CS 15-312: Assignment 4 - *Written Part*

Sample Solution

March 29, 2000

1 Encoding with Parametric Polymorphic Functions [15 points]

1.1 Natural numbers

N ::= `!t.t→(t→t)→t`

Solution 1. We observe that:
`(fun f:t→t in f`^{mn}` end) succ`
`= (fun g:t→t in g`^{m}` end) ((fun f:t→t in f`^{n}` end) succ)`
Therefore, `mult` can be defined as follows:

```
mult =
  fun m:N in
    fun n:N in
      Fun t in
        fun zero:t in
          fun succ:t→t in
            (fun g:t→t in fun h:t in m[t] h g end end)
              ((fun f:t→t in fun i:t in n[t] i f end end) succ)
              zero
          end end end end end
  =
  fun m:N in
    fun n:N in
      Fun t in
        fun zero:t in
          fun succ:t→t in
            m[t] zero (fun i:t in n[t] i succ end)
          end end end end end
```

Solution 2. First solution using `mult`:

```
exp1 =
  fun m:N in
    fun n:N in
      n[N] 1 (mult m)
  =
  fun m:N in
    fun n:N in
      n[N]
        (Fun t in fun zero:t in fun succ:t→t in succ(zero) end end end)
        (mult m)
      end end
```

Second solution : we observe that:

(fun f:t→t in $\mathtt{f}^{m^n}$ end) succ
= (fun f:t→t in $\mathtt{f}^{m}$ end)^n succ
= (fun g:(t→t)→(t→t) in $\mathtt{g}^{n}$ end) (fun f:t→t in $\mathtt{f}^{m}$ end) succ

Therefore,

```
exp2 =
  fun m:N in
    fun n:N in
      Fun t in
        fun zero:t in
          fun succ:t→t in
            (fun g:(t→t)→(t→t) in fun h:t→t in n[t→t] h g end end)
              (fun f:t→t in fun i:t in m[t] i f end end)
              succ zero
          end end end end end
  =
  fun m:N in
    fun n:N in
      Fun t in
        fun zero:t in
          fun succ:t→t in
            (n[t→t] succ (fun f:t→t in fun i:t in m[t] i f end end)) zero
          end end end end end
```

Solution 3. The idea is that given $\underline{n}$, we compute ($\underline{i}$, `pred` $\underline{i}$) for $i = 0, \cdots, n$ in this order. Therefore we can define `pred` as follows:

```
pred =
  fun n:N in
    Fun t in
      fun zero:t in
        fun succ:t→t in
          (n[t*t] (zero, zero) (fun p:t*t in (succ p.1, p.1)).2
        end end end
```

1.2 Lists of natural numbers

```
L ::= !t.t→(N→t→t)→t
```

Solution 4.

```
length =
  fun l:L in
    l[N] 0
      (fun p:N in fun q:N in add 1 q end end)
    end
  =
  fun l:L in
    l[[N] (Fun t in fun zero:t in fun succ:t→t in zero end end end)
      (fun p:N in fun q:N in add
        (Fun t in fun zero:t in fun succ:t→t in succ(zero) end end end) q end end)
```

Solution 5.

```
sumlist =
  fun l:L in
    l[N] 0 add
    end
  =
  fun l:L in
    l[[N] (Fun t in fun zero:t in fun succ:t→t in zero end end end) add
    end
```

1.3 Boolean values

```
B ::= !t.t→t→t
```

Solution 5.

```
and =
  fun a:B in
    fun b:B in
      Fun t in
        fun p:t in
          fun q:t in
            a[t] (b[t] p q) q
          end end end end end
```

2 ADT Equivalence [35 points]

We will consider the case `'a = int` only. Generalization is straightforward. The corresponding signature `SET` and structure `SetRef` are then given as follows:

```
signature SET =
    sig
        type set

        val empty : set
        val insert : int * set -> set
        val member : set -> bool
    end
end

structure SetRef :> SET =
    struct
        type set = int list

        val empty  =  nil
        fun insert (x, s)  =  x :: s
        fun member (x, nil)  =  false
          | member (x, s::r)  =  x = s orelse member (x, r)
    end
end
```

Solution 1 [5 points].

```
structure SetCand :> SET =
    struct
        datatype set = EMPTY | NODE of int * set * set

        val empty  =  EMPTY
        fun insert (x, EMPTY)  =  NODE(x, EMPTY, EMPTY)
          | insert (x, NODE(h, l, r))  =
              if x = h then NODE(h, l, r)
              else if x < h then NODE(p, insert(x, l), r)
              else NODE(p, l, insert(x, r))
        fun member (x, EMPTY)  =  false
          | member (x, NODE(h, l, r))  =
              if x = h then true
              else if x < h then member(x, l)
              else member(x, r)
    end
end
```

Solution 2. We define the simulation relation $\sim$ as follows:

$S1 \sim S2 \;\Leftrightarrow\; \forall x$:`int` , `SetRef.member(x,` $S1$`) = SetCand.member(x,` $S2$`)`

Solution 3. The proof of the equivalence between `SetRef` and `SetCand` considers three cases, each of which corresponds to one of `empty`, `insert`, and `member`.

Case 1. *SetRef.empty* $\sim$ *SetCand.empty*

Proof. $\quad$ `SetRef.empty` $\sim$ `SetCand.empty`

$\Leftrightarrow \forall x$`:int` , `SetRef.member(`$x$`, SetRef.empty) = SetCand.member(`$x$`, SetCand.empty)`
$\Leftrightarrow \forall x$`:int` , `SetRef.member(`$x$`, nil) = SetCand.member(`$x$`, SetCand.EMPTY)`
$\Leftrightarrow \forall x$`:int` , `false = false`
$\Leftrightarrow$ `true`

$\square$

Case 2. $\quad S1 \sim S2 \;\Rightarrow\; \forall x$*`:int`* , *`SetRef.member(x,`* $S1$*`) = SetCand.member(x,`* $S2$*`)`*

Proof. Follows from the definition of $\sim$. $\square$

Case 3. $\quad S1 \sim S2 \;\Rightarrow\; \forall y$*`:int`* , *`SetRef.insert(y,`* $S1$*`)`* $\sim$ *`SetCand.insert(y,`* $S2$*`)`*

Proof. Suppose that $S1 \sim S2$, that is,

$\forall x$`:int` , `SetRef.member(`x`,` $S1$`) = SetCand.member(`x`,` $S2$`)`

We need to show that for any y`:int`,

`SetRef.insert(`y`,` $S1$`)` $\sim$ `SetCand.member(`y`,` $S2$`)`
$\Leftrightarrow \forall x$`:int` , `SetRef.member(`$x$`, SetRef.insert(`$y$`,` $S1$`))`
`= SetCand.member(`x`, SetCand.insert(`y`,` $S2$`))`

Given y, consider the set S_y:

S_y `=` $\{y\} \cup \{z \mid$ `SetRef.member(`z`,` $S1$`) = true`$\}$
`=` $\{y\} \cup \{z \mid$ `SetCand.member(`z`,` $S2$`) = true`$\}$ $\quad$ (*from* $S1 \sim S2$)

Thus it suffices to show that:

1. $\forall\ x \in S_y$, `SetRef.member(`x`, SetRef.insert(`y`,` $S1$`)) = true`
2. $\forall\ x \notin S_y$, `SetRef.member(`x`, SetRef.insert(`y`,` $S1$`)) = false`
3. $\forall\ x \in S_y$, `SetCand.member(`x`, SetCand.insert(`y`,` $S2$`)) = true`
4. $\forall\ x \notin S_y$, `SetCand.member(`x`, SetCand.insert(`y`,` $S2$`)) = false`

Let us first prove the first two cases. We assume `SetRef`. Suppose that $x \in S_y$:

$(x$ `=` $y) \vee$ `member(`x`,` $S1$`) = true`

Then,

`member(`x`, insert(`y`,` $S1$`))`
`= (`x `=` y`) orelse member(`x`,` $S1$`)`
`= true`

Similarly we can show that the second case also holds.

Now let us prove the remaining two cases. We first prove three lemmas. We assume `SetCand`.

Lemma 1. $\quad \forall\ x$*`:int`* , *`member(`*x*`, NODE(h, l, r)) = (`*x *`= h)`* $\vee$ *`member(`*x*`, l)`* $\vee$ *`member(`*x*`, r)`*

Proof. By case analysis: (1) x `= h`; (2) x `< h`; (3) x `> h`.[1] $\square$

[1] To be precise, we need to show that if `member(`x`, l) = true`, x `< h`, and that if `member(`x`, r) = true`, x `> h`.

Lemma 2. $\forall$ *S:set* , $\forall$ *x:int* , $\forall$ *y:int* ,
member(x, S) = false $\Rightarrow$ *member(x, insert(y, S)) = (x = y)*

Proof. By structural induction on S.

- S = EMPTY

```
  member(x, insert(y, S))
= member(x, NODE(y, EMPTY, EMPTY))
= (x = y) ∨ false ∨ false        (by lemma 1)
= (x = y)
```

- S = NODE(h, l, r)

  Assume that `member(x, S) = false`. From lemma 1, we have:

  `(x = h) = false`, `member(x, l) = false`, `member(x, r) = false`

  If y = h,

```
  member(x, insert(y, S))
= member(x, S)
= false
= (x = h)
= (x = y)
```

  If y < h,

```
  member(x, insert(y, S))
= member(x, NODE(h, insert(y, l), r))
= (x = h) ∨ member(x, insert(y, l)) ∨ member(x, r)
= false ∨ member(x, insert(y, l)) ∨ false
= member(x, insert(y, l))
= (x = y)        (by induction hypothesis on l)
```

  y > h can be checked similarly.

$\square$

Lemma 3. $\forall$ *S:set* , $\forall$ *x:int* , $\forall$ *y:int* ,
member(x, S) = true $\Rightarrow$ *member(x, insert(y, S)) = true*

Proof. By structural induction on S.

- S = EMPTY

  `member(x, S) = false`

- S = NODE(h, l, r)

  Assume that `member(x, S) = true`. From lemma 1, we have:

  `x = h ∨ member(x, l) = true ∨ member(x, r) = true`

If y = h,

$$
\begin{aligned}
& \texttt{member}(x,\ \texttt{insert}(y,\ S)) \\
=\ & \texttt{member}(x,\ S) \\
=\ & \texttt{true} \qquad\qquad (\textit{from the assumption})
\end{aligned}
$$

If y < h,

$$
\begin{aligned}
& \texttt{member}(x,\ \texttt{insert}(y,\ S)) \\
=\ & \texttt{member}(x,\ \texttt{NODE(h, insert}(y,\ \texttt{l), r)} \\
=\ & (x\ \texttt{= h}) \vee \texttt{member}(x,\ \texttt{insert}(y,\ \texttt{l))} \vee \texttt{member}(x,\ \texttt{r})
\end{aligned}
$$

From the assumption, we only have to consider three cases: (1) x = h; (2) member(x, l) = true; (3) member(x, r) = true.

In (1),

$$
\begin{aligned}
& \texttt{member}(x,\ \texttt{insert}(y,\ S)) \\
=\ & \texttt{true} \vee \texttt{member}(x,\ \texttt{insert}(y,\ \texttt{l))} \vee \texttt{member}(x,\ \texttt{r}) \\
=\ & \texttt{true}
\end{aligned}
$$

In (2),

$$
\begin{aligned}
& \texttt{member}(x,\ \texttt{insert}(y,\ S)) \\
=\ & (x\ \texttt{= h}) \vee \texttt{member}(x,\ \texttt{insert}(y,\ \texttt{l))} \vee \texttt{member}(x,\ \texttt{r}) \\
=\ & (x\ \texttt{= h}) \vee \texttt{true} \vee \texttt{member}(x,\ \texttt{r}) \qquad (\textit{by induction hypothesis}) \\
=\ & \texttt{true}
\end{aligned}
$$

In (3),

$$
\begin{aligned}
& \texttt{member}(x,\ \texttt{insert}(y,\ S)) \\
=\ & (x\ \texttt{= h}) \vee \texttt{member}(x,\ \texttt{insert}(y,\ \texttt{l))} \vee \texttt{true} \\
=\ & \texttt{true}
\end{aligned}
$$

$\square$

With the above three lemmas, now we are ready to prove that:

$$
\begin{aligned}
\forall\ x\ \in\ S_y\ ,\ \ & \texttt{member}(x,\ \texttt{insert}(y,\ S2)) = \texttt{true} \\
\forall\ x\ \notin\ S_y\ ,\ \ & \texttt{member}(x,\ \texttt{insert}(y,\ S2)) = \texttt{false}
\end{aligned}
$$

Recall that

$$S_y = \{y\}\ \cup\ \{z\ |\ \texttt{member}(z,\ S2) = \texttt{true}\}$$

We use a structural induction on $S2$.

- $S2$ = EMPTY

  Since member(x, $S2$) = false, it follows from lemma 2:

  $$\texttt{member}(x,\ \texttt{insert}(y,\ S2)) = (x\ \texttt{=}\ y)$$

  If $x \in S_y$, (x = y) = true. Therefore,

$$\texttt{member}(x,\ \texttt{insert}(y,\ S2)) = (x = y) = \texttt{true}$$

If $x \notin S_y$, $(x = y) = \texttt{false}$. Therefore,

$$\texttt{member}(x,\ \texttt{insert}(y,\ S2)) = (x = y) = \texttt{false}$$

- $S2 = \texttt{NODE(h, l, r)}$

If $y = \texttt{h}$,

$$\texttt{member}(x,\ \texttt{insert}(y,\ S2)) = \texttt{member}(x,\ S2)$$

$$\begin{aligned} S_y &= \{h\} \cup \{z \mid \texttt{member}(z,\ l) = \texttt{true}\} \cup \{z \mid \texttt{member}(z,\ r) = \texttt{true}\} \\ &= \{z \mid \texttt{member}(z,\ S2) = \texttt{true}\} \end{aligned}$$

Therefore,

$$\texttt{member}(x,\ \texttt{insert}(y,\ S2)) = \texttt{true} \Leftrightarrow x \in S_y$$

If $y < \texttt{h}$,

$$\begin{aligned} &\texttt{member}(x,\ \texttt{insert}(y,\ S2)) \\ &= \texttt{member}(x,\ \texttt{NODE(h, insert}(y,\ \texttt{l), r)}) \\ &= (x = \texttt{h}) \vee \texttt{member}(x,\ \texttt{insert}(y,\ \texttt{l})) \vee \texttt{member}(x,\ \texttt{r}) \end{aligned}$$

$$\begin{aligned} S_y = \{y\} \cup \{h\} &\cup \{z \mid \texttt{member}(z,\ l) = \texttt{true}\} \\ &\cup \{z \mid \texttt{member}(z,\ r) = \texttt{true}\} \end{aligned}$$

Consider $x \in S_y$.
(1) If $x = y$,

$$\begin{aligned} &\texttt{member}(x,\ \texttt{insert}(y,\ S2)) \\ &= (x = \texttt{h}) \vee \texttt{member}(x,\ \texttt{insert}(y,\ \texttt{l})) \vee \texttt{member}(x,\ \texttt{r}) \\ &= (x = \texttt{h}) \vee \texttt{true} \vee \texttt{member}(x,\ \texttt{r}) \qquad (\textit{by lemmas } 2,\ 3) \\ &= \texttt{true} \end{aligned}$$

(2) If $x = h$,

$$\begin{aligned} &\texttt{member}(x,\ \texttt{insert}(y,\ S2)) \\ &= (x = \texttt{h}) \vee \texttt{member}(x,\ \texttt{insert}(y,\ \texttt{l})) \vee \texttt{member}(x,\ \texttt{r}) \\ &= \texttt{true} \vee \texttt{member}(x,\ \texttt{insert}(y,\ \texttt{l})) \vee \texttt{member}(x,\ \texttt{r}) \\ &= \texttt{true} \end{aligned}$$

(3) If $\texttt{member}(x,\ \texttt{l}) = \texttt{true}$,

$$\begin{aligned} &\texttt{member}(x,\ \texttt{insert}(y,\ S2)) \\ &= (x = \texttt{h}) \vee \texttt{member}(x,\ \texttt{insert}(y,\ \texttt{l})) \vee \texttt{member}(x,\ \texttt{r}) \\ &= (x = \texttt{h}) \vee \texttt{true} \vee \texttt{member}(x,\ \texttt{r}) \qquad (\textit{by lemma } 3) \\ &= \texttt{true} \end{aligned}$$

(4) If $\texttt{member}(x,\ \texttt{r}) = \texttt{true}$,

$$
\begin{aligned}
&\texttt{member}(x\texttt{, insert}(y\texttt{, }S2\texttt{))} \\
&= (x \texttt{ = h}) \lor \texttt{member}(x\texttt{, insert}(y\texttt{, l))} \lor \texttt{member}(x\texttt{, r)} \\
&= (x \texttt{ = h}) \lor \texttt{member}(x\texttt{, insert}(y\texttt{, l))} \lor \texttt{true} \\
&= \texttt{true}
\end{aligned}
$$

Consider $x \notin S_y$:

$$(x \texttt{ = } y) \texttt{ = false},\ (x \texttt{ = h) = false},\ \texttt{member}(x\texttt{, l) = false},\ \texttt{member(x, r) = false}$$

Then,

$$
\begin{aligned}
&\texttt{member}(x\texttt{, insert}(y\texttt{, }S2\texttt{))} \\
&= (x \texttt{ = h}) \lor \texttt{member}(x\texttt{, insert}(y\texttt{, l))} \lor \texttt{member}(x\texttt{, r)} \\
&= \texttt{false} \lor (x \texttt{ = } y) \lor \texttt{false} \qquad \textit{(by lemma 2)} \\
&= (x \texttt{ = } y) \\
&= \texttt{false}
\end{aligned}
$$

y `>` `h` can be checked similarly.

$\square$