

Section 2 Handout

Week of 9.28.09

September 27, 2009

Outline

- Recap
- Regular Expressions
- Countability/uncountability

1 Recap of the course so far...

After covering fundamental mathematical preliminaries, we examined the *finite automaton*, a simple computational model with limited memory. We proved that DFAs, NFAs and regular expressions are equal in computing power and recognize the *regular languages*, which are closed under *union*, *concatenation*, *Kleene Star*, *intersection*, *difference*, *complement*, *reversal*. We used the concept of *countability* to prove the existence of *non-regular languages*. A good reminder: if you have questions on the problem set or these section notes, just send an email to cs121@seas.harvard.edu.

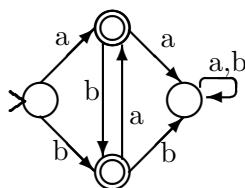
2 Regular expressions

Exercise 2.1. Write a regular expression for the following languages L over the alphabet $\Sigma = \{a, b\}$:

1. $L = \{w \mid w \text{ contains } abb \text{ as a substring} \}$
2. $L = \{w \mid w = a^n b^m \text{ where } n + m \text{ is even} \}$
3. $L = \{w \mid w = a^n b^m \text{ where } n \geq 4, m \leq 3 \}$

Regular expressions are just as expressive as a DFA. As a refresher of this fact, let's use the GNFA construction to obtain a regular expression from the following DFA:

Exercise 2.2. Find the regular expression that generates the same language as the following DFA:



3 Countability/uncountability

Things to note: The differences between cardinalities: finite, countable, countably infinite, and uncountable (uncountably infinite). How do we show that a set is countable? How do we show that a set is uncountable? Understand diagonalization and constructing bijections.

Example problems:

Exercise 3.1. Which of the following sets is countably infinite? Uncountably infinite?

1. $\mathbb{N} \times \{0, 1\}$
2. $\mathcal{P}(\mathbb{N})$
3. The set of all languages over $\Sigma = \{a, b\}$

Exercise 3.2. Here's a proposed way to make a 1-1 map from $[0, 1) \in \mathbb{R}$ to \mathbb{N} . For every $x \in \mathbb{R}$, we map it to the integer corresponding to the reversal of the decimal expansion of x . For example, 0.25 maps to 52, 0.12345 maps to 54321, and so forth. Obviously, to go back from the integer back to $[0, 1)$, we just reverse it again and prepend "0.". Is this a valid construction? Why or why not?

4 Extra Problems

Here are a couple of problems where you can further apply the material learned in section. If you want more practice doing induction proofs, do this problem by induction on the length of a regular expression:

Exercise 4.1. For any language L , let $L^R = \{w^R | w \in L\}$, where w^R is the reversal of string w . Prove that if L is regular, so is L^R .

And if you want something (significantly) more challenging, try this:

Exercise 4.2. An arithmetic progression is a set $\{p+qn : n = 0, 1, 2, \dots\}$ for some $p, q \in \mathbb{N}$. Show that for any Σ , if L is a regular language and $L' = \{|w| : w \in L\}$, then L' is a union of finitely many arithmetic progressions.

Exercise 4.3. A spy is located on a one-dimensional line. At time 0, the spy is at location A . With each time interval, the spy moves B units to the right (if B is negative, the spy is moving left). A and B are fixed integers, but they are unknown to you. You are to catch the spy. The means by which you can attempt to do that is: at each time interval (starting at time 0), you can choose a location on the line and ask whether or not the spy is currently at that location. That is, you will ask a question like “Is the spy currently at location 27?” and you will get a yes/no answer. Devise an algorithm that will eventually find the spy.