Solutions – Practical 2

Michael P. Fourman

February 2, 2010

```
structure Rational:NumberSig =
struct
type num = int * int

fun gcd(0,b) = b
  | gcd(a,b) = gcd(b mod a, a);

fun red(a,b) =
    let val res = gcd(a,b)
    in (a div res, b div res) end;

fun (an, ad) ++ (bn, bd) = red(an*bd + ad*bn, ad*bd);
fun (an, ad) -- (bn, bd) = red(an*bd - ad*bn, ad*bd);
fun (an, ad) ** (bn, bd) = red(an*bn,         ad*bd);
fun (an, ad) // (bn, bd) = red(an*bd,         ad*bn);

fun ~~(n, d) : num = (~n ,d)

fun ((an,ad):num) == ((bn,bd):num) = (an*bd = ad*bn);

end
```

The auxiliary functions, `gcd`, and `red` (reduce), are hidden by the signature, so we don't need to make them local. These auxiliary functions clearly deal with integers, so we don't need (but could have) type constriants on the later function that use `red`. Some type constraints on the last two functions are needed to disambiguate `*` and `~`.

```
structure Interval:NumberSig =
struct
type num = real*real;

local fun min(a, b) :real = if a < b then a else b
      and max(a, b) :real = if a < b then b else a

      and min4(a, b, c, d) :real = min(min(a, b), min(c, d))
      and max4(a, b, c, d) :real = max(max(a, b), max(c, d))
in
      fun (a,b) ++ (c,d) :num = (a+c, b+d)
      and (a,b) -- (c,d) :num = (a-d, b-c)
      and (a,b) ** (c,d) :num =
          let val ends = (a*c, a*d, b*c, b*d)
          in (min4 ends, max4 ends) end
      and (a,b) // (c,d):num =
          if c <= 0.0 andalso d >= 0.0 then error "div zero"
          else (a,b) ** (1.0/d, 1.0/c)

      fun ~~(a, b) :num = (~b, ~a)
end;

fun (a,b) == (c,d) = (a = b) andalso (b = c) andalso (c = d) ;

end
```

Notice the use of local functions, and the way in which a 4-tuple is passed to `min4` and `max4`.

The definition of equality given here is very strict. We consider each interval to represent an indeterminate number that lies *somewhere* in the interval. We can only know that two indeterminate numbers are equal under the conditions given here. However, you could also argue that we only know that they are different when the intervals are disjoint. Proper resolution of this dilemma would require non-boolean truth values.