Commentary on the DFSM → regular expression recursive construction.

This is from the slides for Day 12, starting with slide 12, whose title is *For Every FSM There is a Corresponding Regular Expression.* You should view those slides as you read this.

This material is also covered by Jerry Ullman's video, which is linked form Day 11 on the schedule page. THIs part of the discussion begins at 20:35 in that video. Ullman's notation is slightly different than mine, but I don't think you will have any problems with the translation.

The RegExp→NDFSM algorithm is very natural to do recursively, because of the inherently recursive nature of regular expressions. But recursion is not so natural for the other direction.

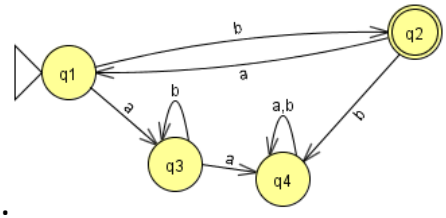
So we introduce an artificial basis for recursion: Consider DFSM M = (Q, Σ , δ , q_1, A) We number the states q_1, q_2, …, q_n. The order is arbitrary (except that q_1 is always the start state). Once we have numbered the states, the numbering is fixed throughout the construction.

Slide 12

A *k-path* (Ullman's terminology) from q_i to q_j in M is a path where none of the intermediate states in the path are numbered higher than k. It is okay if q_i and/or q_j have numbers that are higher than k. Of course the labels of the transitions along such a path is a string in Σ*.

R_{ijk} is a set of strings from Σ*. We will show by induction on k that every R_{ijk} is a regular language, whose regular expression we will call r_{ijk}.

Slide 13: Example. If you take the time understand this well before going on, the rest will be much easier for you.



Call this DFSM M.

R_{110}
This is the set of all strings that take us from state 1 to state 1 without passing through any states numbered higher than 0. Since all states are numbered higher than 0, this means not passing through any states. The only string that does this is ϵ, so R_{110} is {ϵ}, and thus r_{110} is ϵ.

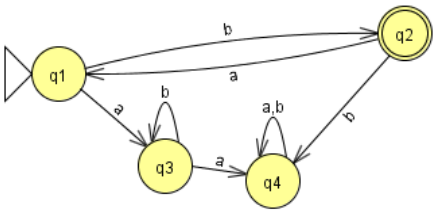
R_{120}
This is the set of all strings that take us from state 1 to state 2 without passing through any states. The only string that does this is a, so R_{120} is {a}, and thus r_{120} is a.

R_{330}
This is the set of all strings that take us from state 3 to state 3 without passing through any states. The only strings that do this are ϵ and b, so R_{330} is {ϵ,b}, and thus r_{330} is ϵ ∪ b.

r_{111}
Since there are no transitions from 1 to 1, the ability to pass through 1 does not add anything. So r_{111} is the same as r_{110} .

r_{221}
First, R_{220} is {ϵ}. But notice that if we are allowed to pass through 1, ab is also in R_{221}. So r_{220} is ϵ ∪ ab.

R_{131} is {a}, so is r_{131} is a



r_{132}

If we are allowed to have 1 and 2 as intermediate states in our path from 1 to 3, we can go around the "12 cycle" as many times as we want before going to state 3. Thus R_{132} is (ab)*a.

r_{333}

Notice that r_{331} and r_{332} are the same as r_{330}, and namely $\epsilon \cup b$. Allowing 3 as an intermediate state enables us to loop on b, so r_{333} is b*.

r_{142} is b(ab)*b.

r_{143} has two kinds of strings: those that have only 1 and 2 as intermediate states and those that have 1, 2, and 3 as intermediate states. So r_{143} is $b(ab)^*b \cup (ba)^*ab^*a$

Note that R_{124} is L(M), since it is the set of all strings that take us frim the start state to the accepting state.

Slide 14: I hope that the above example and discussion makes everything on this slide clear.

Slide 15 (DFSM→Reg. Exp. continued)

Base cases: For examples of these cases, see the first three cases from the Slide 13 example.

Induction formula: Hopefully it is clear that everything in is $R_{ij(k-1)}$ is also in R_{ijk}. All of the other strings in R_{ijk} take us from i to k without passing through k (or any higher-numbed states), then possibly loop from k to k any number of times, and finally take us from k to j. This is the basis for

R_{ijk} is $R_{ij(k-1)} \cup R_{ik(k-1)}(R_{kk(k-1)})^*R_{kj(k-1)}$

Slide 16 (DFSM→Reg. Exp. Proof pt. 1) Everything I would say about this is already in the slide.

Slide 17 (DFSM→Reg. Exp. Proof pt. 2) Everything I would say about this is already in the slide.

Slide 18 (DFSM→Reg. Exp. Proof pt. 3) Everything I would say about this is already in the slide.

Slide 19: Example

Verify that the formulas given in the slide come from the base cases and the inductive definition (and in some cases, simplification of the reg exp that is so obtained).

Here are some of the notes that I have in this slide:

$r_{221} = r_{220} \cup r_{210}(r_{110})^*r_{120} = \varepsilon \cup 0(\varepsilon)^*0 = = \varepsilon \cup 00$

$r_{132} = r_{131} \cup r_{121}(r_{221})^*r_{231} = 1 \cup 0(\varepsilon \cup 00)^*(1 \cup 01) = 1 \cup 0(00)^*(\varepsilon \cup 0)\,1$.

Note that $0(00)^*(\varepsilon \cup 0)$ is equivalent to 0*, so we get $1 \cup 0^*1$ which is equivalent to 0*1.

Have students (on the quiz) do r123 and r133 and simplify them. Compare notes with another student.

$r_{123} = r_{122} \cup r_{132}(r_{332})^* r_{322} = 0(00)^* \cup 0^*1(\varepsilon \cup (0 \cup 1)0^*1)^*(0 \cup 1)(00)^* = 0(00)^* \cup 0^*1((0 \cup 1)0^*1)^*(0 \cup 1)(00)^*$

$r_{133} = r_{132} \cup r_{132}(r_{332})^* r_{332} = 0^*1 \cup 0^*1(\varepsilon \cup (0 \cup 1)0^*1)^*(\varepsilon \cup (0 \cup 1)0^*1 = 0^*1((0 \cup 1)0^*1)^*$

For the entire machine we get $r_{123} \cup r_{133} = 0(00)^* \cup 0^*1((0 \cup 1)0^*1)^*(\varepsilon \cup (0 \cup 1)(00)^*)$