cs4610 - Fall'19 — Pettit
due 9/29/19 @ noon via Collab

HW3.cab8xd

Honor Code Notice Collaboration is a great way to learn, therefore you are encouraged to help one another out, but you must not cheat. When you cheat you only harm yourself. When collaborating with other students you may get help understanding course content, but may not get help completing course requirements unless specifically stated on the assignment. *See syllabus for the full honor statement.*

Answer the following questions. Turn in a single pdf file that contains all of your answers.

1. Suppose there are three variables X, Y, and Z with these types:

   X : *integer that is divisible by* 3
   Y : *integer that is divisible by* 12
   Z : *integer*

   For each of the following assignments, knowing nothing about the values of the variables except their types, answer whether a language system can tell before running the program whether the assignment is safe. Why or why not?

   (a) X := Y Yes, the assignment is safe; integers divisible by 12 (Y) are also integers divisible by three (X).
   (b) X := X Yes, the set or type of X includes itself by definition.
   (c) Y := Y + 1 No. The Y type contains only integers divisible by twelve. Integers in the set of Y + 1 (ex. 13) are not divisible by 12.
   (d) Z := X Yes, integers divisible by 12 (X) are included in in the integer type (Z).
   (e) X := Z No, X cannot contain all integers (Z), only those divisible by 12.
   (f) X := X + 3 Yes, X can contain integers divisible by 3 and X + 3 includes those integers (ex. X = 3, X + 3 = 6 would work).
   (g) X := X + Y Yes, Yes, X can contain integers divisible by 3 and X + Y includes those integers (ex. X = 3, Y = 12, X + Y = 15 would work).

2. Design programming-language type constructors based on set intersection and set complement. What would the representations and sets of supported operations be like? Why do you think programming in languages do not usually have such types? *(Good answers for this question will probably take about half a page of space.)*

The constructions of types based on set intersections and set complements:

$$I = X \cap Y$$

$$C = X^c$$

Where I is a type based on intersections, C is a type based on a set complement, and both X and Y are sets of a generic type.

The variance of the type definitions of these sets would depend on whether a programming language is strict or loose when handling types. And, the languages would have to be able to answer the questions:

* For intersections, what defines overlapping types or subsets shared by different sets?

* For complements, what is the opposite or not included in a given type? Ex. How would a language define the complement of an 'int' variable or some other constructed variable?

* For either type, how would each type's instance relate to overlapping type operator or define their own?

I believe answering these questions would require an immense amount of thought on the developer's end along with additional explanation and proofs via documentation to justify the developer's decisions. The resulting "vagueness" of the intersection and complement definitions (i.e. defining the equality and opposites of sets to work with the types in a programming language) would create massive overhead, and the final product would struggle to encapsulate a universal definition that users would employ efficiently.