Forge 2

Due: 11:59 Sunday, February 16, 2020

You should not use the Int type and the counting operator (#) in Forge for all problems. (Generally you should avoid them if there is an alternative because Int does not scale well in Forge.)
Your solutions for this assignment should terminate within 10 seconds (don't panic if it takes longer! Please reach out to a TA though).

Stencil Files

We have provided four stencil files for the four problems. They can be found in `/course/cs1950y/pub/forge2`. You **must** follow the stencil.

Problem 1: Undirected Tree Properties

A common definition of a tree used in the domain of graph theory is an undirected tree. In an undirected graph, nodes are connected by edges that have no orientation. That is, an edge from a node A to a node B is also an edge from B to A. An undirected graph can be represented as a binary relation, constrained to be symmetric.
An undirected tree is an undirected graph in which there is exactly one path between any two distinct vertices. This is equivalent to stating that undirected trees are acyclic (lacking cycles) and connected (can you see why?).
Complete the stencil file. For this problem, and all the following problems, you are permitted to use any mixture of quantifiers and relational logic.

Hint: a graph must contain a cycle if for some edge `e = (u,v)`, there is a path from `u` to `v` that does not use `e`.

Problem 2: Spanning Trees

A spanning tree of a graph is a subgraph that is an undirected tree and that covers all the nodes in the original graph. Based on this definition, create a model in Forge that will produce instances of a graph with two distinct spanning trees.
The stencil contains the following predicates you will need to complete:

- `pred isUndirectedTree (graph: Node -> Node)`
- `pred spans (graph1: Node -> Node, graph2: Node -> Node)`
- `pred twoSpanningTrees`

where:

- `isUndirectedTree[graph]` means `graph` is an undirected tree. You should be able to simply use the code from Problem 1 (and rename `edges` to `graph`).

- `spans[graph1, graph2]` means that `graph1` spans `graph2` (`graph1` is a subgraph of `graph2` and also covers all the nodes in `graph2`)

We will grade each predicate separately, except `isUndirectedTree` since it will be graded already in Problem 1.
Spanning trees have many uses. In networks, they're often used to set up connections. In the Firewire protocol, for example, a spanning tree is automatically discovered, and the root of the tree becomes a leader that coordinates communication.

Problem 3: Graph Coloring

Given a directed graph with colored edges, write a predicate that tells a if subgraph of same-colored edges constitutes a directed tree. You may find it helpful to look back on the directed tree problem in the Forge 1 assignment.

Problem 4: Rings

Some communication protocols organize nodes in a network into a ring, with links from node to node forming a circle. Characterize, as simply and concisely as you can, the constraints on `next`, the relation from `Node` to `Node`, that ensures that it forms a directed ring.
Fill in the stencil file, and then see if the instances you obtain are indeed rings.

Handing In

Run `cs1950y_handin forge2` from a directory holding your 4 Forge files (`undirected_tree.rkt`, `spanning_tree.rkt`, `colored_graphs.rkt`, `ring.rkt`). You should receive an email confirming that your handin has worked.