Ungraded Homework Solutions

CSC 135 – Programming Languages and Computer Theory

If you see any mistakes, please let me know.

1) *Write a reduction from `median` (the problem of finding the middle integer in a list of integers) to `sort` (the problem of sorting a list of integers). What connection does this reduction establish between the two problems (express as an implication)? What is the contrapositive of this implication?*

```
median(L):
    sortedL = sort(L)
    midOfL = sortedL[len(L)/2]
    return midOfL
```

This reduction establishes that if there exists a sorting algorithm then there exists a median-finding algorithm (ie, $\exists$ `sort` $\rightarrow$ $\exists$ `median`) and also the contrapositive that if there does not exist a median-finding algorithm then there does not exist a sorting algorithm (ie, $\neg\exists$ `median` $\rightarrow$ $\neg\exists$ `sort`).

2) *A famous result says that there is no algorithm that can take as input another algorithm M and its input x and, in every case, determine whether M will eventually halt while processing x. If we wanted to show that some problem A was also not always computable, which of the following two reductions would do so, given that HaltSolver is not computable? Explain why you chose the one you did and write the logical implications the reduction establishes.*

```
HaltSolver(M,w):
   // convert <M,w> into a_instance
   a_answer = ASolver(a_instance)
   // convert a_answer into halt_answer
   return halt_answer
```

```
ASolver(a_instance):
   // convert a_instance int <M,w>
   halt_answer = HaltSolver(M,w)
   // convert halt_answer into a_answer
   return a_answer
```

The left-hand one establishes $\exists$ `ASolver` $\rightarrow$ $\exists$ `HaltSolver` and also the contrapositive $\neg\exists$ `HaltSolver` $\rightarrow$ $\neg\exists$ `ASolver`. Since we know that there is no HaltSolver, the left-hand reduction establishes that there is no ASolver.

3) *Write a reduction from `min` (the problem of finding the smallest integer in a list of integers) to `max` (the problem of finding the largest integer in a list of integers). What connection does this reduction establish between the two problems (express as an implication)? What is the contrapositive of this implication? Hint: You may use a loop and you may call `remove` on an element to remove it. It's okay to call the function you are reducing to multiple times... It still shows how to solve one problem by using the solution to another.*

```
min(L):
    Lcopy = copyOf(L)
    while Lcopy.size > 1
        x = max(Lcopy)
        Lcopy.remove(x)
    minOfL = Lcopy[0]
    return minOfL
```

This reduction establishes that if there exists a max-finding algorithm then there exists a min-finding algorithm (ie, $\exists$ `max` $\rightarrow$ $\exists$ `min`) and also the contrapositive that if there does not exist a min-finding algorithm then there does not exist a max-finding algorithm (ie, $\neg\exists$ `min` $\rightarrow$ $\neg\exists$ `max`).

5) *Let M be an algorithm that takes an input x and always outputs either true or false. It is known that there is no algorithm Any(M) that returns true if M outputs true for at least one of its possible inputs and false if M never outputs true. In other words there is no algorithm that can decide whether M accepts any input at all.*

Show that because Any is not computable Equal(M1, M2) is also not computable, where Equal returns true if M1 and M2 output true for the exact same sets of inputs. To do this, reduce from Any to Equal. Hint: As the first line of your reduction write "Let M2 be the algorithm that always outputs false".

```
Any(M):                  // Returns whether M accepts any inpt
    Let M2(x) always outputs reject no matter what x is
    return not Equal(M, M2)
```

Because M2 doesn't output true for anything, Equal(M,M2) will evaluate to true if M doesn't output true for anything either. Thus, if M outputs true for anything, Equal(M,M2) will evaluate to false. We negate it to get the proper answer.

1) *Do Exercises 1.13, 1.16, 1.20, 1.22 from Theory of Computation.*

Note: this textbook uses ε for the empty string instead of λ. These solutions follow the book's convention, but I'd accept λ too. Also, if the right-of-head or left-of-head string contains only blanks I write ε.

1.13) a) $\langle q_0, 1, \varepsilon, 1\rangle \vdash \langle q_0, 1, 1, \varepsilon\rangle \vdash \langle q_0, B, 11, \varepsilon\rangle \vdash \langle q_1, 1, 1, \varepsilon\rangle \vdash \langle q_1, B, 1, \varepsilon\rangle \vdash \langle q_2, 1, \varepsilon, \varepsilon\rangle \vdash \langle q_2, B, \varepsilon, 1\rangle \vdash \langle q_3, 1, \varepsilon, \varepsilon\rangle$.

b) $\langle q_0, 1, \varepsilon, 1B11\rangle \vdash \langle q_0, 1, 1, B11\rangle \vdash \langle q_0, B, 11, 11\rangle \vdash \langle q_1, B, 11, 11\rangle \vdash \langle q_1, 1, 11, 11\rangle \vdash \langle q_2, 1, 11, 11\rangle \vdash \langle q_2, 1, 1, 111\rangle \vdash \langle q_2, 1, \varepsilon, 1111\rangle \vdash \langle q_2, B, \varepsilon, 11111\rangle \vdash \langle q_3, B, \varepsilon, 11111\rangle \vdash \langle q_3, 1, \varepsilon, 1111\rangle \vdash \langle q_4, B, \varepsilon, 1111\rangle \vdash \langle q_5, 1, \varepsilon, 111\rangle$

c) By configuration sequences, I assume they mean similar to the drawings on Page 6. The above traces require more attention-to-detail than the configuration drawings, so I omit those.

1.16) $\langle q_0, B, \varepsilon, \varepsilon\rangle \vdash \langle q_0, B, \varepsilon, \varepsilon\rangle \vdash \langle q_0, B, \varepsilon, \varepsilon\rangle \vdash \langle q_0, B, \varepsilon, \varepsilon\rangle \vdash \langle q_0, B, \varepsilon, \varepsilon\rangle \vdash \langle q_0, B, \varepsilon, \varepsilon\rangle \vdash \langle q_0, B, \varepsilon, \varepsilon\rangle \vdash \langle q_0, B, \varepsilon, \varepsilon\rangle \vdash \cdots$

1.20) "Not" by toggling the one bit without moving the head. "And" by blanking the first character and altering the second character, if necessary. If the first character is 1, the second character can be left alone. If the first character is 0, then the second character gets changed to 0. In either case move the head over the second character. See diagrams below.

1.22) One way to do it is to blank all existing 1's, write three 1's, return head to beginning. See diagrams below.

Note: Drawings are sometimes clearer without labels. If you wanted to turn these diagrams into a set of instructions, you'd need to assign each state a unique label (with start q_0).