Ars Digita University
Theory of Computation
Recitation 2, 05/04/01

Main Topics

- More practice with NFAs, Converting to DFAs, getting rid of epsilons.
- Formal defintion of Automata
  The math mumbo-jumbo formulation is: Some five-tuple (Q, Sigma, delta, q, F)....
  But that's not as bad as it seems. This is just some object that contains four other objects and a method. Which one is the method?
- Operations on Languages: Our friends PREFIX and MIN
- An example Proof. Just so you have a reference, we'll show you how to "prove" that Regular languages are closed under difference, ie. L1-L2 is regular if L1 and L2 are.

```
Let M1 = (Q1, Sigma, delta1, q1, F1) accept L1, and let
Let M2 = (Q2, Sigma, delta2, q2, F2) accept L2
Then
M = (Q1xQ2, Sigma, delta, (q1,q2), F1 x(Q2-F2)) accepts L1-L2
where  delta(r1,r2,a) = (delta1(r1,a), delta2(r2,a))
```

  At this point you should give a few words of justification about why you think this new machine would accept L1-L2. If the mathematical notation is just way to much for you, just try to explain things in intuitively.
- Regular expressions

If we run out of things to talk about :-)

- Generalized NFAs
  NFA + reg. ex. transitions = GNFA
- Converting NFAs to Regular expressions
- Converting Regular expressions to NFAs

Problems to work on

1. Write out the NFA corresponding to the language and convert to a DFA: All strings that end in 0.
2. Same for all strings whose penultimate character is 0.
3. If you like this sort of thing: same things for all strings that that have 0 as the third to the last character.
4. One more if you'd like the practice: All strings ending in {0,1}
5. Write out the NFA corresponding to this state/transition table:

```
              0           1
->p         {p,q}       {p}
  q          {r}        {r}
  r          {s}         {}
```

```
 *s         {s}          {s}
```

Here the star means we have an accept state

6. Write regular expressions for all strings that end in aaba. Where Sigma is just the usual alphabet.
7. Do the same for all strings that do not contain a.
8. Write a regular expression for the strings that contain the substring 0101, where Sigma = {0,1}.
9. Do the same for all strings that alternate between 0 and 1 and end in 0.
10. Do the same for strings that alternate between 0 and 1.
11. Hard problem: Write a regular expression for the strings over {0,1} that are divisible by 3.
12. Hard Problem: Build an automaton that recognizes the strings that have the same number of zeros and ones.

Dimitri Kountourogiannis, dimitrik@alum.mit.edu