i.e. the start state is q_e.
3. The state accepts $F = F_1 \cup F_2$. I.e. the machine accepts if either one of the two machines accepts.

4. The transition function is defined as

$$\delta(q,a) = \begin{cases} \{\delta_1(q,a)\} & q \in Q_1 \\ \{\delta_2(q,a)\} & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases} \tag{1}$$

II. CONCATENATION

Now lets do concatenation, our original motivating example. Again the idea is to use construct a NFA out of two machines which recognize the languages we wish to concatenate. Recall that the concatenation of two languages A and B is

$$A \circ B = \{xy | x \in A, y \in B\} \tag{2}$$

The trick here is to get the machine that accepts A to branch off every time it reaches and accept state and check to see if the rest of the string corresponds to B.

Formally: Define machines $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ which recognize regular languages A and B respectively. The NFA constructed from these NFAs which recognizes $A \circ B$ is then $N = (Q, \Sigma, \delta, q_0, F)$ with

1. $Q = Q_1 \cup Q_2$.
2. $q_0 = q_1$. The start state is the start state of M_1.
3. $F = F_2$. The accept states are the states that end up accepting even the second string.
4. The transition function is

$$\delta(q,a) = \begin{cases} \{\delta_1(q,a)\} & q \in Q_1 \text{ and } q \notin F_1 \\ \{\delta_1(q,a)\} & q \in F_1 \text{ and } a \neq \varepsilon \\ \{\delta_1(q,a)\} \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \{\delta_2(q,a)\} & q \in Q_2 \end{cases} \tag{3}$$

This new transition function says that the system behaves as if it was machine M_1 until it finds itself in an accept state for M_1. In that case it is in an accept state, the transition function is now amended to include ε transitions from such a state to the start state of M_2.

III. STAR

Recall that the star operation on a language is defined as

$$A^* = \{a_1 a_2 \dots a_k | k \geq 0, a_i \in A\} \tag{4}$$

Note that A^* always contains the empty set. To show that regular languages are closed under the star operation, we take the machine which recognizes A and send all of the accept states back to the original start state via ε transitions. That way, every time the machine reaches and accept start, it can basically check if starting over again will lead to an accepting state. In addition we need to deal with the fact that we have to accept the empty string, ε. To do this we make the start state a new, ancillary state, which is an accept state and attach it to the original start state of the automata via a ε transition.

Formally: Define machines $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ which recognize regular language A . The NFA constructed from this NFA which recognizes A^* is then $N = (Q, \Sigma, \delta, q_0, F)$ with

1. $Q = Q_1 \cup \{q'\}$ where q' is the extra state we need in order to accept the empty string.
2. $q_0 = q'$. The new start state is the extra state.
3. $F = F_1 \cup \{q'\}$. The accept state is all the original accept states plus the new start state.

4. The transition function is

$$\delta(q,a) = \begin{cases} \{\delta_1(q,a)\} & q \in Q_1 \text{ and } q \notin F_1 \\ \{\delta_1(q,a)\} & q \in F_1 \text{ and } a \neq \varepsilon \\ \{\delta_1(q,a)\} \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q' \text{ and } a = \varepsilon \\ \emptyset & q = q' \text{ and } a \neq \varepsilon \end{cases} \tag{5}$$