6CCS3COM Computational Models

Tutorial 1 Solutions

Josh Murphy

1. Give examples of total and partial functions on natural numbers.

   There are many examples of total functions. Addition, multiplication, and any combination of these, as well as the well-known factorial function, are all total. Subtraction is partial on natural numbers (but total on integers).

2. To test whether a number is even or odd, a student has designed the following function:

```
test(x) :
    if x = 0 then ‘‘even’’
    else if x = 1 then ‘‘odd’’
    else test(x-2)
```

   Is this a total function on the set of integer numbers? Is it total on the natural numbers?

   The set of natural numbers contains 0 and all the positive integers. For these, the test provided above gives a result: For 0 the result is "even", for 1 the result is "odd", and for any number x greater than 1 the number x-2 is still a natural number that can be tested again. Since each recursive call to the function test carries a smaller argument, it is easy to see that eventually the function will be called with either 0 or 1 and will produce a result. Therefore the function is total on natural numbers.

   However, if the function is called with a negative number, for example test(-1), then there is no result. Therefore the function is partial on integers.

3. Consider the following variant of the Halting problem:

   "Write an algorithm `H'` such that, given the description of an algorithm `A` that requires one input, `H'` will return `1` if `A` stops for any input `I` and `H0` will return `0` if there is at least one input `I` for which`A` does not stop."

   In other words, the algorithm `H'` should read the description of `A` and decide whether it stops for all its possible inputs or there is at least one input for which `A` does not stop.

   Give the intuition of a proof that shows that this version of the Halting problem is also undecidable.

   We assume this problem is decidable (that is, there is an algorithm H' as specified) and then we show that there is an algorithm H to decide the Halting problem (contradiction). We proceed as follows.

   Assume we have an algorithm H' such that for any algorithm A:

   - H'(A)=1 if A stops for every input.
   - H'(A)=0 if there exists an input I for which A does not stop.

   Using the algorithm H' we build an algorithm H as follows:

   - H(A, I) = H' (A_I), where A_I is an algorithm with one input and one output defined as follows:
     - A_I(X): if X = I then A(I), else 0
     - In other words, A_I(X) returns 0 for every input except I, for which it performs the same computation as A(I).

Since by definition H(A, I) = H' (A_I), then H(A, I)=1 if A(I) stops and H(A, I)=0 if A(I) runs forever (i.e., H solves the Halting problem).