CS 311 Homework 7

November 18, 2013

1. (from exercise 3.2 in Sipser) Consider the Turing machine presented in Example 3.9 in Sipser. In each of the parts, give the sequence of configurations that the Turing machine enters when started on the indicated input string.

   (a) 1#1

   (b) 1##1

   (c) 10#11

   (d) 10#10

2. (from 3.8 in Sipser) Give the state machine descriptions for Turing machines that recognize the following languages

   (a) $\{w|w$ contains twice as many 0s as 1s$\}$

   (b) $\{w|w$ does not contain twice as many 0s as 1s$\}$

3. (from 3.16 in Sipser) Show that the Turing-recognizable languages are closed under

   (a) concatentation

   (b) star

   This problem requires providing constructions that take individual Turing machines and combines them into a new machine that recognizes the new language. We're looking for a high level description in the style of what Sipser calls a "implementation" level in pages 184-185. Also look at the answers to 3.16.a in the back of the chapter for an example of the kind of solution we're looking for.

4. In the lecture notes we introduced a grammar for an algebra of primitive recursive functions.

```
Term ::= Z
       | S
       | P N
```

```
       | C  Term  [ Term1, ... ,TermN ]
       | PR  Term  Term

N ::=  1 | 2 | 3 | 4 | ...
```

In lecture notes and in the text below, primitive recursive functions over natural numbers are defined. I have also written a Haskell interpreter for that formalization. It is in the file NaturalPR.hs. for those who might want to study it. This file is available on the class index page.

For this assignment you have two options, either of which is acceptable. You may do either of the following:

(a) do the original assignment below with pencil and paper,

(b) do the original assignment below as a programming exercise (possibly starting from NaturalPR.hs)

1 Original Assignment

(a) In lecture I presented five schemas for defining primitive recursive functions. They are as follows:

i. [Zero] There is a constant function zero of every arity.

$$Z(x_1, \ldots, x_k) = 0$$

ii. [Successor] There is a successor function of arity 1.

$$S(x) = x + 1$$

iii. [Projection] There are projection functions for every argument position of every arity.

$$P\ i\ (x_1, \ldots, x_k) = x_i \qquad \text{where } k > 0,\ i \leq k$$

iv. [Composition (also called substitution)] The composition of the function f of arity k with functions $g_1, \ldots g_k$, each of arity l, defines a $C\ f\ [g_1 \ldots g_k]$ of arity l satisfying:

$$C\ f\ [g_1 \ldots g_k](x_1, \ldots, x_l) = f\ (g_1(x_1, \ldots, x_l), \ldots, g_k(x_1, \ldots, x_l))$$

v. [Primitive Recursion] The arity k function defined by primitive recursion from a function h of arity $k-1$ and a function g of arity $k+1$ is indicated PR h g. It satisfies:

$$\begin{array}{lcl} \text{PR } h\ g(0, x_2, \ldots, x_k) & = & h(x_2, \ldots, x_k) \\ \text{PR } h\ g(x+1, x_2, \ldots, x_k) & = & g(x, \text{PR } h\ g(x, x_2, \ldots, x_k), x_2, \ldots, x_k) \end{array}$$

In lecture we showed how to define addition by primitive recursion:

$$\text{add} = \text{PR}(P1)(C\ S\ [P\ 2])$$

Using primitive recursion define:

i. Multiplication
ii. Constant true function (e.g. $true(x_1, \ldots, x_n) = 1$)
iii. Constant false function (e.g. $false(x_1, \ldots, x_n) = 0$)
iv. If-then-else (e.g. $\text{ITE}(1, x, y) = x, \text{ITE}(0, x, y) = y$)
v. Or
vi. And
vii. Not
viii. Minus (e.g. $x - y$ when $x > y$ and 0 otherwise)
ix. Integer equality
x. The factorial function

Note that each of these solutions is a term in the grammar given above.