Homework 1 for Programming languages

Sugeerth Murugesan

April 22, 2014

1 Problem 0

- My research area is in the general area of computer graphics. One of the research projects that I am currently working on is to develop a refined, cross-disciplinary, and user friendly method of prototype that can be used by the medical community. I am motivated by a desire to help improve the process of medical diagnosis and treatment in patients whose motor skills are in question.Although I have not exclusively specified my research problem, I hope to find a solid research problem by the end of this quarter.
- The top three problems in the area of computer graphics are :
  - One of the problems in the area is produce the next generation Human computer Interaction with user gestures and features. This is one of the problems that I am really excited about.
  - There is also the problem of resolution. Graphics applicaitons are always encountered with poor resolution and precision in the pictures that are produced. If one was able to find the best resolution graphics in a not so very advanced graphics cards, that would really solve the problem of graphical computation.
  - GPUs and graphics related hardware also face great deal of problem with the "power wall" problem. The amount of transistors that could fit into one die is very limited, this in turn affects the performance of the CPU as well as the GPUs. Technological and research advancements can be made to improve the performance of graphics cards.
- With regard to language design the concept of inheritance and modularity has really benefitted to software development. These concepts have provided a system, essentially improved programming languages. Inheritance has made sure that one need not rewrite rudandant piece of code. Also, most of the IDEs have contributed to programming. One could efficiently debug and rectify the error that needs to overcome. The concept of visualizing the run of the program using Java reflections has also proved to be a good tool to analyze the runtime of program.

- A research that makes people's lives better is an impactful research. Impactful research is the one whose research could be used by/for/to masses.The research that incorporates real world problems allows the real world to tackle problems of all kinds. The research that not only satisfies knowledge dissemination but also allows the masses to benefit from it. This is just my opinion and I could be wrong.

2 Problem 1

- To find the number of functions from X $\rightarrow$P(Y) ,we assume that X has x elements and Y has y elements, as a result the number of functions to have in the in X $\rightarrow$ P(Y) is found to be 2^{xy}

  As we already know that the number of elements in P(Y) is known to be 2^{y}, every possible x element has a unique value with the P(Y) and hence the total number of possible functions is 2^{xy}

  To encounter the case two where the number of relations is

  P(X x Y) = 2^{xy} This is proved by an example: the number of elements in (XY) is xy the numebr of elements in P(XY) is 2^{xy}

  As relation is a subset of cartesian product number of relations = subsets of cartesian product Since P(X x Y) is already a subset of the relations (X x Y) the number of possible subsets which is numbe rod distinctive relations is 2^{xy}

  Hence the total number of relations = total number of functions in (X P(Y))

  Also if we analyze every element of the P(X x Y) and the function (X - P(Y)), they are found to be the same.

  Thus we see that there is a 1-1 correspondence for the (X $\rightarrow$P(Y)) and P(X x Y) as the number of elements in the set are found to be 2^{xy}

- Number of elements in X = x Number of elements in Y = y

  Total number of functions X $\rightarrow$ Y = y ^{x}

  For having a 1-1 correspondance, the basic requirement is that every element of set X is paired with exactly one element of the other set Y, and every element of the set Y is paired with exactly one element of the set X. This will only be true for the x=y=1 case and not otherwise.

  Thus if the number of elements are different in the both sets then there cannot be a way where we can have a 1-1 correspondence between the set X and the function X-¿Y.

3 Problem 2

With the expression sub-language extended with a division operator changes we will be there in the operational semantics.
We are interested in the natural style semantics only where it is the semantics that describes the overall results of the execution. We are trying to find out the difference in meaning of the programming language when there is an division operator.
Assume that there are two numbers for division b1 $\div$ b2 = b3
Our solution uses the multiplication operator to describe the division operator. For a particular combination of b1 and b2 we can find a number b3 that will satisfy the equation b2 $\star$ b3 = b1.
For the special case where there is no number that satisfies the equation, it is the case where the denominator is zero or a 0 $\div$ 0 case. As it is of the form 0 $\div$ 0 or anything $\div$ 0 value of b3 in the case is inf. The equation holds for real numbers too.
Evaluation Rules:

$$\frac{< e_1, \sigma > \Downarrow n1 < e_2, \sigma > \Downarrow n2}{< e1/e2, \sigma > \Downarrow n}$$

Where n= n1/n2 and n2!=0
For n2 =0 we introduce a new literal "inf" which means undefined,
So,

$$\frac{< e_1, \sigma > \Downarrow n1 < e_2, \sigma > \Downarrow inf}{< e1/e2, \sigma > \Downarrow inf}$$

As a result the evaluation rules for other operations are found to be:
$< e1 + e3, \sigma > \Downarrow inf$
$< e1 - e3, \sigma > \Downarrow inf$
$< e1 * e3, \sigma > \Downarrow inf$
$< x := e3, \sigma > \Downarrow inf$

Although in the abstract syntax the arithmetic expressions which is added:
$e ::= n \mid x \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \mid e_1/e_2$
NOTE: for $< e_1, e_2 > \in$ Aexp and $< e_1, e_2$ *is not* $\Downarrow 0 >$

The natural style semantics is given as follows:

(Definition of the Division Rule)

$$\frac{< e_1, \sigma > \Downarrow k_1 \;\; < e_2, \sigma > \Downarrow k_2 \;\; < e_3, \sigma > \Downarrow k_3 \;\; B \textit{ is } k_2 \star k_3 = k_1}{< (e_2 \star e_3) = e_1, \sigma > \Downarrow B}$$

$$\cfrac{\cfrac{< B, \sigma > \Downarrow False \;\; < c_2, \sigma > \Downarrow \sigma'}{< if B \; then \; c_1 \; else \; c_2, \sigma > \Downarrow \sigma'}}{< e1 \div e2, \sigma > \Downarrow inf}$$

4 Problem 3

- Considering a new construct in the IMP language which is *let* $x = e$ *in* c. The informal construct states that the expression e is evaluated to have the value x within the lexical scope of c and also initialized with the result of evaluating e.

  To extend the natural style semantics we add the follwoing rules to the semantic judgement so that it deals with the let command. We will also make sure that the scope of the newly declared variable is handled.

  Rules to the Operational semantics:

  $< x, \sigma > \Downarrow K$: states the expression x evaluates to the value of K. This value that is assigned to the expression X before the "let" statement.

  $< c, \sigma[x \to K] > \Downarrow \sigma'$: states that we are assuming that the command "c" which has the lexical scope "c" evaluates to σ'. By executing such a command we update the state of the system from σ to σ'

  $< let \; x \; = \; e \; in \; c, \sigma > \Downarrow \sigma'[x \to \sigma(x)]$: states that the expression is evaluates the value of x and is being reverted back to the state where the previous (old) value of x outside the scope "c" was stated.

  Rules of x:

$$\frac{< x, \sigma > \Downarrow K \quad < c, \sigma[x \to K] > \Downarrow \sigma'}{< let \; x \; = \; e \; in \; c, \sigma > \Downarrow \sigma'[x \to \sigma(x)]}$$

  Redexes and contexes

  Local reduction rule: $< let \; x \; = \; e \; in \; c, \sigma > \to < x := e; c; x = \sigma(x), \sigma >$
  Redex: $< let \; x \; = \; H \; in \; c, \sigma >$

  Also to evaluate in a single step would be to substitue a number, which is : $< let \; x \; = \; 2 \; in \; c, \sigma >$

  Context would be to substitute H = . then it comes to:

  $< let \; x \; = \; . \; in \; c, \sigma >$

5 Problem 4

To prove ,
D :: $< while \; b \; do \; x = x + 2, \sigma > \Downarrow \sigma'$ If σ is even then $\sigma'(x)$ is also found to be even.

Base case is when b is found to be false, if that is the case then we have,

$$\frac{< b, \sigma > \Downarrow False \quad < c, \sigma > \Downarrow \sigma_0}{D_1 :: < while\ b\ \ do\ x = x + 2, \sigma > \Downarrow \sigma_0}$$

Which means that there is no change in the state of the system. Moving on to the case where the state actually changes when the value of b is actually True. The idea here is to prove that the proposition P(x) holds for all x which is defined as a sort of recursively defined structure like tree. There can be two sub parts of the structural inductions namely 1) Structure of expressions and 2) Structure of derivations. The decision tree for the expression would evaluate to:

Inductive Case

$$\frac{< b, \sigma > \Downarrow True \ \ < x = x + 2\ ;\ while\ b\ \ do\ x = x + 2, \sigma > \Downarrow \sigma_1(x)'}{D_1 :: < while\ b\ \ do\ x = x + 2, \sigma > \Downarrow \sigma_1(x)'}$$

We can see that σ' is also even as x got increased only by a value 2. Inducing this fact from the above decision tree we can tell that σ_1' = σ_0 hence σ_1'(x) is even and so is $\sigma_0(x)$'