Formal Specification of Abstract Datatypes Exercise 2 (May 5)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

April 3, 2014

The result for each exercise is to be submitted by the deadline stated above via the Moodle interface as a .zip or .tgz file which contains

- a PDF file with
  - a cover page with the title of the course, your name, Matrikelnummer, and email-address,
  - the content required by the exercise (specification, source, proof),
- (if required) the CafeOBJ (.mod) file(s) of the specifications.

Exercise 2: Loose Specifiction of Integers

Assume you are already given a strictly adequate specification of the classical algebra of Boolean values by a sort *bool* with free constructors $True :\to bool$, $False :\to bool$ and the logical operations $Not : bool \to bool$, $And : bool \times bool \to bool$, $Or : bool \times bool \to bool$.

Based on this specification, first write a strictly adequate loose specification of the classical algebra of natural numbers by a sort *nat* with free constructors $0_n :\to nat$ and $succ_n : nat \to nat$, constant $1_n :\to nat$, and operations $-_n : nat \to nat$ ($x -_n y = 0$ for $x \leq y$), $+_n : nat \times nat \to nat$, $*_n : nat \times nat \to nat$, $=_n$: $nat \times nat \to bool$, $<_n$: $nat \times nat \to bool$.

With the help of these sorts and operations, your next task is to write a loose specification (potentially with constructors) of a sort *int* with constants $0_i :\to int$, $1_i :\to int$ and operations $-_i : int \to int$, $+_i : int \times int \to int$, $-_i : int \times int \to int$, $*_i : int \times int \to int$, $=_i$: $int \times int \to bool$, $<_i$: $int \times int \to bool$. This specification shall be strictly adequate with respect to the classical algebra of integer numbers.

Actually, your task is two write *two* alternative versions of this specification:

1. The idea of the first version is that every integer can be represented as a "natural numbers with a sign". Introduce corresponding constructors $p : nat \to int$ and $n : nat \to int$ with informal interpretation $p(x) = +x$ and $n(x) = -(x+1)$ (why better not choose the interpretation $n(x) = -x$?).
2. The idea of the second version is that every integer can be represented as a difference of two natural numbers. Introduce a corresponding constructor $i : nat \times nat \to int$ with informal interpretation $i(x, y) = x - y$.

Based on either representation, you may write (potentially recursive) definitions of the required constants and operations.

Compare the versions and highlight their advantages and disadvantages. Which logic did you use for each specification? Which specification do you prefer? Actually, only one of the two versions is *strictly* adequate. Which one and why?

Implement an executable version of *one* of the specifications (justify your choice) in CafeOBJ (i.e. a tight module `module! MYINT {...}` based on the existing modules `BOOL` and `NAT`). The specification should be as close to the chosen loose specification as possible.

Test the executable specification with a couple of sample reductions. In particular, check whether

$$-_i(0_i) =_i 1_i +_i -_i(1_i)$$

reduces to *true* (deliver the corresponding output).