Homework 1

Please put your solutions into the mailbox of the instructor at Warren Weaver Hall until the deadline, or hand them in before class on Mondays. Solutions to programming exercises should be submitted via email to the instructor with CC to ly603@nyu.edu.
The deadline for Homework 1 is February 11.

Problem 1 Alloy Structures, Expressions, and Constraints (2 + 6 + 2 Points)

Consider the Alloy structure I that is given by the following relations:

$$\texttt{father} = \{(M_2, M_0), (W_2, M_0), (M_3, M_1), (M_4, M_2), (W_4, M_2), (W_5, M_3), (M_5, M_3)\}$$
$$\texttt{mother} = \{(M_0, W_0), (M_4, W_1), (W_4, W_1), (W_5, W_2), (M_5, W_3)\}$$

(a) Draw the snapshot graph for structure I.

(b) In structure I, compute the relations `parent`, `sibling`, and `ancestor` that are defined as follows:

(i) `parent = mother + father`

(ii) `sibling = parent.~parent - iden`

(iii) `ancestor = ^parent`

(c) Which of the following Alloy constraints holds in structure I?

(i) `all x : univ | some y : univ | y in x.father`

(ii) `no p : univ | p in p.^(mother + father)`

Problem 2 Refactoring Navigation Expressions (6 Points)

When writing "navigation expressions" in Alloy, you may notice repeated subexpressions that can be factored out, making the overall expression more succinct. For example, the expression `p.mother.brother + p.father.brother`, denoting p's uncle, can be written instead as `p.(mother + father).brother`. Simplifications like this rely on assumptions that certain algebraic identities hold, such as

(a) `s.(p + q) = s.p + s.q`

(b) `s.(p - q) = s.p - s.q`

(c) `s.(p & q) = s.p & s.q`

(d) `~(p + q) = ~p + ~q`

(e) `p.~q = q.p`

(f) `s.~p = p.s`

for a given set (unary relation) `s` and binary relations `p` and `q`.

For each putative identity, say whether it holds, and if not, give a counterexample. Here is an example of how you might check the first identity using the Alloy Analyzer:

```
module identities
assert unionDistributes {
  all s: set univ, p, q: univ->univ | s.(p + q) = s.p + s.q
}
check unionDistributes for 4
```

Problem 3 Address Book Constraints and Expressions (9 Points)

Consider a model of a multilevel address book consisting of a set *Addr* of addresses, and a set *Name* consisting of two disjoint subsets *Alias* and *Group*. The mapping from names to addresses is represented by a relation *address*, but a name can map not only to an address, but also to a name.

Express the following facts about the address book model using Alloy formulas:

(a) All names eventually map to an address.

(b) There are no cycles; if you resolve a name repeatedly, you never reach the same name again.

Express the following *simulation constraints*, which you might add during exploration of a model in order to see more interesting instances:

(c) The address book has at least two levels.

(d) Some groups are non-empty.

(e) All aliases map to groups.

Finally, write expressions for each of the following, without using comprehension syntax:

(f) the set of names that are members of groups;

(g) the set of groups that are empty;

(h) the mapping from aliases to the addresses they refer to, directly or indirectly;

(i) the mapping from names to addresses which, when a name maps to some addresses directly, and some other addresses indirectly, includes only the direct addresses.

For example, the following expression denotes the set of all addresses that appear in the address book: `Name.address & Addr`