Homework 2

Please put your solutions into the mailbox of the instructor at the ground floor of Warren Weaver Hall until the deadline, or hand them in before class. Solutions to programming exercises should be submitted via email to the instructor with CC to ly603@nyu.edu.

The deadline for Homework 2 is February 18.

Problem 1 Spanning Trees (8 Points + 2 Bonus Points)

Spanning trees are an important concept in routing protocols. A *spanning tree* of a graph is a tree-like subgraph that covers all the nodes. Make this definition precise, and give an example of a graph with two distinct spanning trees.
Here is a template to help you:

```
module homework/spanningTree
pred isTree [r: univ->univ] { /* ...your constraints here */ }
pred spans [r1, r2: univ->univ] { /* ...your constraints here */ }
pred show [r, t1, t2: univ->univ] {
  spans [t1, r] and isTree [t1]
  spans [t2, r] and isTree [t2]
  t1 not = t2
}
```

Hint: It is up to you whether you consider the graph and trees to be directed or undirected. The undirected case is a bit trickier, and more interesting. If you solve the undirected case, you will get two bonus points.

Problem 2 Telephone Switch Connections (3 + 3 + 3 Points)

Consider the following model of connections in a telephone network:

```
module homework/phones
sig Phone {
  requests: set Phone,
  connects: lone Phone
}
pred show [] { /* ...your simulation constraints */ }
```

The signature `Phone` represents a set of telephones. For a phone `p`, `p.requests` is a set of phones that `p` is requesting connections to, some of which may have been granted, and `p.connects` is the phone that `p` is currently connected to (or none).

(a) Simulate the model by running the predicate `show`. Add some constraints to the predicate to ensure that you do not get boring cases; for example, you might say that there should be some requests and some connections.

(b) Add two facts to the model: 1) that every connection has a matching request (on the assumption that requests don't disappear until the connection they spawned are torn down), and 2) that there are no conference calls (in which a phone is involved in more than one connection).

(c) Now incorporate call forwarding by extending the state with a new relation `forward` from phones to phones, where `p.forward`, if non-empty, is the phone that an incoming call to `p` should be forwarded to. Change the constraint relating requests and connects to account for forwarding. Simulate some interesting examples of call forwarding, adding some extra simulation constraints, if necessary.

Problem 3 Goat, Cabbage, Wolf (8 Points)

A farmer wants to ferry across a river a goat, a cabbage, and a wolf, but his boat only has room for him to take one at a time. If he leaves the goat with the cabbage, the goat will eat it; if he leaves the goat with the wolf, the goat will be eaten. How does he do it? Solve the problem by modeling it in Alloy, and using the analyzer to find a solution.

Hint: the standard distribution of Alloy includes a module util/ordering that defines a total ordering. You may find it useful in ordering the steps the farmer takes to obtain a trace that shows the solution.

Problem 4 Bonus Exercise: Surgeon's Gloves (5 Bonus Points)

This is a famous problem by Paul Halmos. A surgeon must operate on three patients, but has only two pairs of gloves. There must be no cross-contamination: the surgeon must not come into contact with the blood of any patient, and no patient must come into contact with the blood of another patient. The surgeon needs two hands to work. How does she do it? Express this problem in Alloy, and use the analyzer to find a solution.