CS 410/510spec Homework 2 – Tuesday, January 26, 2016

This homework has two parts: pencil&paper exercises to be turned in on paper in class, and a modelling exercise in TLA+, to be turned in by zipping up your `.tla` file and `.toolbox` directory and emailing them to `tolmach@pdx.edu`.

Do the following Huth&Ryan exercises:

2.1.2.

2.1.3.

2.1.5.

2.2.3.(b)

2.2.5.

2.3.1.(a)

2.3.2.

2.3.4.(a,b)

2.3.7.(b)

2.3.9.(e)

2.3.13.(e)

2.3.14

2.4.1.

2.4.2.

2.4.6.

2.4.12.(e,k)

TLA+ Exercise. Write and check a TLA+ specification for a vending machine that stocks a small number of different kinds of products (e.g. different kinds of candy or soda). The machine is initally stocked with zero or more products of each kinds, and zero or more coins in its internal bank. When in operation, it responds to the following external actions:

- Customer deposits a coin.
- Customer selects a product. If they have deposited enough coins, the product is delivered, the money needed to pay for it is retained by the machine, and any amount of extra money they may have deposited is returned to them (the machine "makes change.") Otherwise, nothing happens.
- Customer selects "cancel." Any coins they have deposited so far are returned.

Your specification should allow you to use TLC to check the property that the total value of products in the machine's stock plus money in its internal bank is invariant during operation. Actually, this isn't quite true: you'll need to come up with a more precise invariant that is true.

Do this exercise in two steps:

(a) To begin with, assume that products all cost some multiple of \$1, and the machine only accepts dollar coins.

(b) As a second step, assume that products all cost some multiple of \$0.10 and that the machine accepts either dimes or dollar coins, with separate internal storage for each kind of coin in its bank. Note that in this scenario, it is possible for the machine to be unable to make change in certain circumstances (e.g. consider depositing a dollar coin and selecting an item that costs less than \$1). Make sure that your specification considers this possibility and prevents an item from being selected if the machine cannot make change for it.

Here are some guidelines and hints for your specification. Feel free to ignore/change any of these if you think you have a better way to express the problem. Use comments to explain anything non-obvious in your specification.

- The set of products and their prices should be `CONSTANTS`. Represent products as a set of model values. Represent prices as a function from products to `Nat` (you'll need to make your spec `EXTEND Naturals`). Hint: to describe a concrete function given by the ordered pairs $\{(d_1, r_1), (d_2, r_2), \ldots, (d_n, r_n)\}$ as the value of a constant, use the notation
  `(` d_1 `:>` r_1 `@@` d_2 `:>` r_2 `@@ ... @@` d_n `:>` r_n`)`.
- The `VARIABLES` should include the current stock (a function from product to `Nat`), the current internal bank (a `Nat` for each kind of coin), and the amount deposited so far (again, a `Nat` for each kind of coin).
- The initial values of these variables should be specified as `CONSTANTS` too (except for the coins deposited so far, which can be initialized to zero).
- There should be three actions; the `Next` action should just be a disjunction of these.
- You may find the following higher-order operator (familiar from functional languages) useful for computing the value of the products in the machine:

```
(* Operator to fold over a set s using function f and initial value i *)
Fold(f(_,_),i,s) ==
   LET DFold [s0 \in SUBSET s] ==
      LET elt == CHOOSE e \in s0 : TRUE
      IN IF s0 = {} THEN i ELSE f(elt,DFold[s0 \{elt}])
   IN DFold[s]

(* Example: compute sum of a set *)
Sum(s) == Fold(LAMBDA e,r : e + r,0,s)
```

Feel free to send me mail at `tolmach@pdx.edu` if you have any questions.