Homework 2

COSC 560
Temporal Verification and Real-Time Systems

Due at the start of class on: 10/22/2019

Note: You can submit your homework through e-mail to Stanley.Bak@georgetown.edu. For written portions, you can also optionally hand them to me in class. Homework is due at the start of class. Late homework incurs a 50% penalty.

Problem 1 [Hoare Logic: Extensions] Consider the rules for Hoare logic, i.e., the rules for combining Hoare triples for inferring proerties of a given program. In this problem, we are going to extend Hoare logic for new constructs of programs.

1. Consider the following **if-then-elseif-else** statement

   if(C_1) then
   S_1;
   elseif(C_2) then
   S_2;
   elseif(C_3) then
   S_3;
   $\vdots$
   elseif(C_{k-1}) then
   S_{k-1};
   else
   S_k;

   Combine the existing proof rules in Hoare logic to infer the pre and post conditions of the above given **if-then-elseif-else** program construct. **[8 points]**

2. A variant of the **while** loop is the **do-while** loop which executes the statements in the while loop first and then check for the condition. Formally given as:

   {
   S
   }do-while(B)

   Combine the existing proof rules in hoare logic to infer the pre and post conditions for **do-while** loop program construct. **[7 points]**

Problem 2 [Weakest Preconditions, Strongest Postconditions, and Program Verification Using VCC]

1. Give the weakest preconditions for the following statements and the postconditions. **[10 points]**

   (a)
   $y = x^2 + 3y^2$
   {post-condition: $x < 42 \land y > 391$ }

   (b)
   if($2x + 3y - 10z < 42$) then
       $y = 3x + 12y$;
   else
       $y = 15x$;
   {post-condition: $z < 51 \lor 31x - 3y > 1024$ }

2. Give the strongest postconditions for the following statements and the preconditions. Your postconditions can have existential quantifiers. **[10 points]**

   (a)
   {pre-condition: $x < 42 \land y > 391$ }
   $y = x^2 + 3y^2$

   (b)
   {pre-condition: $z < 51 \lor 31x - 3y > 1024$ }
   if($2x + 3y - 10z < 42$) then
       $y = 3x + 12y$;
   else
       $y = 15x$;

3. Consider the following program which performs multiplication. Use the VCC online portal `http://rise4fun.com/vcc` to prove the correctness of the following program.

```
#include <vcc.h>
int multiply(int i, int j)
{
  int l, k;

  l = 0;
  k = 0;

  while(l < i)
  {
    l = l+1;
    k = k+j;
  }

  return k;
}
```

You are required to add assumptions, assertions, and loop invariants to prove that the returned value is the product of the two integers supplied. You can assume a precondition that $-10 \leq j \leq 10$. Avoid over-constraining the inputs (do not for example, use a precondition that $i = 0$). Your submission should include your code annotated with the assertions and assumptions (e-mail this to me) and also include a screenshot of the verified result. **[15 points]**

4. **(Bonus)** Using the same program as before, remove the restriction that $-10 \leq j \leq 10$, and instead prove correctness for as general of a precondition as possible. As before, e-mail the code to me and include a screenshot of the result. **[Bonus: 10 points]**

Problem 3 [Abstract Interpretation] Consider verification of programs where the values of variables are integers ($\mathbb{Z}$) by using an abstract domain of intervals of integers extended with $-\infty$ and ∞ (all values $[a, b]$ where $a, b \in \mathbb{Z} \cup \{-\infty, \infty\}$ with $a \leq b$). To perform verification soundly, we want to construct a Galois connection.

1. In the concrete domain, what are the types of the elements in the lattice? What are the top and bottom elements? **[5 points]**
2. In the abstract domain, what are the types of the elements in the lattice? What are the top and bottom elements? **[5 points]**
3. Formally define the most precise abstraction function and concretization functions α and γ, and prove your functions meet the conditions of a Galois connection. **[10 points]**
4. Define the addition, subtraction, multiplication, and division operations in the abstract domain. **[10 points]**
5. Consider the initial valuations of the integer variable x be in $1, 2, 3, 4, 5, 30$ and y be in $0, 7, 10$, using abstract interpretation, compute the overapproximation for the valuations of the variable z performed according to the following computations. **[9 points]**

(a) $z = 2x + \frac{y}{10}$.

(b) $z = (x^2 + 10) - 2y$.

(c) $z = \frac{x}{x-y}$.

6. Other than precision, what is the main disadvantage of performing verification with this abstract domain? **[6 points]**

Problem 4 [SMT Solvers] In this problem we will develop new algorithms for solving a theory on *less-than* over real numbers with monotonic functions. Consider the set of reals $\mathbb{R}$ and the $<$ relation with the usual meaning. A function $f : \mathbb{R} \to \mathbb{R}$ is said to be monotonic if $\forall x, y \in \mathbb{R}$, if $x < y$ then $f(x) < f(y)$. Consider a *less-than* theory which involves variables assigned to reals and assume that all the functions used in the theory are **monotonic**. In the signature of this theory, the terms are denoted as T and the formulas are denoted as ϕ. The precise syntax is given as follows:

$$T = x \mid f(T) \tag{1}$$

$$\phi = T < T \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \tag{2}$$

An example formula in the domain is of the following format:

$$(x < y) \wedge (y < z) \wedge ((f(z) < f(y)) \vee (f(x) < f(z)))$$

1. Given a formula in the theory of *less-than* over real numbers with monotonic functions, encode the problem of checking whether the formula is satisfiable as a boolean satisfiability problem. Describe the encoding and prove that the above formula is satisfiable. What is the worst case complexity to solve a formula of size n using your approach. **[15 points]**

2. Consider a special set of formulas where there are no negations or disjunctions in the formula. Can you provide a polynomial time solution for such formulas? **[10 points]**

<u>Hint:</u> See the class slides on the theory of equality and different way to solve the problems in the domain of theory of equality.

Problem 5 [Hard SAT Solving] **(Bonus)** In theory, SAT solving is NP-Complete, so no algorithm should be able to perform well on all input instances. The algorithms described in class (DPLL and CDCL) encoded with some additional optimizations in a tool like Z3, however, work well in practice for many problems with even hundreds of thousands of variables. A boolean SAT problem with even 300 variables will have 2^{300} rows in its look up table, more than the number of atoms in the universe (about 2^{265}). Using Z3py, your goal is to find one of these difficult SAT instances.

Create a SAT problem with exactly 300 boolean variables that takes Z3py longer than 10 seconds to solve. Provide the z3py script as well as a `.smt2` file encoding the problem instance (this can be produced using the `Solver.to_smt2()` method). **[Bonus: 10 points]**