CS 290C – Spring 2013 – Homework Assignment 2

Due Thursday, May 16th

Do not discuss the problems with anyone other than the instructor.

Submission Instructions: Email the zipped directory containing your solutions (subject line: 290C HW2). Name the files as pn.als (should contain the Alloy specification for problem n), pn.out (should contain the output generated by Alloy for the properties you checked for problem n). Include a readme file to explain your solutions if necessary.

1. Consider a **Doubly Linked List (DLL)**. If a DLL is not empty, its **head** is a **node** that does not have a **prev** node. If a DLL is not empty, its **tail** is a node that does not have a **next** node. An empty DLL does not have a head node. The **contents** of a DLL is the set of nodes that are reachable from the head node by following the next links, plus the head node. The tail node is included in the contents. For all nodes that are reachable from the head node, the next of the prev of the node is itself.

Write the above DLL specification in Alloy.

Specify and check or simulate the following properties for scopes 2 and 3 using the Alloy Analyzer: 1) There exist DLLs with 0, 1, 2, and 3 nodes. 2) A DLL does not have a tail node if and only if it is empty. 3) For all DLLs, if head and tail are the same node, then the size of the DLL is 1. 4) No node in a DLL is the prev of two different nodes or the next of two different nodes. 5) There are no cycles (i.e., a node is not reachable from itself by just following next links or by just following prev links).

2. Extend the above DLL specification in Alloy by writing the predicates for the **add** and **delete** operations. Assume that the add operation adds the new node to the head of the DLL (i.e., the new node becomes the new head) and the delete operation removes the tail node (i.e., the prev of the tail becomes the new tail).

Hint: Specify prev and next as relations (using the cross product `->`).

Specify and check or simulate the following properties for scopes 2 and 3 using the Alloy Analyzer: 1) It is possible to obtain an empty DLL after a delete. 2) After an add the DLL contains at least one node. 3) Adding a node to a DLL increases the size of its contents by one. 4) Deleting a node from a DLL decreases the size of its contents by one. 5) After an add, the next of the new head is the old head. 6) After a delete, the new tail is the prev of the old tail.

3. Consider the Active Records data model shown in Figure 2 in the paper titled "Unbounded Data Model Verification Using SMT Solvers." Write two Alloy Models for this specification using the two approaches described in Sections 4 and 4.1 of the paper titled "Bounded Verification of Ruby on Rails Data Models."

Specify and check or simulate the following properties on this data model using the Alloy Analyzer: 1) A user's photos are the same as the photos in that user's profile. 2) When a user is deleted, all the photos and videos that belong to that user are also deleted. 3) Each

profile is associated with a user. 4) One user can only have one role. 5) A photo and a video can have the same tag. 6) Two users can have the same profile.

4. Consider the following specification: Computer science graduate program is made up of three types of people: professors, graduate students with an advisor and graduate students without an advisor.

Each graduate student can have at most one professor as an advisor. A professor can have zero or more advisees. Each professor can have research grants which are used to fund his/her advisees. A professor can advise students only if he/she has at least one grant. An advisee of a professor must be funded by a grant of that professor.

Professors with research grants offer research seminars to graduate students when they are looking for advisees. If a graduate student has a professor as an advisor, he/she must have taken a research seminar from that professor.

Write an Alloy model for the above specification. Write predicates for the following actions: *hire* where a student becomes the advisee of a professor, *offer* where a professor offers a new seminar, *take* where a student takes a seminar.

Write 3 properties about this specification in Alloy (at least one of them should fail and at least one should be correct), explain them in English and check them on your model.