

Mathematical logic – coursework 4, exercise 4

László Treszkai

March 26, 2017

4/4. Give an example of a structure that fulfills every axioms of the Robinson arithmetics, which contains two elements that are neither greater than or equal to, nor smaller than or equal to one another, or prove that such a structure doesn't exist.

Let the universe of the \mathcal{Q}' structure be $\omega \cup \{\kappa, \lambda\}$, where $\kappa, \lambda \notin \omega$ and $\kappa \neq \lambda$. \mathcal{Q}' fulfills the axioms of Robinson arithmetics, if we define the $+$ and \cdot operations as follows ($n \in \omega$):

$$\begin{array}{ll} \kappa \cdot 0 := 0 & \lambda \cdot 0 := 0 \\ n + \kappa := \kappa & n + \lambda := \lambda \\ \kappa + n := \kappa & \lambda + n := \lambda \\ \kappa + \kappa := \kappa & \lambda + \lambda := \lambda \end{array}$$

In case $n, m \in \omega$, let $n + m$ and $n \cdot m$ be defined as the result of addition and multiplication of the corresponding natural numbers. Furthermore, a few values can be chosen arbitrarily:

$$\begin{array}{l} \kappa + \lambda := \kappa \text{ or } \kappa + \lambda := \lambda \\ \lambda + \kappa := \kappa \text{ or } \lambda + \kappa := \lambda \\ n \cdot \kappa := \kappa \text{ or } n \cdot \kappa := \lambda \\ n \cdot \lambda := \kappa \text{ or } n \cdot \lambda := \lambda \\ 0 \cdot \kappa := 0 \text{ or } 0 \cdot \kappa := \kappa \text{ or } 0 \cdot \kappa := \lambda \\ 0 \cdot \lambda := 0 \text{ or } 0 \cdot \lambda := \kappa \text{ or } 0 \cdot \lambda := \lambda \end{array}$$

With the choice of $\kappa + \lambda := \lambda$ and $\lambda + \kappa := \kappa$, we get $\neg \exists z (z + \kappa = \lambda)$ and $\neg \exists z (z + \lambda = \kappa)$, so neither $\kappa \leq \lambda$, nor $\lambda \leq \kappa$. ■

We can easily prove that \mathcal{Q}' fulfills all of the required axioms for any two elements, using the above definitions of the operations and the properties of ω . For illustration, below is the proof of $x \cdot (y + 1) = (x \cdot y) + x$, for the case of $x = \kappa$, $y = n \in \omega$:

$$\begin{aligned}
n \in \omega &\Rightarrow n + 1 = m \in \omega \\
\kappa * (n + 1) &= \kappa * m = \kappa \\
(\kappa * n) + \kappa &= \kappa + \kappa = \kappa
\end{aligned}$$

The Haskell program listed in the appendix checks whether a finite substructure of \mathcal{Q}' fulfills every axiom, for a given definition of the operations.

Appendix for coursework 4, exercise 4

robinsonTest.hs:

```

1 import NonStdRobinson
2
3 main =
4     mapM_ ( \ (n, phi) →
5         print $ "Axiom " ++ show n ++
6             if phi testNumbers
7                 then " fulfilled"
8                 else " failed"
9     ) (zip [1..] axiomTestList)
10     where testNumbers = [Nat 0, Nat 1, Nat 2, Kappa, Lambda]
11           -- "Nulla, egy, ketto, sok." Urban J.
12
13 axiomTestList = [axiom1, axiom2, axiom3, axiom4, axiom5, axiom6, axiom7]
14
15 -- x + 1 != 0
16 axiom1 :: [CuteNumber] → Bool
17 axiom1 xs = all phi xs
18     where phi x = x.+Nat 1 /= Nat 0
19
20 -- (x != 0) → (exists y: x = y + 1)
21 axiom2 :: [CuteNumber] → Bool
22 axiom2 xs = all phi xs
23     where phi x = (x == Nat 0) || any (\y → x == y.+Nat 1) xs
24
25 -- (x != y) → (x + 1 != y + 1)
26 axiom3 :: [CuteNumber] → Bool
27 axiom3 xs = all phi (pairs xs)
28     where phi (x,y) = (x == y) || (x.+Nat 1 /= y.+Nat 1)
29
30 -- x + 0 = x
31 axiom4 :: [CuteNumber] → Bool
32 axiom4 xs = all phi xs
33     where phi x = x.+Nat 0 == x

```

```

34
35 --  $x * 0 = 0$ 
36 axiom5 :: [CuteNumber] → Bool
37 axiom5 xs = all phi xs
38     where phi x = x.*.Nat 0 == Nat 0
39
40 --  $x + (y + 1) = (x + y) + 1$ 
41 axiom6 :: [CuteNumber] → Bool
42 axiom6 xs = all phi (pairs xs)
43     where phi (x,y) = x.+. (y.+.Nat 1) == (x.+.y).+.Nat 1
44
45 --  $x * (y + 1) = (x * y) + x$ 
46 axiom7 :: [CuteNumber] → Bool
47 axiom7 xs = all phi (pairs xs)
48     where phi (x,y) = x.*. (y.+.Nat 1) == (x.*.y).+.x
49
50 pairs :: [a] → [(a,a)]
51 pairs xs = (,) <$> xs <*> xs

```

NonStdRobinson.hs:

```

1 module NonStdRobinson
2 (
3
4     CuteNumber(Nat, Kappa, Lambda),
5     (.+.), (.*)
6 ) where
7
8 data CuteNumber = Nat Int | Kappa | Lambda deriving (Eq, Ord, Show)
9
10 -- + + + + + + + + + + + + + + + + + + + + + +
11
12 (.+.) :: CuteNumber → CuteNumber → CuteNumber
13 Nat n1 .+. Nat n2 = Nat (n1 + n2)
14
15 Nat _ .+. Kappa = Kappa
16 Kappa .+. Nat _ = Kappa
17 Kappa .+. Kappa = Kappa
18 Kappa .+. Lambda = Lambda
19
20 Nat _ .+. Lambda = Lambda
21 Lambda .+. Nat _ = Lambda
22 Lambda .+. Lambda = Lambda
23 Lambda .+. Kappa = Kappa
24
25 -- * * * * * * * * * * * * * * * * * * * * * *
26
27 (.*.) :: CuteNumber → CuteNumber → CuteNumber
28 Nat n1 .*. Nat n2 = Nat (n1 * n2)
29
30 Nat 0 .*. _ = Nat 0

```

```

31  _      .*. Nat 0  = Nat 0
32
33  Nat _   .*. Kappa  = Kappa
34  Kappa   .*. _      = Kappa
35
36  Nat _   .*. Lambda = Lambda
37  Lambda  .*. _      = Lambda

```