ECS120 Fall 2006

Discussion Notes

October 18, 2006

Announcements

- **Homework Pickup:** We are going to be more strict with homework turnin. Homework is due at the **start** of class on Thursdays. Homework placed in the homework box will be picked up at 1:00pm on Thursday **before** class.
- **Quiz:** Hopefully everyone was present on Tuesday for our first quiz. It was a 20 minute quiz (worth 20 points) given at the start of class. There will be another quiz someone later in the quarter. At the end of the quarter, you will have the option to drop 1 homework grade and 1 quiz grade.
- **Book Problems:** Homeworks will often include problems from the book. If you do not have the correct edition of the book, please find someone that does. It is difficult to post the problems on the discussion board due to the notation.

Homework 3 Quick Hints

Problem 1 (Sipser 1.19)

This problem is given on page 86 of your book. Follow the algorithms discussed in class or in the book to create the NFAs for the given regular expressions. It is more important to provide a **correct** NFA versus a minimal NFA, so just focus on following the algorithm precisely.

Page 67 in your book gives NFAs for the base cases. Pages 59-62 show how to combine these NFAs under the regular operations (union, concatenation, and star).

Problem 2 (Sipser 1.48)

This problem is given on page 90 of your book. I suggest listing out some of the possibilities, and try creating a regular expression from there.

Problem 3

There are two ways we discussed in class for showing that a language is regular.

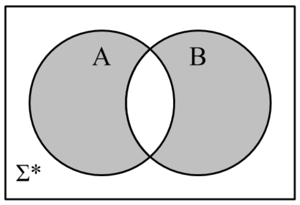
1. Show how to express the language using operations already proven to be closed under the regular languages.

   For example, consider the intersection example we covered in the last discussion section. We already know that union and complement are closed under the regular langauges. Therefore, since $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, intersection is also closed under the regular languages.

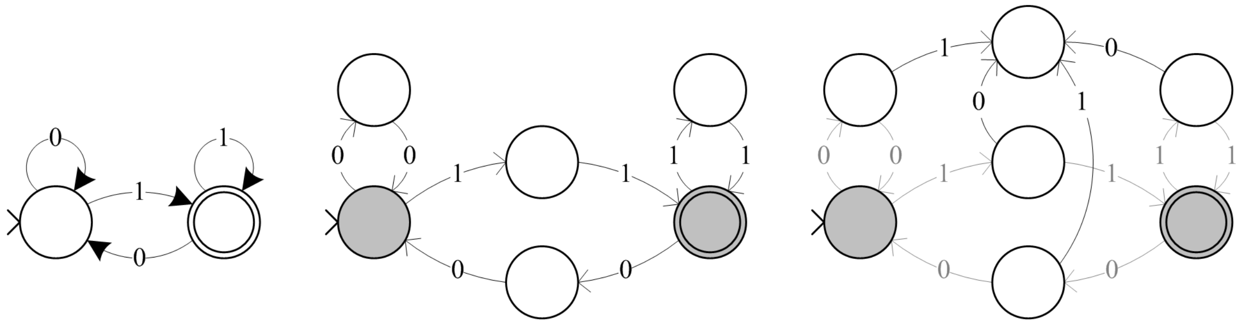
2. Show how to construct a new finite state machine to accept the new language.

   For example, this is how we proved that union, concatenation, and star are regular operations.

For part (b) notice that the Venn diagram for $A \oplus B$ looks like:



For part (c) try working out an example first:



Problem 1.40 (b) is given on page 89 of your book. It is the counterpart to the $NOPREFIX(A)$ example covered in class and in the book.

For example, let $L = \{ a, ab, abc \}$ where $\Sigma = \{ a, b, c \}$. Lets look at each string:

> Let $y = a$. This can be written as $y = \epsilon \circ a$ or as $y = a \circ \epsilon$. Therefore the prefix of y can be $x = \epsilon$ or $x = a$. The proper prefix however is $x = \epsilon$ since if $x = a$ then $y = x$.
>
> Let $y = abc$. This can be written as $y = a \circ bc$, $y = ab \circ c$, $y = \epsilon \circ abc$ and $y = abc \circ \epsilon$. The prefixes of y can be either a, ab, abc, or ϵ. However the proper prefixes are a, ab, and ϵ.

Let $L_1 = NOPREFIX(L)$, $L_2 = NOEXTEND(L)$, and x be the proper prefix of y. We can summarize everything with the following table:

y	x	$x \in A$	$y \in L_1$	$y \in L_2$
$y = a$	ϵ	*no*	*yes*	*no*
$y = ab$	ϵ, a	*yes*	*no*	*no*
$y = abc$	ϵ, a, ab	*yes*	*no*	*yes*

Problem 4

Decision procedures are basically algorithms which are able to produce a yes or no answer with certainty. Several examples were given in Tuesday's lecture, including:

- Is w in L?
- Is $L = \emptyset$?
- Is $L = \Sigma^*$?
- Is $L_1 \subseteq L_2$?
- Is $L_1 = L_2$?

We know that L is regular, but we do not know how L is expressed. It could be expressed as a regular expression, NFA, or DFA. Try to provide a polynomial time algorithm (or better) for each of the possible cases.

Decision Procedures

Decision procedures are basically algorithms which provide with certainty a yes or no answer. When giving us a decision procedure make sure to (1) specify your algorithm and (2) explicitly state when your algorithm outputs a `YES` answer and when it outputs a `NO` answer.

There are two useful techniques for creating decision procedures. If you are lucky, you can break down the problem you are solving into sub-problems which already have solutions. You can then reuse those solutions in solving the main problem.

Otherwise, you need to construct an algorithm which uses what you know or can assume. If we know a language L is regular, we know it can be represented by a DFA, NFA, or regular expression. Therefore we can show how to solve our problem given a DFA, NFA, and regular expression. The catch is that we should try and provide a polynomial-time algorithm for each case.

Example 1

Let L_1 and L_2 be regular languages. Describe a decision procedure that determines if L_1 and L_2 have at least 1 string in common.

First, we should try and break down the problem. How do we know when L_1 and L_2 share at least 1 string? This is true whenever $L_1 \cap L_2 \neq \emptyset$.

Therefore, we want to build an algorithm:

$$\textsc{Similar}(L_1, L_2) = \begin{cases} \texttt{YES} & \text{if } L_1 \cap L_2 \neq \emptyset \\ \texttt{NO} & \text{otherwise} \end{cases}$$

So now the problem has become determining if $L_1 \cap L_2 \neq \emptyset$. Notice that now we can break this into sub-problems. We need to find $L_1 \cap L_2 = L_3$, and then determine if $L_3 \neq \emptyset$.

We already know how to do $L_1 \cap L_2 = L_3$. For now, lets assume we have a procedure $\textsc{Intersect}(L_1, L_2)$.

We know that L_3 is regular. Therefore, we can also use the decision procedure discussed in class for determining if $L_3 = \emptyset$. Lets call this procedure $\textsc{IsEmpty}(L)$.

Now we can put the two together, and define our $\textsc{Similar}$ algorithm:

```
$\textsc{Similar}(L_1, L_2)$ :

    1. Find $\textsc{Intersect}(L_1, L_2) = L_3$
    2. Run $\textsc{IsEmpty}(L_3)$:
        • If $\textsc{IsEmpty}(L_3) = \texttt{YES}$, then output NO.
        • Else, output YES.
```

Example 2

Let L_1 and L_2 be regular languages. Describe a decision procedure that determines if w is not in L_1 or in L_2.

Essentially, this is determining if $w \notin L_1 \cup L_2$. We know how to take the union of two languages, and already discussed a decision procedure for $w \in L$ in class. Therefore we get:

```
$\textsc{Example2}(L_1, L_2, w)$ :

    1. Find $\textsc{Union}(L_1, L_2) = L_3$
    2. Run $\textsc{IsMember}(L_3, w)$:
        • If $\textsc{IsMember}(L_3, w) = \texttt{YES}$, then output NO.
        • Else, output YES.
```