Alex Jia CS 205
157000846

Homework 9

PART I - Equivalence Relations (Cont.)

Consider the domain of values D={1, 2, 3, 4}, and the binary relation
R4 = { (a,b) | a/b = 2^k for some natural number k=0,1,2,...} over D.
(i.e., a divided by b is a non-negative power of 2.)

I. *List systematically the pairs of values related by this relationship, by showing in order what values are related to 1, 2, 3, and 4.*

The following (a,b) pairs correlate to k= 1:
(2,1), (4,2) (6,3), (8,4), (10,5), (12,6), (14,7), (16,8)
The following (a,b) pairs correlate to k= 2:
(4,1), (8,2), (12,3), (16,4)
The following (a,b) pairs correlate to k= 3:
(8,1), (16,2)
The following (a,b) pairs correlate to k= 4:
(16,1)

II. *Show that the relation R4 is not an equivalence relation using Theorem 2/Section 9.5 on Equivalence Relations in the textbook (p.613) / class notes. In other words, do not check the reflexive, etc. properties of R4 directly, but consider instead the sets*
[c]_R4 = {x | c is related to x by R4}

Consider the set [c]_R4 to contain all tuples mentioned in the previous part:
[c]_R4 = { (2,1), (4,1), (4,2), (6,3), (8,1), (8,2), (8,4), (10,5), (12,3), (12,6), (14,7), (16, 1), (16,2), (16,4), (16,8) }
Realising that this set contains all the tuples that are valid with R4, we can prove that it, and by extension R4, is not an equivalence relation. This is because an equivalence relation ought to be reflexive, symmetric, and anti-symmetric. However, [c]_R4 is *not* symmetric, because none of its members are valid when their order is switched. For example, Take (4,1). In it's current state, 4 / 1 is equal to 4, which is 2^2, a valid option. However, switching the 1 and 4 to be (1,4) (which the definition of symmetry requires in order to be considered valid), would yield 1 / 4, which is not an integer, thereby breaking the validity of the tuple in regards to R4. Based off this reasoning alone, the set [c]_R4, and consequently the relation R4, are not equivalence relations.

PART II

I. *Find a loop invariant that will ensure that E implies POST, and show E -> POST.*

INV E: j :=1 + … + (2i-1); substituting ‘i’ with ‘n’, we get j := 1 + … + (2n-1)

1. For n:=1, INV E states that j:= 1, a valid statement (1 == 1^2)
2. Now suppose that INV E is true for some n := k>=1
3. This would imply that 1 + … + (2k-1) = k^2
4. Given n := k + 1, we get 1 + … + (2k-1) + (2k + 1) ?= (k+1)^2
5. Substituting statement 3, we simplify the equation to: k^2 + (2k+1) = (k+1)^2.
6. Thus, INV E is equivalent to the POST assumption.

II. *Which of the assertion(s) named (A thru E) are exactly the “loop invariant”?*

A, D, and E. (I don’t count B’, because it includes the !<conditional>).

III. *List the actual assertions for this proof besides each of the names A, B, B’, C, D, E.*

Assertion	Formula
PRE	n>0
E	j := 1 + … + (2i-1)
D	j := 1 + … + (2i-1)
C	j + (2i-1):= 1 + … + (2i-1)
B	j + (2i) := 1 + … + ((2i-1)+1)
B’	j := 1 + … + (2i-1) /\ (i != n)
A	j := 1 + … + (2i-1)

PART II (CHALLENGING)

i) Trace the program for a small value of N, and try to understand why it works.

Letting N:= 3; m:=2.
Post while-loop, s:= 0 + 1 + (1 + 1), k:=3.
Since N (which is 3) != m (which is 2), s:=s + N
s:= 3 + 3
s:=6

ii) State a loop invariant that you could use to prove the program achieves the postcondition if it terminates.

s:= 0 + ... + (2k+1)