Principles of Programming Languages - Homework 11

Abhi Agarwal

1 Problem 1

(a)

$t_0 <: t_1$: type t_0 can be safely substituted by values of type t_1.

(i) True: Number can safely be substituted by values of Number. Follows the SubRefl rule.

(ii) False.

(iii) True: Number can safely be substituted by values of Any. Rule SubAny.

(iv) True: Var to Const is permitted by rule SubObjMut, and Number to Any by rule SubAny.

(v) False.

(vi) False.

(vii) False.

(viii) True.

(ix) False.

(b)

(i) (1): It will safely evaluate. It will produce a value. The value that fun(x).f would return would be 4. (2): TypeCall requires that the type of the argument in a call expression precisely matches the type of the function parameter that it is passed to. So it will not be well-typed with subtyping since that is not the case. The const x is missing the field g (which is a boolean). Also, x (declared on line 1) is not a subtype of y (in the parameter declared on line 2).

(ii) (1): It will safely evaluate. It will produce a value. The value that fun(x).f would return would be 3. (2): It is well-typed with subtype since x (declared on line 1) is a subtype of the parameter y (declared on line 2).

(iii) (1): It will not safely evaluate. It will get stuck or throw an error because we use static typing, and not dynamic typing so the field g is not present to y on line 3, and so would return with only the field f inside that object. When the field g is looked on line 5 it won't be able to find it. (2): It is well-typed with subtype since x (declared on line 1) is a subtype of the parameter y (declared on line 2).

(iv) (1): It will safely evaluate. It will produce a value. The value would be 1. (2): It is well-typed. Both of the branches x and y can be resolved to any common supertype {f: any, g: any}.