CITS5501 Software Testing and Quality Assurance
Semester 1, 2018
Workshop 10 – Specifications in Alloy

This workshop contains some exercises using the Alloy visualizer. It can be downloaded from http://alloytools.org/download.html.

Exercises

1. The full address-book specification (Jackson, 2006) used in lectures is as follows:

```
sig Name, Addr {}

sig Book {addr: Name -> lone Addr}

pred show (b: Book) {
   #b.addr > 1
   #Name.(b.addr) > 1
   }

run show for 3 but 1 Book

pred add (b, b': Book, n: Name, a: Addr) {b'.addr = b.addr + n -> a}
pred del (b, b': Book, n: Name) {b'.addr = b.addr - n -> Addr}
fun lookup (b: Book, n: Name): set Addr {n.(b.addr)}

pred showAdd (b, b': Book, n: Name, a: Addr) {
   add [b, b', n, a]
   #Name.(b'.addr) > 1
   }

run showAdd for 3 but 2 Book

assert delUndoesAdd {
   all b,b',b'': Book, n: Name, a: Addr |
     no n.(b.addr) and
     add [b,b',n,a] and del [b',b'',n] implies b.addr = b''.addr
   }

assert addIdempotent {
   all b,b',b'': Book, n: Name, a: Addr |
     add [b,b',n,a] and add [b',b'',n,a] implies b'.addr = b''.addr
   }
```

```
check delUndoesAdd for 10 but 3 Book
check addIdempotent for 3
```

The "add" predicate represents adding a name to the address book, and "del" represents deleting one.

Try running the various "show" predicates (e.g. "showAdd"), and checking the assertions. Can you work out what the assertions are doing?

2. How can we model the *natural numbers* (starting from zero) in Alloy?
   Suggestion: Specify that *zero* is a natural number; and specify a function that takes you from a number, to the next number larger than it.

   Using the Alloy visualizer, what do instances of your model look like? Are they what you expect?

3. How can we model the process of transferring money between two bank accounts? What entities and operations will be needed?

   (NB: Recall that `Int`s in Alloy are limited in width – so be careful about how you represent things like currency.)