Finite Automata and Regular Languages - Unions of FSA’s

The connection between regular languages and languages accepted by some FSA is quite simple: the regular languages just are the languages accepted by FSA’s. In this class we won’t be going in detail through the proof that this is true. (It is complicated, with many challenging details.) But there are some highlights of the proof it is important to appreciate. First of all, note the basic structure of the argument: we are proving two sets equal. As usual, we address this by breaking the problem into two parts: prove that the class of regular languages contains the class of languages recognized by FSA’s and then prove that the class of languages recognized by FSA’s contains the class of regular languages. I’ll say some things here about one of these directions: the proof that every regular language is recognized by some FSA.

The proof exploits this simple fact about the definition of the class of regular languages over the alphabet \( \{a_1, a_2, ... a_n\} \): it is defined as the closure of the set \( \{\{a_1\}, \{a_2\}, ..., \{a_n\}, \emptyset, \{\epsilon\} \} \) under union (\( \cup \)), intersection (\( \cap \)), Kleene *, and set-concatenation (\( \circ \)). That is, the class of regular languages over \( \{a_1, a_2, ... a_n\} \) is the smallest set that contains \( \{\{a_1\}, \{a_2\}, ..., \{a_n\}, \emptyset, \{\epsilon\} \} \) and is closed under \( \cup, \cap, \text{Kleene }*, \text{and } \circ \). This fact supports a strategy for proving containment one way: We can prove that all regular languages can be recognized by some FSA if we can prove:

i) Every member of the set \( \{\{a_1\}, \{a_2\}, ..., \{a_n\}, \emptyset, \{\epsilon\} \} \) can be recognized by some FSA and

ii) The class of languages recognized by FSA’s is closed under \( \cup \), intersection \( \cap \), Kleene *, and \( \circ \).

Part i) is simple, and we’ve already addressed it in problem sets. So we are left with ii). This one is more challenging, and working through parts of it will give us a better sense of the power and limits of FSA’s. In the problem set for this week we cover parts of the proof (closure under \( \cap \), closure under *). The most challenging part is closure under \( \cup \); I’ll walk you through that one now. (Closure under \( \cap \) is also challenging if you are beginning from scratch, but it is not so bad if you already have the proof of closure under \( \cup \) in front of you.)

**Problem:** Suppose that you are given two FS automata, \( F_1 \) and \( F_2 \). \( F_1 \) accepts the set \( X \) and \( F_2 \) accepts the set \( Y \). Define a new FSA in terms of \( F_1 \) and \( F_2 \) that accepts \( X \cup Y \).
In a lot of cases, when you have to define an FSA from two others $F_1$ and $F_2$, it is easiest to try to use $F_1$ and $F_2$ as modules, gluing them together without changing the inner workings. But this won’t work here. For this problem, and problem b), you have to get into the inner workings of the machines $F_1$ and $F_2$ to splice together new states, and then assemble a new machine from those states.

The basic idea: take all the states of the two machines $F_1$ and $F_2$ and “glue them together” to form new states that imitate both $F_1$ and $F_2$’s actions when reading a letter. (Of course, this talk of “gluing states together” is figurative - what we are literally doing is characterizing new states in terms of how the states of both $F_1$ and $F_2$ behave when reading a letter.)

This is an example of a problem whose core idea is easy, but whose precise execution is very involved. There are lots of little details to keep track of - lots of variables and subscripts - but you should be able to follow what is going on if you always keep the basic idea in mind: The machine $F_{new}$ is defined below so as to act like a combination of $F_1$ and $F_2$.

Let $q_1, \ldots, q_n$ be the states of $F_1$ and let $q'_1, \ldots, q'_m$ be the states of $F_2$. Say that the alphabet of $F_1$ and $F_2$ is $\{a_1, \ldots, a_k\}$; when we need to refer to an arbitrary letter in this alphabet, we’ll write $a_i$.

Let the states $q_{\alpha\beta}$ of $F_{new}$ behave as follows:

If $\langle q_{\alpha}, a_i, q_{\alpha'} \rangle$ is a transition of $F_1$ and $\langle q'_{\beta}, a_i, q'_{\beta'} \rangle$ is a transition of $F_2$ then $q_{\alpha\beta}$ and $q_{\alpha'\beta'}$ are states of $F_{new}$ and $\langle \tilde{q}_{\alpha\beta}, a_i, q_{\alpha'\beta'} \rangle$ is a transition of $F_{new}$.

If $\langle q_{\alpha}, a_i, q_{\alpha'} \rangle$ is a transition of $F_1$ and $F_2$ does nothing in state $q'_{\beta}$ when reading $a_i$, then $q_{\alpha\beta}$ and $q_{\alpha\beta'}$ are states of $F_{new}$ and $\langle \tilde{q}_{\alpha\beta}, a_i, q_{\alpha\beta'} \rangle$ is a transition of $F_{new}$.

If $\langle q'_{\beta}, a_i, q'_{\beta'} \rangle$ is a transition of $F_2$ and $F_1$ does nothing in state $q_{\alpha}$ when reading $a_i$, then $\langle \tilde{q}_{\alpha\beta}, a_i, q_{\alpha\beta'} \rangle$ is a transition of $F_{new}$.

If $\tilde{q}_{\alpha\beta}$ is a state of $F_{new}$ and $\langle q_{\alpha}, a_i, q_{\alpha'} \rangle$ is a transition of $F_1$ then $\tilde{q}_{\alpha\beta'}$ is a state of $F_{new}$ and $\langle \tilde{q}_{\alpha\beta}, a_i, q_{\alpha\beta'} \rangle$ is a transition of $F_{new}$.

If $\tilde{q}_{\beta\beta}$ is a state of $F_{new}$ and $\langle q'_{\beta}, a_i, q'_{\beta'} \rangle$ is a transition of $F_1$ then $\tilde{q}_{\beta\beta'}$ is a state of $F_{new}$ and $\langle \tilde{q}_{\beta\beta}, a_i, q_{\beta\beta'} \rangle$ is a transition of $F_{new}$.

The initial state of $F_{new}$ is $\tilde{q}_{00}$.

The only transitions and states in $F_{new}$ are those given by the above definitions.

**Useful Aside:** Note in particular that with these definitions, there is no transition with a "$\tilde{q}_{\alpha\beta}$" or "$\tilde{q}_{\beta\beta}$" as a first component unless it also has some "$\tilde{q}_{\alpha\beta}$" or "$\tilde{q}_{\beta\beta}$" as a second component. This will come in handy later in this proof! It means basically that if a computation in $F_{new}$ has a star in one of the subscripts at some state, then every subsequent state in the computation will also have a star in that coordinate. That is, if you find a "$\tilde{q}_{\alpha\beta}$" in a computation in $F_{new}$ then every later state in that computation will have * as its second subscript too. (Similarly with the first subscript.)

Now, the point of the machine we have defined is that it imitates the action of both $F_1$ and $F_2$ when reading a given letter.
We set the final states of $F_{new}$ to be those states $\tilde{q}_{\alpha\beta}$ (where either $\alpha$ or $\beta$ but not both can be *) such that either $q_\alpha$ is a final state of $F_1$ or $q_\beta'$ is a final state of $F_2$.

Claim: A string is accepted by $F_{new} \iff$ if it is accepted by either $F_1$ or $F_2$.

Proof (of Claim):

$\implies$ Say that some string $u_1 \cdots u_l$ is accepted by $F_{new}$.

We want to show that there is a computation in $F_1$ or $F_2$ that reads $u_1 \cdots u_l$.

If $u_1 \cdots u_l$ is accepted by $F_{new}$ then there is a computation reading $u_1 \cdots u_l$ and ending in a final state. That is, there is a series of states $\tilde{q}_{\alpha_1\beta_1}, \ldots, \tilde{q}_{\alpha_l\beta_l}$ of $F_{new}$ such that for every $i$ between 1 and $l$, $\langle \tilde{q}_{\alpha_{i-1}\beta_{i-1}}, u_i, \tilde{q}_{\alpha_i\beta_i} \rangle$ is a transition of $F_{new}$, and $\tilde{q}_{\alpha_l\beta_l}$ is a final state of $F_{new}$.

Recall the point mentioned above in the “aside”. By the definition of $F_{new}$ none of the transitions in $F_{new}$ can have as a first component a state with a subscript * unless the second component state in the transition has a * in the same position. In other words, if for any $j$, $\tilde{q}_{\alpha_j\beta_j}$ has a * as one of its subscripts - that is, if $\alpha_j$ is * or $\beta_j$ is *, then there are *’s in the corresponding places in every subscript on a state later in the computation. If $\tilde{q}_{\alpha_j*}$ is a state in the above computation, then so are $\tilde{q}_{\alpha_{j+1}*}, \ldots, \tilde{q}_{\alpha_l*}$. Likewise for $\tilde{q}_{*\beta_j}$.

Since $\tilde{q}_{\alpha_l\beta_l}$ is a final state of $F_{new}$ at least one of $q_{\alpha_l}$ and $q_{\beta_l}'$ must be a final state of $F_1$ or $F_2$ (depending on whether we are talking about $\alpha$ or $\beta$) and one of $\alpha_l$ and $\beta_l$ must be something other than *. We can assume without loss of generality that $q_{\alpha_l}$ is a final state of $F_1$ and $\alpha_l$ is something other than *, since the reasoning will be the same if $\beta_l$ isn’t * and $\tilde{q}_{\beta_l}$ is be a final state of $F_2$.

Since $\alpha_l$ is something other than *, then each $\alpha_i$ for $i$ between 1 and $l$ inclusive is something other than *. By the definition of $F_{new}$, this means that $q_{\alpha_0}, \ldots, q_{\alpha_l}$ are all states of $F_1$ and for every $i$ from 1 to $l$, $\langle q_{(i-1)}', u_i, q_i \rangle$ is a transition of $F_1$. But since $q_{\alpha_l}$ is a final state of $F_1$ this means that $F_1$ successfully computes $u_1 \cdots u_l$, which successfully proves this direction.

$\impliedby$

Say that one of $F_1$ or $F_2$ successfully computes $u_1 \cdots u_l$. Since it doesn’t matter which it is, let’s just say it is $F_1$. Then there is a sequence of states $q_0, \ldots, q_{\alpha_l}$ of $F_1$ such that for every $i$ between 1 and $l$ inclusive, $\langle q_{i-1}, u_i, q_i \rangle$ is a transition of $F_1$.

From the definition of $F_{new}$, there must be a sequence of states $\tilde{q}_{\alpha_0}, \tilde{q}_{\alpha_1\beta_1}, \ldots, \tilde{q}_{\alpha_l\beta_l}$ (where some of the $\beta_i$’s may be *’s) with $\langle \tilde{q}_{\alpha_{i-1}\beta_{i-1}}, u_i, \tilde{q}_{\alpha_i\beta_i} \rangle$ a transition of $F_{new}$. Since $F_1$ accepts $u_1 \cdots u_l$, $q_{\alpha_l}$ must be a final state, so again by the definition of $F_{new}$, $\tilde{q}_{\alpha_l\beta_l}$ is a final state of $F_{new}$.

Hence $F_{new}$ accepts $u_1 \cdots u_l$, which proves this direction.