CIS 511 Homework 3

Stephen Phillips, Dagaen Golomb

February 21, 2015

Problem 1

Show a Turing Machine that accepts the language $L = \{x \in a, b, c^* \mid x \text{ contains more } a\text{'s than } b\text{'s and } c\text{'s combined}\}$

Problem 2

Show that a language is recognizable by a queueing automaton if and only if it is recognizable by a Turing Machine.

- ($\Longleftarrow$) If we have a Turing recognizable language, by definition there must be a Turing Machine that recognizes it. Therefore if we can make a queueing automaton replicate the actions of the Turing Machine we have show that Turing recognizable languages are recognized by queueing automata.

  To do this, we simplfy the start and say that the queueing automaton pushes a 'start of tape' symbol followed by the entire string x into its queue before it starts computation. Now

Problem 3

Show that Turing recognizable languages are closed under Kleene Star.

To do this we consider the Turing recognizable language L, and the machine that recognizes it M. On a given input x, there are a large but finite number of ways to split the string into substrings. For a given partition, we can test if all the strings are in L using the following subroutine:

function A($x_1, x_2, \ldots, x_n$)
  Create list to store which runs on strings x_i have halted
  for $i = 1, \ldots, \infty$ **do**
    for j in incomplete list **do**
      Simulate the next step of $M(x_j)$
      if $M(x_j)$ has halted **then**
        If M rejects x_j, reject
        If M accepts x_j, remove j from incomplete list
      end if
    end for
    if Incomplete list empty **then**
      Accept
    end if
  end for
end function

This halts and accepts if and only if all the $x_i \in L$. It is fairly apparent, as it rejects if even one of the x_i gets rejected by M and accepts only after M has halted and accepted all of the x_i. If neither of these happens, one of the x_i must have made M loop forever, in which case A also loops forever since it will keep waiting until M halts before it halts.

Using this subroutine, we can get the following to accept L^*:

function N(x)
  Let $n = |x|$

for $i = 1, \dots, \infty$ **do**
  for $j = 1, \dots, n$ **do**
    for All divisions of x into i strings, $x_1, x_2, \dots, x_j$ **do**
      Run one step of $A(x_1, x_2, \dots, x_j)$
      If $A(x_1, x_2, \dots, x_i)$ halts and accepts, accept
    end for
  end for
end for
Reject
end function

This machine N accepts if and only if $x \in L^*$.

- ($\Longleftarrow$) If $x \in L$ then by definition there exist strings $x_1, \dots, x_n \in L$ such that $x = x_1 \dots x_n$. As N goes through all possible partitions of x, this partition will be found and therefore $A(x_1, \dots x_n)$ will accept, and therefore N will accept.
- ($\Longrightarrow$) If N accepts x, then A must have accepted for some $i \geq 1$ and with some partition $x = x_1 \dots x_i$. But $A(x_1, \dots, x_i)$ accepts if and only if all the x_j are in L, which means by definition x must be in L^*. If it wasn't then no such partition could have been formed and none of the $A(x_1, \dots, x_i)$ calls would have accepted. So therefore N only accepts strings in L^*

As N accepts all of L^* and only accepts strings in L^* the language of N must be L^*. Since this was done for arbitrary Turing recognizable language L it is true for all Turing recognizable languages.

Problem 4

We show that a language L is decidable if and only if it is enumerable in the standard string order.

- ($\Longleftarrow$) If L has an enumerable in the standard string order, then there exists an Turing machine, the enumerator E, that does exactly that: output all the strings in the language in the standard string order. Using that enumerator we can create a decider D:

  function D(x)
    for All strings y output by enumerator E **do**
      If $y = x$, accept
      If y comes after x in the standard string order, reject
    end for
  end function

  If E truly outputs all the strings in the language in the standard string order, then if x is in the language, it will eventually output x. Then D will accept when it is output by E. If it is not in the language, then since E outputs all stings in the language in the standard string order, when we see a string y that comes after x in that order, we know we will never see x in the language, so we will reject. So therefore the language of D is L. And it is a decider, since x has some finite index in the standard ordering of strings, one of these two outcomes will be reached in finite time. Therefore we will halt in finite time on all strings.

- ($\Longrightarrow$) If L is decidable, there must be some decider D that decided the language. Therefore we can make the following enumerator E:

  function E(z)
    Ignore input z
    for All strings x in the standard string order **do**
      If $D(x)$ accepts, output x
    end for
  end function

  As D is a decider, for every string x it will halt in finite time. Since it only accepts strings in L, E can only output strings in L. Since we are enumerating stings in the standard order, decide once whether or not to output them, we therefore must output the strings in L in the standard sting order, and E functions as we wanted it to.

Problem 5

Show that the language $L = \{\langle G \rangle \mid G \text{ is a CFG over } \{0,1\} \text{ and } L \cap 1^* \neq \varnothing\}$ is Turing decidable.

We use a similar method as finding if a CFG generates the empty language. To assist us, we have two types of marks on the symbols, $\dot{V}$ and $\ddot{V}$. The first is to mark that we have seen it and it can generate a string in 1^*. The second is to mark we have seen it but it cannot generate a string in 1^*.

The idea is that we look through the production rules and find strings in 1^*. We start at production rules that have right hand sides with only terminals, specifically terminal strings with only 1 in it. For every production rule we find with only 1 in it, we mark the corresponding variable V with the first kind of mark $\dot{V}$. If there are variables that can generate termial strings but not ones in 1^*, then we mark them with the second kind of symbol $\ddot{V}$. We denote the first group S_1 and the second group S_2.

Now we look at production rules with right hand sides with only symbols in $S_1 \cup S_2$. If one of these right hand sides has only symbols in S_1 we place the corresponding variable from the left hand side into S_1. The rest of the variables we put into S_2. We repeat this until we have all variables in $S_1 \cup S_2$. If the starting symbol $S \in S_1$ then we know that this grammar generates strings in 1^*, and we accept. Otherwise we reject.

Problem 6

Show that $L = \{\langle A \rangle \mid L(A) = \varnothing\}$ is co-Turing recognizable.

To show this is co-Turing recognizable, we must show that $\bar{L}$ is Turing recognizable. In other words $\bar{L} = \{\langle A \rangle \mid L(A) \neq \varnothing\}$. This is clearly recognizable, since we can just simulate A on all strings by the usual dove-tailing method. This is described in detail below, described by the machine M.

function M(x)
    for $i = 1, \ldots, \infty$ **do**
        for x in the first i strings in Σ^* **do**
            Run $A(x)$ for i steps
            If $A(x)$ halts and accepts, accept
        end for
    end for
end function

So this machine will run forever on machines A which have the empty language which is fine since this is a recognizer so it does not need to halt on strings not in the languages. However if A has even one string in its language, it will accept, since A must halt and accept that string in some finite time and therefore M will as well.

As $\bar{L}$ is recognizable, L must be co-Turing recognizable.

Problem 7

Problem 8