CS510 Assignement #4 Solution

May 1, 2017

1 Predicate Abstraction (20p)

In order to mitigate state explosion in explicit state model checking, predicate abstraction is often used to reduce state space. Counter-example guided refinement may be needed during the process.

```
void main (void)
{
   int    a, b;
   a=1;
   b=1;
   if (a > b) {
      a--;
   } else {
      a++;
   }

   assert(a>b);
}
```

(a) Starting with predicate `a>b`, apply predicate abstraction to the above program.

(b) Perform explicit state model checking on the abstract program, present your execution tree and the counter example, if there is one.

(c) If there is a counter example in (b), test if it is a counter example in the original program.

(d) If the counter example is bogus, refine your abstraction so that either you find a real counter example or show the correctess of the program.

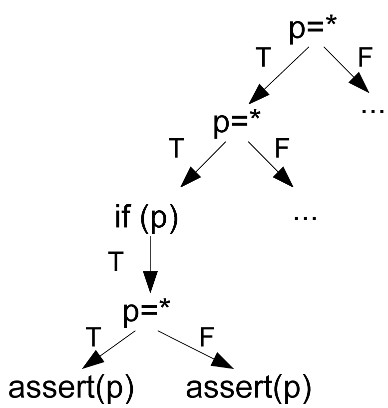
Answer:

(a) Assume `p` represents `a>b`

```
void main (void)
```

```
{
   bool p;
   p=*;
   p=*;
   if (p) {
      p=*;
   } else {
      p=*;
   }

   assert(p);
}
```



(b) The counter example is shown in the tree-like figure.

(c) However, note that the true branch in the original program cannot be taken. So this is a bogus counter example.

(d) As the contradiction occurs at the first three constraints, we refine our model with the first two constraints. Now we have three predicates: `p1` is for `a==1`, `p2` is for `b==1`, `p3` is for `a>b`.

```
void main (void)
{
   bool p;
   p1= T; //a=1
   p2= *;  //a=1
   p3= *; //a=1

   p2= T; //b=1
   p3=p1? F, *; //b=1

   if (p3) {
```

```
        p1=p1? F, *;
        p3=p1 & p2? F, *;
    } else {
        p1=p1?F, *;
        p3= p1 & p2 ?T, *;
    }

    assert(p);
}
```

The program always model-checks. Note that the refinement is not unique, you can also have `p1` for `a==b`, `p2` for `a>b`.