Language Engineering - Problem Sheet 1

Dom Hutchinson

October 14, 2018

Question 1.1 - Robot
We have a very basic turtle robot which can move forward a specific amount (as an *Int*), rotate left and right 90^o as well as stop.

Question 1.1.1
Design a recursive data structure (without using lists) which encodes the basic *Robot*. It should have four data constructors relating to the different operation is can do.

My Solution 1.1

```
data Robot = Forward Int Robot
           | LeftTurn Robot
           | RightTurn Robot
           | Stop
```

Question 1.1.2
Write a function which calculates the total distance travelled by the robot, it should have the type

$$distTrav :: Robot \rightarrow Int$$

My Solution 1.2

```
distTrav (Forward n r) = n + distTrav r
distTrav (LeftTurn r)  = distTrav r
distTrav (RightTurn r) = distTrav r
distTrav (Stop)        = 0
```

Question 1.1.3
Design a function that calculates the distance travelled in the direction that the robot was first facing.

My Solution 1.3

```
distTravForward :: Int → Robot → Int
distTravForward 0 (Forward n r) = n + distTravForward 0 r
distTravForward d (Forward n r) = distTravForward d r
distTravForward d (LeftTurn r) = distTravForward d r
distTravForward d (RightTurn r) = distTravForward d r
distTravForward _ (Stop r) = 0

distTrav :: Robot → Int
disTrav r = distTravForward 0 r
```

Question 1.1.4
Design a function that calculates the distance the robot ends up away from its starting position, in a straight line, with the output as a *Float*.

My Solution 1.4

```
distTrav :: Robot → Float
distTrav r = sqrt(fromIntegral(x*x) + fromIntegral(y*y))
  where
    y = distTravForward 0 r
    x = distTravForward 1 r
```

Question 2 - Cooking Master We will attempt to model a basic set of recipes. However, to make life easier we shall initally only include potatoes. This will be done using a shallow embedding, where the semantic output will be the properties of the potatoes:

i) Time Taken : The time it takes, as an *Int*, to prepare the potato;

ii) Weight : The weight of potatoes prepared, as an *Int*;

iii) Cooked : Whether the potatoes have been cooked, as a *Bool*;

iv) Description : Information about the potatoes as a *String*.

Question 1.2.1
Create a value encoding a single potatoe, with the below type, which represents the various semantic outputs of a potato dish.

$$potatoe :: (Int,\ Int,\ Bool,\ String)$$

The data type $(Int,\ Int,\ Bool,\ String)$ in this case relates directly to $(time\ taken,\ weight,\ cooked,\ description)$ and *potatoe* will encode a single potato into this semantics. This means that we would assume the time taken to be nothing, we will only have one potato with weight 3, it won't be cooked and the only description we can given will be *"potato"*.

My Solution 2.1

```
potato = (0, 3, False, "potato")
```

Question 2.2
Create functions for each of the following different culinary experises exercised on potatoes. They will need to map the current semantics to an update semantics.

$$ce :: (Int,\ Int,\ Bool,\ String) \rightarrow (Int,\ Int,\ Bool,\ String)$$

i) peel (takes 2 mins for ech potato and adds *"peeled"* to the description)

ii) roast (takes 70 mins, makes them cooked and adds *"roasted"* to the description)

iii) boil em (takes 25 mins, makes them cooked and adds *"boiled"* to the description)

iv) mash em (takes 1 min per potato and adds *"mashed"* to the description)

v) stick em in a stew (takes 120 mins, makes them cooked and adds *"stewed"* to the description)

My Solution 2.2

```
peel (t, w, c, s)                 = (t+(2*w), w, c, "peeled " ++ s)
roast (t, w, c, s)                = (t+70, w, True, "roasted " ++ s)
boil em (t, w, c, s)              = (t+25, w, True, "boiled " ++ s)
mash em (t, w, c, s)              = (t+w, w, c, "mashed " ++ s)
stick em in a stew (t, w, c, s) = (t+120, w, True, "stewed " ++ s)
```

Question 1.2.3
Create a function that lets you mix two sets of potatoes, this should combine the time taken and weights, become uncooked if either is uncooked and combine the two descriptions. The type signature is given below:

$$mix :: (Int,\ Int,\ Bool,\ String) \rightarrow (Int,\ Int,\ Bool,\ String) \rightarrow (Int,\ Int,\ Bool,\ String)$$

My Solution 2.3

```
mix (t1, w1, False, s1) (t2, w2, _, s2) = (t1+t2, w1+w2, False, s1 ++ " " ++
 s2)
mix (t1, w1, _, s1) (t2, w2, False, s2) = (t1+t2, w1+w2, False, s1 ++ " " ++
 s2)
mix (t1, w1, _, s1) (t2, w2, _, s2)     = (t1+t2, w1+w2, True, s1 ++ " " ++
 s2)
```

Question 1.2.4
Now allow for your next two favourite root vegetables to be added.

My Solution 2.4

```
carrot  = (0, 1, False, "carrot")
parsnip = (0, 1, False, "parsnip")

addcarrot (t, w, c, s)  = mix (t, w, c, s) carrot
addparsnip (t, w, c, s) = mix (t, w, c, s) parsnip
```

Question 1.3 - Languages

Question 1.3.1
What is the difference betwen a GPL and a DSL?

My Solution 3
A general purpose language is Turing Complete so can be used to develop programs in any domain, whereas a domain specific language is not. This means GPLs require their own support systems, while DSLs can make usr of these support systems and so don't need to fully develop their own.

Question 1.3.2
What programming methods should be used in shallow embeddings and deep embeddings when dealing with dependent interpretation?

My Solution 3
When using multiple semantics in shallow embeddings you provide all the semantics as a tuple and providing their interpretations with them. In deep embeddings you define a new function for each semantics.