Model Checking, SS2011: Exercise Sheet 13

June 30, 2011

Exercise 13.1. Construct a liquid type derivation for each of the following typing propositions.

1. `incr` $<:$ `(x:int -> {v : int | v = x + 1})` $\vdash$ `incr 1` $<:$ `{v : int | v = 2}`
2. $\emptyset \vdash$ `let incr x = x + 1 in incr 1` $<:$ `{v : int | v = 2}`
3. $\emptyset \vdash$ `fun x -> fun y -> x - y` $<:$ `x:int -> y:int -> {v : int | v = x - y}`
4. $\emptyset \vdash$ `let d = fun x -> y -> x - y in d 1 2` $<:$ `{v : int | v = -1}`
5. $\emptyset \vdash$ `if true then 0 else 1` $<:$ `{v : int | v = 0` $\vee$ `v = 1}`
6. $\emptyset \vdash$ `let rec fact x = if x <= 0 then 1 else x * fact(x - 1) in fact 2` $<:$ `{ v : int | v = 2}`

Exercise 13.2. Construct a set of liquid typing constraints for each of the following typing propositions. Note that an expression of type `unit` performs some computation and returns the unit value `()`.

1. `incr` $<:$ `(x:int -> {v : int | v = x + 1})` $\vdash$ `let i = incr 1 in assert(i > 1)` $<:$ `{v : unit | true}`
2. $\emptyset \vdash$ `let incr x = x + 1 in assert(incr 1 = 2)` $<:$ `{v : unit | true}`
3. $\emptyset \vdash$ `fun x -> fun y -> assume(x > 0` $\wedge$ `x > y); assert(x - y > 0)` $<:$ `x:{ v : int : x > 0} -> y:{v : int | x > y} -> {v : unit | true}`
4. $\emptyset \vdash$ `let d = fun x -> y -> x - y in assert(d 1 2 = (-1))` $<:$ `{v : unit | true}`
5. $\emptyset \vdash$ `assume(x = true); assert((if x then 0 else 1) = 0)` $<:$ `{v : unit | true}`
6. $\emptyset \vdash$ `let rec fact x = if x <= 0 then 1 else x * fact(x - 1) in assert(fact 2 = 2)` $<:$ `{ v : unit | true}`

Exercise 13.3. Let e range over expressions. Give liquid typing rules for expressions `assume(` e `)`, and `assert(` e `)`.

Exercise 13.4. Construct a liquid type derivation for each of the typing propositions in Exercise 13.2..