Assignment on Logical Abstract Interpretation.

1. Give an example of a linear arithmetic formula ϕ that is satisfiable over rationals but not satisfiable over integers. ϕ should involve only integral constants.

2. Let n and m be relatively prime positive integers. What is the strongest atomic fact in the theory of uninterpreted functions that is implied by $y = F^n(y) \wedge y = F^m(y)$.

3. Give an example of a theory that is not stably infinite but that is convex.

   Aside: In class, we saw an example of a theory that is not stably infinite. However, that theory was also not convex. Quite often, theories that are not stably infinite are also not convex, but not always.

4. The purpose of this example is to demonstrate the importance of disjointedness condition required on theories T1 and T2 in the Nelson-Oppen combination methodology. Consider the following parity theory that shares constants (which can be treated as nullary functions) as well as the binary operators $\pm$ with the integer linear arithmetic.

$$\begin{aligned} \text{Expressions } e &:= y \mid c \mid e_1 \pm e_2 \\ \text{Atomic facts } g &:= \texttt{IsOdd}(e) \mid \texttt{IsEven}(e) \\ \text{Axioms: } &\texttt{IsOdd}(i) \text{ for any odd integer } i \\ &\forall e_1, e_2 : \texttt{IsOdd}(e_1) \wedge \texttt{IsEven}(e_2) \Rightarrow \texttt{IsOdd}(e_1 + e_2) \\ &\text{and so on.} \end{aligned}$$

   Give an example of a formula ϕ s.t.

   - ϕ is over combination of integer linear arithmetic and parity theory, (i.e., ϕ only uses the binary relation $\geq$ and unary relations `IsOdd`, `IsEven`).
   - ϕ is unsatisfiable.
   - Nelson-oppen combination methodology (in which we also share disjunction of equalities between variables) would fail to identify unsatisfiability.

5. Write down the generic transfer function for logical abstract interpreter for the non-deterministic assignment $x :=?$ in terms of existential quantification. In class, we saw that the generic transfer function for an assignment statement $x := e$ transforms the formula ϕ before the assignment into the formula $\exists x' : \phi[x'/x] \wedge (x = e[x'/x])$. Write a similar transformation for the non-deterministic assignment $x :=?$.

   Aside: Non-deterministic assignments are a commonly used abstraction for modeling those assignments in which the expression being assigned cannot be modeled by the underlying abstract domain.

6. Consider the abstract domain of conjunctions of difference constraints. For this domain, write down the results of the following operations (as would be computed by the algorithm described in the class).

   (a) $\texttt{Join}(x = 0 \wedge y = 1,\ x = 4 \wedge y = 6)$

   (b) $\texttt{Eliminate}(x - y \leq 3 \wedge y - z \leq 4,\ y)$

7. Consider the following program.

```
1  if (*) {
2      x := 0; y := 2;
3  }
4  else {
5      x := 2; y := 4;
6  }
7  if (x == 1) {
8      Assert(y = 3);
9      Assert(y = 6);
10 }
11 if (x == 3) {
12     Assert(y = 3);
13 }
```

   (a) Which assertions in the above program are valid?

   (b) Would performing abstract interpretation over the abstract domain of difference constraints (which was discussed in class) validate the valid assertions?

   (c) Write down the invariant required at program point 7 to prove validity of those assertions that cannot be proved valid by performing abstract interpretation over the abstract domain of difference constraints.

8. Consider the following program.

```
1 y := 0;
2 if (x == 1) {
3     y := 10;
4 }
5 if (x == 1) {
6     Assert(y ≥ 10);
7 }
```

   (a) Would performing abstract interpretation over the abstract domain of difference constraints (which was discussed in class) validate the assertion in the above program?

   (b) What is the invariant needed at program point 5 to validate the assertion?

   Aside: An analysis that can validate the above assertion would be called a *path-sensitive analysis*.

9. Consider the abstract domain of conjunctions of equalities between uninterpreted function terms. For this domain, write down the results of the following operations (as would be computed by the algorithm described in the class).

   (a) `Join`$(y = F^4(y),\ y = F^6(y))$

(b) `Eliminate`$(x = F^3(x) \wedge x = F^5(x) \wedge y = F^4(x),\ x)$

10. The domain of conjunctions of atomic facts over the theory of uninterpreted functions is not closed under disjunction. For example, consider the following two facts:

$$E_1: \quad x = y$$
$$E_2: \quad x = F(x) \ \wedge \ y = F(y) \ \wedge \ G(x) = G(y)$$

(a) The number of independent atomic facts that are implied by both E_1 and E_2 individually is infinite. Write down one such infinite family of atomic facts.

(b) The transfer function for join that we described in class for the theory of uninterpreted functions is thus not complete, i.e., it does not generate all atomic facts that are implied by each of the inputs to the join algorithm. Write down the result of the join transfer function that we studied in the class for the above example.

Aside: However, the join algorithm that we discussed in class is complete for the case when there are no cyclic dependencies like $x = F(x)$ and this leads to a PTIME algorithm for assertion checking in presence of non-deterministic conditionals since cyclic dependencies can arise only in presence of deterministic conditionals.

11. Consider the following program.

```
a1 := 0;  a2 := 0;
b1 := 1;  b2 := F(1);
c1 := 2;  c2 := 2;
while(*) {
      a1 := a1 + 1;  a2 := a2 + 2;
      b1 := F(b1);  b2 := F(b2);
      c1 := F(2c1 − c2);  c2 := F(c2);
}
Assert(a2 = 2a1);
Assert(b2 = F(b1));
Assert(c2 = c1);
```

(a) For each assertion in the above program, write down the inductive loop invariant required to validate the assertion.

(b) Which of these assertions can be validated by which of the following abstract domains: Difference Constraints (discussed in class), Linear Equalities (Karr, 1976), Linear Inequalities (Cousot, POPL 1978), Uninterpreted Functions (Gulwani, Necula SAS 2004; discussed in class), Combination of any two of these (Gulwani, Tiwari, PLDI 2006; discussed in class).

12. The combined domain of atomic facts is not closed under disjunction even if the individual domains of atomic facts are closed under disjunction. For example, consider

the following two facts:

$$E_1 : \quad x = 0$$
$$E_2 : \quad x = 1$$

(a) Write down the set of all atomic facts that are implied by both E_1 and E_2 individually in the theory of linear arithmetic.

(b) Write down the set of all atomic facts that are implied by both E_1 and E_2 individually in the theory of uninterpreted functions.

(c) The number of independent atomic facts that are implied by both E_1 and E_2 individually in the combined theory of linear arithmetic and uninterpreted functions is infinite. Write down one such infinite family of atomic facts.

(d) The transfer function for join that we described in class for combined domain in terms of the join transfer function for individual domains is thus not complete. Write down the result of the join transfer function that we studied in the class for the above example.

Aside: However, the join algorithm that we discussed in class is *partially* complete; it generates all atomic facts that involve terms that are semantically represented in both the inputs.

13. Consider the universally quantified abstract domain over the base domain of combination of linear arithmetic and uninterpreted functions. For this domain, we would like to compute join of the following two facts using the algorithm discussed in class.

$$E_1 : \quad (i = 0) \wedge \forall k(\texttt{false} \Rightarrow F[k] = 0)$$
$$E_2 : \quad (i = 1) \wedge \forall k(k = 0 \Rightarrow F[k] = 0)$$

(a) First compute $\lfloor (i = 0 \Rightarrow \texttt{false}) \ \wedge \ (i = 1 \Rightarrow k = 0) \rfloor$, i.e., compute a weakest conjunction of atomic facts ϕ such that

$$\phi \Rightarrow ((i = 0 \Rightarrow \texttt{false}) \ \wedge \ (i = 1 \Rightarrow k = 0))$$

(b) Now using ϕ, write down the result of $\texttt{Join}(E_1, E_2)$.

Aside: In the array init example that we discussed in class, we had unrolled one loop iteration. If we do not perform any loop unrolling, we can still discover the desired invariant and one of the join operations that we will need to perform is the one in the above example.