CS 512, Formal Methods, Spring 2014 Instructor:Assaf Kfoury

Technical Report

May 5, 2014 Noah McCarn and Shan Sikdar

1 Simple Graph

First we needed to formulate a model of a graph that captures the concepts of node, edge, path, and directness. First we created two signatures, node and edge that are abstract as to easily implement each node and edge when encoding the Konigsburg bridge problem. Node took no parameters and Edge took one parameter, connects. Connects is a set of exactly two Nodes. We also created another signature called DirEdge that also is abstract. DirEdge would be implemented if you wanted a directed graph whereas Edge would be implemented if one would want an undirected graph.

To implement a path of the graph, we constructed two signatures, Path and Step. Path took one parameter which was a Step. Step took four parameters, from and to which are Nodes, via which is an Edge and nextStep which is of type Step and set the nextStep to be taken in the path. The fact, all curr : Step, next : curr.nextStep | next.from = curr.to, ensures that the destination of the current step is the start of the next step. The function steps return the next steps. Finally, we implement a predicate for Alloy to find a model given the constraints called path.

2 Seven Bridges

Formulate the concepts of is connected and has a Eulerian path . Encode the Konigsburg bridge problem (shown below) using Alloy initializers and demonstrate that it is connected, but does not have any

Eulerian path. Create the smallest possible modification to the Konigsburg bridge problem such that the modified version is connected and has a Euleria path.

(From wikipedia: In graph theory, an Eulerian trail (or Eulerian path) is a trail in a graph which visits every edge exactly once.)

(a) **Formulation.**

(i) **Connectedness**
We defined connectedeness as there exists a path for all nodes such that all nodes are in this path. We then tested on different cases. We first had one isolated node in the graph and alloy returned that the model was inconsistent. We then created a graph that contained two connected components that were disjoint. Alloy still returned that the model was inconsistent. Finally we created a connected graph and alloy found an instance. After that we encoded the seven bridges graph and alloy found an instance. Therefore the seven bridges contains a path that contains all nodes. Therefore the seven bridges problem is connected.

(ii) **Eulerian Path.**
To formulate the concept of having an eulerian path, we first defined what it means to have a Path across different edges. (This is done above). To make sure that the path crossed all edges we had a predicate that made sure that there was one path that went through edges. (So that there was a path that didn't simply go from one node to itself forever).
To ensure that the path crossed each edge exactly once, we limted running the path to seven and see that it run for exactly one path. (run path for 7 but exactly 1 path). When the model runs it couldn't find an instance and therefore we know the predicate is inconsistent. Therefore the graph has no Eulerian graph.

(b) **The smallest possible modification.**

From wikipedia: For the existence of Eulerian trails it is necessary

that zero or two vertices have an odd degree. Looking at the graph for the seven bridges, we see that degrees of the nodes are 3,5,3,3. So if we add or subtract an edge we should be able to get a Eulerian trail. So we tried adding an edge between East and West will create an eulerian path. Changing the code to have run show() for 8 but 1 path, alloy can find an instance of the model, and say it's consistent. Therefore we know our new formulation has an Eulerian path.

Acknowlements and References:

(i) Software Abstractions. (Daniel Jackson)

(ii) A Guide to alloy. (Wong,Herrman,Tayeb)
found here:
`https://www.doc.ic.ac.uk/project/examples/2007/271j/suprema_on_alloy/Final%20Report/LaTeX/report.pdf`