# Model Checking, SS2011: Exercise Sheet 8

May 30, 2011

**Exercise 8.1.** We can represent natural numbers with *Peano arithmetic terms*. Let `0` denote the natural 0, and `s(N)` denote the successor of the natural denoted by `N`. Do the following tasks.

1. Create a Prolog procedure `ptoi`/2 that for each Peano term gives the corresponding natural.
2. Create a Prolog procedure `itop`/2 that for each natural gives the corresponding Peano term. *Hint: define* `itop(0, 0).` *before any other rule for* `itop`.
3. Define `itop(0, 0).` after any other rule for `itop` and follow the execution trace of `itop(2,P)`.
4. Create `itop2`/2, a procedure that prints an error message and terminates when given a negative number, and else gives for each natural the corresponding Peano term.

**Exercise 8.2.** In the interactive prompt of Prolog, encode a graph by executing the following commands.

```
assert(edge(1,2)).
assert(edge(2,2)).
assert(edge(2,3)).
assert(edge(3,4)).
assert(edge(3,5)).
```

Query Prolog as many times as you need to confirm that the graph is connected.

**Exercise 8.3.** Create a Prolog procedure `listnumber`/1 that for each element of a list prints its index and the element.

**Exercise 8.4.** Create a Prolog procedure that given a list of indices `Li` and an arbitrary list `L` enumerates all elements of `L` corresponding to some index in `Li`.

**Exercise 8.5** Create a Prolog procedure that given a procedure `P/1` and a list `L` enumerates all elements of `L` for which `P` succeeds.

**Exercise 8.6.** Consider the following encoding of a directed graph in Prolog.

```
start(1).
edge(1,2).
edge(2,2).
edge(2,3).
edge(3,4).
edge(3,5).
```

Create a Prolog procedure `creach`/0 that computes the set of reachable nodes $C$ and prints $C$ upon termination.

**Exercise 8.7.** Code the abstract reachability model checker presented in class. Enable SICStus Prolog trace mode and test it on various past exercises.

**Exercise 8.8.** Do the following modifications to the abstract reachability model checker `check` presented in Lecture 9.

1. Upon reaching an error state, print the potential counter example path.
2. Modify `abst_reach_step` so that new abstract states are printed when discovered.

**Exercise 8.9 (optional).** Code the Forward-Symbolic-Reachability algorithm in Prolog. Enable SICStus Prolog trace mode and test the algorithm on the programs found in exercises 2.6, 3.3, and 3.4.