CS 231 : Types and Programming Languages
Homework #4

Ronak Sumbaly
UID : 604591897

December 4, 2015

Question 1 - Type and Effect Rules

Part a. Augment Typing Rules

$$\frac{}{\Gamma \vdash \text{throw} : \text{T} \;;\; \{\text{exn}\}} \qquad \text{(T-Throw)}$$

$$\frac{\Gamma \vdash \text{t}_1 : \text{T} \;;\; \Phi_1 \qquad \Gamma \vdash \text{t}_2 : \text{T} \;;\; \Phi_2}{\Gamma \vdash \text{try } \text{t}_1 \text{ catch } \text{t}_2 : \text{T} \;;\; \Phi_2} \qquad \text{(T-Try)}$$

Part b. Typing Derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma(x) = \mathit{Bool}}{\text{x : Bool} \vdash \text{x : Bool} \;;\; \emptyset}\text{T-Var} \quad \dfrac{}{\Gamma \vdash \text{true : Bool} \;;\; \emptyset}\text{T-True} \quad \dfrac{}{\Gamma \vdash \text{throw : T} \;;\; \{\mathit{exn}\}}\text{T-Throw}}{\Gamma, \text{x : Bool} \vdash \text{if x then true else throw : Bool} \;;\; \{\mathit{exn}\}}\text{T-If}}{\Gamma \vdash \text{function x:Bool} \rightarrow \text{if x then true else throw : Bool} \xrightarrow{\{\mathit{exn}\}} \text{Bool} \;;\; \emptyset}\text{T-Fun} \quad \dfrac{}{\Gamma \vdash \text{true : Bool} \;;\; \emptyset}\text{T-True}}{\Gamma \vdash \text{(function x:Bool} \rightarrow \text{if x then true else throw) true : Bool} \;;\; \{\mathit{exn}\}}\text{T-App} \quad \dfrac{}{\Gamma \vdash \text{false : Bool} \;;\; \emptyset}\text{T-False}}{\Gamma \vdash \text{try ((function x:Bool} \rightarrow \text{if x then true else throw) true) catch false : Bool} \;;\; \emptyset}\text{T-Try}$$

Question 2 - Subtyping

a. `Top` → `Ref Top` ∧ `Bool` and `(Top` → `Top)` ∧ `Top`
Answer: Yes.

b. `(Ref Top)` → `Top` and `Top` → `Top`
Answer: No.
Term: `(function (x : Ref Top)` → `!x) 42`

c. `Ref Top` and `Ref (Top` ∧ `Top)`
Answer: No.
Term: `snd !(Ref 42)`

d. `Ref (Top` ∧ `Top)` and `Ref Top`
Answer: No.
Term: `! Ref(42` ∧ `3)` + `1`

Question 3 - Constant Propagation

Part a. Typing Rules

$$\frac{}{\Gamma \vdash \mathrm{n} : \mathrm{n}\ \mathrm{Int}} \qquad \text{(T-Constant)}$$

$$\frac{\Gamma \vdash \mathrm{t}_1 : \mathrm{n}_1\ \mathrm{Int} \quad \Gamma \vdash \mathrm{t}_2 : \mathrm{n}_2\ \mathrm{Int} \quad \mathrm{n}_1\ [[+]]\ \mathrm{n}_2 = \mathrm{q}}{\Gamma \vdash \mathrm{t}_1 + \mathrm{t}_2 : \mathrm{q}\ \mathrm{Int}} \qquad \text{(T-Add-1)}$$

$$\frac{\Gamma \vdash \mathrm{t}_1 : \mathrm{any}\ \mathrm{Int} \quad \Gamma \vdash \mathrm{t}_2 : \mathrm{any}\ \mathrm{Int}}{\Gamma \vdash \mathrm{t}_1 + \mathrm{t}_2 : \mathrm{any}\ \mathrm{Int}} \qquad \text{(T-Add-2)}$$

$$\frac{\Gamma \vdash \mathrm{t}_1 : \mathrm{n}\ \mathrm{Int} \quad \Gamma \vdash \mathrm{t}_2 : \mathrm{any}\ \mathrm{Int}}{\Gamma \vdash \mathrm{t}_1 + \mathrm{t}_2 : \mathrm{any}\ \mathrm{Int}} \qquad \text{(T-Add-3)}$$

$$\frac{\Gamma \vdash \mathrm{t}_1 : \mathrm{any}\ \mathrm{Int} \quad \Gamma \vdash \mathrm{t}_2 : \mathrm{n}\ \mathrm{Int}}{\Gamma \vdash \mathrm{t}_1 + \mathrm{t}_2 : \mathrm{any}\ \mathrm{Int}} \qquad \text{(T-Add-4)}$$

$$\frac{\Gamma \vdash \mathrm{t} : \mathrm{n}\ \mathrm{Int} \quad [[\text{-}]]\ \mathrm{n} = \mathrm{q}}{\Gamma \vdash \text{-}\mathrm{t} : \mathrm{q}\ \mathrm{Int}} \qquad \text{(T-Negation-1)}$$

$$\frac{\Gamma \vdash \mathrm{t} : \mathrm{any}\ \mathrm{Int}}{\Gamma \vdash \text{-}\mathrm{t} : \mathrm{any}\ \mathrm{Int}} \qquad \text{(T-Negation-2)}$$

Part b. Code Analysis

```
                    {}
x := 8;
                    {x : 8 Int} Program Point
if(x > 0)
  y := x + x;
                    {y : 16 Int, x : 8 Int}
  x := x + 1;
                    {y : 16 Int, x : 9 Int}
else
  y := 0;
                    {y : 0 Int, x : 8 Int}
  x := x + 1;
                    {y : 0 Int, x : 9 Int}
                    {y : any Int, x : 9 Int} Program Point
while(x > 0) do
  x := x - 1;
  y := 5;
                    {x : 0 Int, y : 5 Int} Program Point
```

Question 4 - Weakest Precondition

Part a. WP Computation at every program point

Bottom - Up Approach of Evaluation

```
if (x ! = null) then
  n := x.f;
                 WP(n := x.f, (a != null && n >= 0 && n < a.length))
                      = (a != null && x.f >= 0 && x.f < a.length)
else
  n := z-1;
                 WP(n := z-1, (a != null && n >= 0 && n < a.length))
                      = (a != null && z > 0 && z <= a.length)
res := a[n];
                 WP(res := a[n]) = a != null && n >= 0 && n < a.length

WP(IF, WP(res := a[n], true)) = WP(IF, a != null && n >= 0 && n < a.length)
                              = ((x != null) => WP(n := x.f, n >= 0 && n < a.length))
                               && ((x = null) => WP(n := z-1,n >= 0 && n < a.length))
                              = ((x != null) => a != null && x.f  >= 0 && x.f < a.length)
                                && ((x = null) => a != null && z >= 1 && (z-1) < a.length)
```

Part b. Loop invariant

Loop Invariant: **I** $= \{r = k^2, s = 2k+1, k <= n\}$

Properties satisfied by the loop invariant:

1. `I && (k = n) => r = n`^{2}
2. `I && (k != n) => WP(s, I)`
   where `s = {`

```
        r := r + s;
        s := s + 2;
        k := k + 1; }
```

Question 5 - Linear Temporal Logic

Part a. Liveness Property of Queueing System

Liveness property: $\forall x, F(G(\neg P(x))) \rightarrow F(T(x))$

Part b. Liveness Property of New Queueing System

Assumption: Token(x) : returns the token number of x
Liveness property: $\forall x, \forall y \; where(x \neq y) \wedge (Token(y) < Token(x)), F(FG(T(y)) \vee P(y)) \rightarrow F(T(x))$