CS615 Autumn 2005 Quiz 3 Solutions

1(a), (b) The solution to part (a) is on the left. The changes needed for part (b) are on the right.

```
      ap_n := m;                           b2 := b0; b3 := b1;
                                           /* all other vars  unchanged */
      A(ap_n);                             A(b2, b3);
      x := ret_A;                          b9 := b6 ? F : *;
                                           /* all other vars  unchanged */
      if (*) then
        assert ((x > 0) AND (m < 100));    assert (b9 AND !b0)
                                           b9 := true; b0 := false;
                                           /* all other vars  unchanged */
    L:  skip;
      else
        assert (!(x > 0) OR !(m < 100));   assert (!b9 OR b0);
                                           b9 := b0 := *;
                                           /* all other vars  unchanged */
        skip;

    A(fp_n)                                A(b4, b5)
      if (*) then
        assert(fp_n >= 100);               assert(b4);
                                           b4 := T; b5 := F;
                                           /* all other vars  unchanged */
        ret_A := 100;                      b6 := F; b7 := F; b8 := T;
                                           /* all other vars  unchanged */
    L1: return;
      else
        assert(!(fp_n >= 100));            assert(!b4);
                                           b4 := F;
                                           /* all other vars  unchanged */
        if (*)
            assert(fp_n <= 50);            assert (b5);
                                           b5 := T; b4 := F;
                                           /* all other vars  unchanged */
            ret_A := 0;                    b6 := T; b7 := T; b8 := F;
                                           /* all other vars  unchanged */
    L2:     return;
        else
            assert(fp_n > 50);             assert (!b5);
                                           b5 := F;
                                           /* all other vars  unchanged */
            ap_n := fp_n - 10;             b2 := (!b4 OR b5) ? F : *;
                                           b3 := b5 ? T : *;
                                           /* all other vars  unchanged */
            A(ap_n);                       A(b2, b3);
            ap_n := ret_A;                 b2 := b8 ? T :
                                                 (!b8 OR b6 OR b7) ? F : *;
                                           b3 := (b6 OR b7) ? T :
                                                 (!b7 OR b8) ? F : *;
                                           /* all other vars  unchanged */
            A(ap_n);                       A(b2, b3);
            return;
```

In constructing the Boolean program, we have used underapproximations to the $\mathcal{F}$ functions when translating `assert` and assignment statements. These underapproximations are chosen such that they suffice to show that location `L` is not reachable in the program. Of course, one could come with better approximations to $\mathcal{F}$ and that would give a perfectly valid Boolean program as well. However, the intent of using underapproximations is to use as simple forms of $\mathcal{F}$ as possible that still suffice for proving desired reachability properties. When we say `/* all other vars unchanged /*` in the Boolean program above, we mean `bi := bi ? T : !bi ? F : *` for all other boolean variables `bi`.

Note that to reach $\widehat{\text{L}}$ in the Boolean program, we need `b9 AND !b0` to be `true` after the invokation of `A` in the main function of the Boolean program. From the assignment statement following the invokation of `A` in the main function, we further note that if `b6` is `true`, then `b9` must be `false`, and hence location `L` cannot be reached.

Since we need `b0` to be `false` in order to reach `L` in the Boolean program, the invokation of `A` from the main function of the Boolean program must have its first parameter set to `false`. Since the first call to `A` is with the first parameter set to `false`, it is now easy to see that in every subsequent recursive call to `A` in the Boolean program, the first parameter is always set to `false`. Therefore, if we consider the execution of the Boolean program, either function `A` goes into an infinite recursion, or it returns from location `L2`. Hence, if the invokation of `A` from the main function returns at all, it must return from location `L2`, where `b6` is set to `true`. This, of course, causes `b9` to be set to `false` (see assignment statements following the invokation of `A` in the main functiona), and therefore location `L` is not reachable in the Boolean program.

Recall now that if a location is not reachable in a Boolean program, it is not reachable in the original program as well. Therefore, this Boolean abstraction preserves the reachability of `L` in the original program.

1(c) As discussed above, the Boolean program can indeed go into an infinite recursion. This can happen if, for example, `b0` and `b1` are initialized to `false` at the beginning of the program.

2 To prove $FGFGp \Rightarrow FGp$. Let π be a path satisfying $FGFGp$. In the following, quantified variables refer to indices in π, and $s_i \models p$ is used to denote atomic propostion p holds at the i^{th} state along π.

$\pi \models FGp$ implies $\exists x((x \geq 0) \ \wedge \ \forall y((y \geq x) \rightarrow (s_y \models p)))$.

Similarly, $\pi \models FGFGp$ implies $\exists i((i \geq 0) \ \wedge \ \forall j((j \geq i) \ \rightarrow \ \exists k((k \geq j) \ \wedge \ (\forall l((l \geq k) \ \rightarrow \ (s_l \models p))))))$

In the above formula, let us replace the existentially quantified variable k by a Skolem function $f(j)$ (since $\exists k$ is embedded within the outer $\forall j$). The above formula then simplifies to:

$\exists i((i \geq 0) \ \wedge \ \forall j((j \geq i) \ \rightarrow \ ((f(j) \geq j) \ \wedge \ (\forall l((l \geq f(j)) \ \rightarrow \ (s_l \models p))))))$

Now, let us instantiate j to i. Since the above formula is `true` for all j, it must also be `true` for $j = i$. Hence, the above formula implies

$\exists i((i \geq 0) \ \wedge \ ((i \geq i) \ \rightarrow \ ((f(i) \geq i) \ \wedge \ (\forall l.((l \geq f(i)) \ \rightarrow \ (s_l \models p))))))$

Since $(i \geq i)$ is a tautology, the above simplifies to
$\exists i((i \geq 0) \ \wedge \ (f(i) \geq i) \ \wedge \ (\forall l((l \geq f(i)) \ \rightarrow \ (s_l \models p))))$

Let $f(i) = x$. Then, $f(i) \geq i \ \wedge \ i \geq 0 \rightarrow x \geq 0$. Hence the above formula implies
$\exists x((x \geq 0) \ \wedge \ \forall l((l \geq x) \rightarrow (s_l \models p)))$.

This is exactly the first-order formulation for $\pi \models FGp$ as we had seen above. This completes the proof.