Midterm 3 (22C:123)

Open Books and Notes

1. (50) **(Algebraic Semantics)** On page 490 of the textbook, the module SymbolTable is defined as an instance of Mapping. You are asked to add the function *table-union* (*tu* for short), which is used in the attribute grammar of Wren (on page 79), into this module.

   (a) Please formally define *tu*, which should have the same behavior as the one given on page 79.

   Answer: We add the following into Module `WrenTypeChecker` on pages 490-491:

```
operations
    tu: SymbolTable, SymbolTable --> SymbolTable

variables
    (same as on page 491)

equations
    tu(nullSymTab, symtab) = symtab
    tu(update(name, type, symtab1), symtab) =
                      update(name, type, tu(symtab1, symtab))
           when apply(symtab, name) = ErrorType
    tu(update(name, type, symtab1), symtab) = tu(symtab1, symtab)
           when apply(symtab, name) != ErrorType
```

   (b) Please decide whether the following two properties are true; if it is true, give a complete formal proof, otherwise, give a counter-example.

$$tu(t1,\ t2) = tu(t2,\ t1)$$
$$tu(t1,\ tu(t2,\ t3)) = tu(tu(t1,\ t2),\ t3)$$

   where *t1, t2, t3* are variables of SymbolTable.

   Answer: (1) In general, tu(t1, t2) is NOT equal to tu(t2, t1).

   Couterexample: Let t1 = update("x", naturalType, nullSymTab),
   and t2 = update("y", naturalType, nullSymTab). Then
   tu(t1, t2) = update("x", naturalType, update("y", naturalType, nullSymTab));
   tu(t2, t1) = update("y", naturalType, update("x", naturalType, nullSymTab)).

   (2) In general, tu(t1, tu(t2, t3)) = tu(tu(t1, t2), t3). We prove this by the structural induction on variable t1.

   Basic case: t1 = nullSymTab.

   Left = tu(nullSymTab, tu(t2, t3)) = tu(t2, t3).

   Right = tu(tu(nullSymTab, t2), t3) = tu(t2, t3).

   Inductive case: t1 = update(name, type, t4).

   As the induction hypothesis, we have tu(t4, tu(t2, t3)) = tu(tu(t4, t2), t3), and we assume that name does not appear in t4.

There are two subcases:

Case 1: name is neither in t2 nor in t3. That is, apply(t2, name) = ErrorType and apply(t3, name) = ErrorType.

In this case, we can prove that the following lemma holds:

Lemma 1: apply(tu(t2, t3), name) = ErrorType

Left = tu(update(name, type, t4), tu(t2, t3))
= update(name, type, tu(t4, tu(t2, t3)) (by Lemma 1)
Right = tu(tu(update(name, type, t4), t2), t3)
= tu(update(name, type, tu(t4, t2)), t3)
= update(name, type, tu(tu(t4, t2), t3))
= update(name, type, tu(t4, tu(t2, t3)) (by Induction Hypothesis)
= Left

Case 2: name is in t2 or in t3. That is, apply(t2, name) != ErrorType or apply(t3, name) != ErrorType.

Lemma 2: apply(tu(t2, t3), name) != ErrorType

Left = tu(update(name, type, t4), tu(t2, t3))
= tu(t4, tu(t2, t3)) (by Lemma 2)

There are two more subcases:

Case 2.1: apply(t2, name) != ErrorType

Right = tu(tu(update(name, type, t4), t2), t3)
= tu(tu(t4, t2), t3)
= tu(t4, tu(t2, t3)) (by Induction Hypothesis)
= Left

Case 2.2: apply(t2, name) = ErrorType and apply(t3, name) != ErrorType.

Right = tu(tu(update(name, type, t4), t2), t3)
= tu(update(name, type, tu(t4, t2)), t3)
= tu(tu(t4, t2), t3)
= tu(t4, tu(t2, t3)) (by Induction Hypothesis)
= Left

This completes the proof.

2. (50) **(Axiomatic Semantics)** The following is a BabyWren program segment C which computes the ceiling of the square root of $n > 0$:

```
x := 1; y := 1;
while (y < n) do
  y := y + 2*x + 1;
  x := x + 1;
end while;
```

(a) Please state and establish the loop invariant formally and clearly.

Sketch of the answer: The loop invariant is

$$P(x, y) : (y = x^2) \wedge (x - 1)^2 < n$$

You need to prove that (i) P is true right before the loop; (ii) P is true right after the loop.

(b) Plases establish the total correctness of $\quad \{n > 0\} \;\; C \;\; \{x = \lceil \sqrt{n} \rceil\}$.

Sketch of the answer: For the partial correctness, we need to prove that $P(x, y) \wedge y \geq n$ imply $\{x = \lceil \sqrt{n} \rceil\}$. This is true, because, from $P(x, y) \wedge y \geq n$, we have $(y = x^2) \wedge (x - 1)^2 < n \wedge y \geq n$, or simply $x^2 \geq n$ and $(x - 1)^2 < n$, which is equivalent to $\{x = \lceil \sqrt{n} \rceil\}$.

For the total correctness, we need to prove that the while loop will terminate. Pick the natural numbers with the well-founded order $>$, it's easy to see that $n - x$ is a natural number after each loop and its value is decreasing after each loop.