Logic exercises 8

Hand in solutions for questions marked **(PMT)** to the SAO by Thursday 30th November 2017.

1. Let L, M be the signature and L-structure of slide 85 of the notes for lists of type `Nat`, the invented Haskell type for the natural numbers $\{0, 1, 2, \ldots\}$.

   (a) Write an L-formula $Const(xs)$ expressing that all elements of the list xs (if any) are equal.

   (b) Consider the property: 'xs contains at most two distinct numbers, possibly repeated'. Example: [3,7,7,3,7], [5,5,5]. Non-example: [3,4,7,3,4]. Write L-formulas expressing this, in two ways: (i) directly, (ii) using $Const$ and `merge`.

   (c) Write an L-formula expressing that xs has no duplicate entries (e.g., [1,4,2,3] — all four entries are different).

   (d) Write an L-formula expressing that (in standard Haskell terms) $xs = \texttt{filter} <\!n\ ys$.

   Remember to state the sorts of variables clearly. Do not write $[x]$, $[x, y]$, etc. in formulas.

2. Rewrite the following in 3-sorted first-order logic with sorts **lecturer, Sun, other** (assume every object is of one of these sorts) and a binary relation symbol $\texttt{bought}_{\mathbf{s},\mathbf{s}'}(\mathbf{s}, \mathbf{s}')$ for each pair $(\mathbf{s}, \mathbf{s}')$ of sorts. Some of them are *really sneaky:* remember that types of terms in atomic formulas must be correct. $x \neq y$ abbreviates $\neg(x = y)$. Example: $\forall x(\texttt{Sun}(x) \to \exists y(\texttt{lecturer}(y) \wedge \texttt{bought}(y, x)))$ rewrites to $\forall x : \texttt{Sun}\ \exists y : \texttt{lecturer}(\texttt{bought}_{\texttt{lecturer},\texttt{Sun}}(y, x))$.

   (a) $\forall x(\texttt{Sun}(x) \to \exists y\ \texttt{bought}(y, x))$

   (b) $\forall x(\neg\texttt{lecturer}(x) \to \exists y(\texttt{Sun}(y) \wedge \texttt{bought}(x, y)))$

   (c) $\forall x \forall y(\texttt{lecturer}(x) \wedge \texttt{Sun}(y) \to x \neq y)$

3. For each of the following function specifications, give a suitable pre-condition and write the post-condition in logic.

   (a)
   ```
   better:: Int -> Int ->  Bool
   --post: result is true iff argument1 is greater than argument2.
   ```

   (b)
   ```
   pred:: Int ->  Int
   --post: result is the number before the input and result is always >0.
   ```

   (c)
   ```
   issquare:: Int -> Bool
   --post: result is true iff the input is a perfect square
   ```

4. What should the following function compute (in simple words)?

   ```
   guess:: Int -> Int -> Int
   --pre: none
   --post: (guess x y) <= x & (guess x y) <=y & ((guess x y)=x or (guess x y)=y)
   ```

5. **(PMT)** Work out what the following are doing, and express it by a post-condition in logic:

   (a)
   ```
   guess1::[Nat] -> [Nat]
   --pre:  None
   guess1 []    = []
   guess1 x:xs  = (1+x):(guess1 xs)
   ```

   (b)
   ```
   guess2::Nat -> [Nat] -> [Nat]
   --pre: 2nd input is sorted
   guess2  x  []         = [x]
   guess2 x y:ys   | x<y  = x:(y:ys)
                   | x>=y = y:(guess2 x ys)
   ```

(c)
```
guess3::[Nat] -> Nat
--pre: input is not the empty list
guess3 [x]                   = x
guess3 (x:(y:ys))  | x <= y  = guess3(y:ys)
                   | x > y   = guess3(x:ys)
```

(d)
```
guess4::[Char] -> Bool
--pre: xs <> []
guess4 xs = guess5 xs []

guess5::[Char] -> [Char] -> Bool
--pre:  at least one input is not the empty list
guess5 [] ys                    = false
guess5 (x:xs) zs  | (x:xs)==zs  = true
                  | xs==zs      = true
                  | otherwise   = guess5 xs (x:zs)
```

6. Using the terminology of question 1, write pre- and post-conditions in logic for the Haskell functions informally specified below. You can use formulas etc. in the notes/Q1.

```
contains: [Nat] -> [Nat] -> Bool
-- post: contains xs ys is true when every entry in xs is an entry in ys

make_unique :: [Nat] -> [Nat]
-- post: make_unique xs is a list containing exactly one occurrence of
-- each element that occurs in xs (not necessarily in the same order).

perm :: [Nat] -> [Nat] -> Bool
-- post: perm xs ys means that ys is a reordering of xs
--  eg perm [1,2,3,1] [3,1,1,2] and perm [1,2] [1,2] are true
--     perm [1,2,3,1] [3,1,2] and perm [1,2] [1,1] are false
```

7. **(PMT for JMC only)** (CS1 students cover the necessary material for this question in course C145 Mathematical Methods. JMC1 should cover most or all of it in mathematics.)

Let L be the three-sorted signature with sorts `Nat`, `Real`, `Seq`, binary function symbols $+^n$: $\texttt{Nat} \times \texttt{Nat} \to \texttt{Nat}$, $+^r, -, \times : \texttt{Real} \times \texttt{Real} \to \texttt{Real}$, binary relation symbols $<^r(\texttt{Real},\texttt{Real})$, $\leq^r(\texttt{Real},\texttt{Real})$, $<^n(\texttt{Nat},\texttt{Nat})$ and $\leq^n(\texttt{Nat},\texttt{Nat})$, and an 'evaluation' function symbol !! : $\texttt{Seq} \times \texttt{Nat} \to \texttt{Real}$. There are also constants $\underline{1}^n : \texttt{Nat}$ and $\underline{0}^r, \underline{1}^r : \texttt{Real}$. We let $n, m, \ldots$ be variables of sort `Nat`, $x, y, \ldots$ be variables of sort `Real`, and $s, t, \ldots$ be variables of sort `Seq`.

Let the L-structure M be as follows: the objects (in its domain) of sort `Nat` are the natural numbers $0, 1, 2, \ldots$, the objects of sort `Real` are the real numbers, and the objects of sort `Seq` are all infinite sequences of real numbers. If $s = (s_0, s_1, \ldots)$ is a sequence of real numbers (an object of sort `Seq`), then $s!!n = s_n$ for each n of sort `Nat`. All other symbols are interpreted in the usual way.

We can now say *a lot* about sequences. For example, $\forall n \forall m(n <^n m \to s!!n <^r s!!m)$ expresses that the sequence s is strictly increasing. In a similar way, write formulas expressing:

(a) x is equal to zero (in `Real`— it says above that x is of sort `Real`).
(b) y is strictly positive
(c) the absolute value $|x|$ of x is strictly less than y
(d) x is the limit of sequence s
(e) the sequence s converges
(f) challenge: the sequence s satisfies D'Alembert's 'ratio test' condition for $\sum_{n=0}^{\infty} s_n$ to converge: to wit, $\lim_{n\to\infty} |s_{n+1}/s_n| < 1$.
(g) x is $\sup s$

Logic exercises 8 (unassessed) solutions

For discussion in pmt in week 10

1. Below, variables x, y, z, k, m, n are of sort `Nat`, and $xs, ys, \ldots$ are of sort [`Nat`].
   (a) $Const(xs)$ can be $\forall m \forall n(m < \sharp(xs) \wedge n < \sharp(xs) \to xs!!m = xs!!n)$, or $\exists n \forall m(m < \sharp(xs) \to xs!!m = n)$.
   (b) xs has ≤ 2 distinct entries: (i) $\exists m \exists n \forall k(k < \sharp(xs) \to xs!!k = m \vee xs!!k = n)$.
   (ii) $\exists ys \exists zs(Const(ys) \wedge Const(zs) \wedge \mathtt{merge}(ys, zs, xs))$, where $Const$ is as in (1a).
   (c) xs has no duplicate entries: $\forall m \forall n(m < n \wedge n < \sharp(xs) \to xs!!m \neq xs!!n)$, or $\forall y \forall z \forall ys \forall zs \forall vs(xs = ys{+}{+}y{:}(zs{+}{+}(z{:}vs)) \to y \neq z)$.
   (d) $xs = \mathtt{filter}\ {<}n\ ys$:
   $\exists zs(\mathtt{merge}(xs, zs, ys) \wedge \forall m(\underbrace{(m < \sharp(xs) \to xs!!m < n)}_{\text{all items in } xs \text{ are } < n} \wedge \underbrace{(m < \sharp(zs) \to n \leq zs!!m)}_{\text{all other items are } \geq n}))$.

2. (a) If some 'y' bought x, then y could be of sort **lecturer, Sun**, or **other**. We must allow for all possibilities:
   $$\begin{aligned}\forall x : \textbf{Sun}\ \big(&\exists y : \textbf{lecturer}(\mathtt{bought}_{\textbf{lecturer},\textbf{Sun}}(y, x))\\ &\vee \exists z : \textbf{Sun}(\mathtt{bought}_{\textbf{Sun},\textbf{Sun}}(z, x))\\ &\vee \exists v : \textbf{other}(\mathtt{bought}_{\textbf{other},\textbf{Sun}}(v, x))\big)\end{aligned}$$
   (b) To say that all non-**lecturer**s bought a Sun is to say that all **Sun**s and all **other**s bought a Sun. So $\forall x(\neg\mathtt{lecturer}(x) \to \exists y(\mathtt{Sun}(y) \wedge \mathtt{bought}(x, y)))$ translates to:
   $\forall x : \textbf{Sun}\ \exists y : \textbf{Sun}(\mathtt{bought}_{\textbf{Sun},\textbf{Sun}}(x, y)) \wedge \forall v : \textbf{other}\ \exists y : \textbf{Sun}(\mathtt{bought}_{\textbf{other},\textbf{Sun}}(v, y))$.
   (c) $\forall x \forall y(\textbf{lecturer}(x) \wedge \textbf{Sun}(y) \to x \neq y)$ translates to $\top$, since no two objects of different sorts can be equal. We *could not* write $\forall x : \textbf{lecturer}\ \forall y : \textbf{Sun}(x \neq y)$, because the variables x, y have different sorts so $x \neq y$ is not well-formed.

3. (a)
```
better:: Int -> Int ->  Bool
--pre: none
--post: better x y <--> x>y
```
   (b)
```
pred:: Int ->  Int
--pre: input > 1
--post: pred x = x-1 & pred x > 0
```
   (c)
```
issquare:: Int -> Bool
--pre: none
--post: issquare x <--> (E)y(y*y=x)
```

4. guess x y = minimum of x,y

5. **(PMT)** Below, variables x, y, n are of sort `Nat`, and $xs, ys, \ldots$ are of sort `[Nat]`.
   (a) `Guess1 xs` adds 1 to every entry in xs. So post:
   $\sharp(xs) = \sharp(ys) \wedge \forall n(n < \sharp(xs) \to ys!!n = \underline{1} + xs!!n)$, where $ys = Guess1\ xs$.
   (b) `guess2 x xs` inserts an x into xs in the right place. So post-condition is:
   $\exists zs \exists vs(xs = zs{+}{+}vs \wedge ys = zs{+}{+}(x{:}vs) \wedge sorted(ys))$ where $ys = \mathtt{guess2}\ x\ xs$.
   We did 'sorted' in lectures: $sorted(ys)$ is $\forall n \forall m(n \leq m \wedge m < \sharp(ys) \to ys!!n \leq ys!!m)$.
   (c) `guess3 xs` returns the maximum entry in xs. So post (letting $y = \mathtt{guess3}\ xs$) is:
   $\exists n(n < \sharp(xs) \wedge xs!!n = y) \wedge \forall n(\underline{0} \leq n \wedge n < \sharp(xs) \to xs!!n \leq y)$.
   (d) Given the pre-condition, `guess4 xs` is true if xs is a palindrome (is self-reverse — e.g., [l,e,v,e,l]), and false if not (e.g, [a,b]). So post-condition could be
   $\forall n \forall m((n + m) + \underline{1} = \sharp(xs) \to xs!!n = xs!!m))$.

6. Below, variables m, n are of sort `Nat`, and xs, ys of sort [`Nat`] or (in last part) [`Char`].

- contains $xs\ ys \leftrightarrow \forall n(n < \sharp(xs) \to \exists m(m < \sharp(ys) \wedge xs!!n = ys!!m))$. You can't use merge directly, as the order of entries in xs may differ from in ys.
- contains $xs\ ys \wedge$ contains $ys\ xs \wedge$ no-dups(xs), where $ys =$ make-unique xs and no-dups is the formula in Q1c. Or, using 'count' from lectures, we can say it by expressing "$\forall n(\mathit{count}(n, ys) = \min(\mathit{count}(n, xs), \underline{1}))$" (this is the idea, not a proper formula). It can be done by: $\forall n \forall k(k \leq \mathit{count}(n, xs) \wedge k \leq \underline{1} \leftrightarrow k \leq \mathit{count}(n, ys))$.
- Using 'count' from lectures: $\forall n(\mathit{count}(n, xs) = \mathit{count}(n, ys))$.

7. **(PMT for JMC only)** Reminder: the only symbols in the signature are binary function symbols $+^n$: $\mathtt{Nat} \times \mathtt{Nat} \to \mathtt{Nat}$, $+^r, -, \times$: $\mathtt{Real} \times \mathtt{Real} \to \mathtt{Real}$, binary relation symbols $<^r, \leq^r$: $\mathtt{Real} \times \mathtt{Real}$, and $<^n, \leq^n$: $\mathtt{Nat} \times \mathtt{Nat}$, and an 'evaluation' function symbol $!!$: $\mathtt{Seq} \times \mathtt{Nat} \to \mathtt{Real}$. There are also constants $\underline{1}^n$: $\mathtt{Nat}$ and $\underline{0}^r, \underline{1}^r$: $\mathtt{Real}$, which are not essential to have, but are a biiig help. Also, $n, m, \ldots$ are variables of sort $\mathtt{Nat}$, $x, y, \ldots$ are variables of sort $\mathtt{Real}$, and $s, t, \ldots$ are variables of sort $\mathtt{Seq}$.

   (a) x is equal to zero (in $\mathtt{Real}$): $x = \underline{0}^r$.

   (b) y is strictly positive: $\underline{0}^r <^r y$.

   (c) the absolute value $|x|$ of x is strictly less than y: this says that (i) $x < y$ and (ii) $-x < y$ (equivalently, $x + y > 0$). So we can use $x <^r y \wedge \underline{0}^r <^r x +^r y$. Below, I abbreviate this formula as "$\underline{|x| < y}$".

   (d) x is the limit of sequence s: $\forall y(\underline{0}^r <^r y \to \exists n \forall m(n <^n m \to \underline{|s!!m - x| <^r y}))$.

   (e) the sequence s converges: $\exists x$(the previous formula).

   (f) the sequence s satisfies D'Alembert's 'ratio test' condition for $\sum_{n=0}^{\infty} s_n$ to converge: to wit, $\lim_{n\to\infty} |s_{n+1}/s_n| < 1$.
   First, we need to express $|s_{n+1}/s_n| = z$ in our language, and we have no division operator. Well, $|s_{n+1}/s_n| = z$ just when $z \geq 0$ and $z \cdot s_n = \pm s_{n+1}$ — that is,

   $$(\underline{0} \leq^r z) \wedge \big(z \times s!!n = s!!(n +^n \underline{1}^n) \vee (z \times s!!n) + s!!(n +^n \underline{1}^n) = \underline{0}^r\big).$$

   Below, I write this formula as "A". So A is true just when $|s_{n+1}/s_n| = z$. Now we just need to say

   $$\exists x(x <^r \underline{1}^r \wedge \forall y(\underline{0}^r <^r y \to \exists n \forall m[n <^n m \to \exists z(A \wedge \underline{|z - x| <^r y})]).$$

   (g) x is $\sup s$: $\forall n(s!!n \leq^r x) \wedge \forall y(\forall n(s!!n \leq^r y) \to x \leq^r y)$. The first conjunct says that x is an upper bound of s. The second conjunct says that every upper bound of s is at least as big as x, so that x is the least upper bound of s.
   Or, shorter: $\forall y(\forall n(s!!n \leq^r y) \leftrightarrow x \leq^r y)$.