Sample solutions for the examination of Computability (DAT415/DIT310/DIT311/DIT312/TDA184) from 2020-01-15

Nils Anders Danielsson

1. (a) $A = \mathbb{N}$, $B = \mathbb{N}$.

   (b) The set is not countable: Let us assume that the set is countable. This means that there is an injection from $List\ (\mathbb{N} \to \mathbb{N})$ to $\mathbb{N}$. There is also an injection from $\mathbb{N} \to \mathbb{N}$ to $List\ (\mathbb{N} \to \mathbb{N})$: the function mapping f to $[f]$ (this function is injective because if $[f] = [g]$, then $f = g$). Thus, because compositions of injections are injective, we get that there is an injection from $\mathbb{N} \to \mathbb{N}$ to $\mathbb{N}$. However, this is impossible, because $\mathbb{N} \to \mathbb{N}$ is not countable. Thus we have arrived at a contradiction.

2. $\mathsf{True}(\lambda x.\mathsf{True}(x))$.

3. Yes. If f and g are both χ-computable, then there are closed χ expressions $\underline{f}$ and $\underline{g}$ witnessing the computability of f and g, respectively. There is also a witness $\underline{add}$ of the computability of the function $add \in \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined by $add\ (x, y) = x + y$. For any variable x the closed expression

   $$\lambda x.\ \underline{add}\ \mathsf{Pair}(\underline{f}\ x, \underline{g}\ x)$$

   (written using a mixture of concrete syntax and meta-level notation) witnesses the computability of h, because for any $n \in \mathbb{N}$ we have

   $$\begin{aligned}
   &[\![(\lambda x.\ \underline{add}\ \mathsf{Pair}(\underline{f}\ x, \underline{g}\ x))\ \ulcorner n \urcorner]\!] &&= \\
   &[\![\underline{add}\ \mathsf{Pair}(\underline{f}\ \ulcorner n \urcorner, \underline{g}\ \ulcorner n \urcorner)]\!] &&= \\
   &[\![\underline{add}\ \mathsf{Pair}(\ulcorner f\ n \urcorner, \ulcorner g\ n \urcorner)]\!] &&= \\
   &[\![\underline{add}\ \ulcorner (f\ n, g\ n) \urcorner]\!] &&= \\
   &\ulcorner f\ n + g\ n \urcorner &&= \\
   &\ulcorner h\ n \urcorner.
   \end{aligned}$$

4. No. We can prove this by reducing the halting problem (which is not χ-decidable) to f.

   If f is χ-decidable, then there is a closed χ expression $\underline{f}$ witnessing the computability of f. We can use this expression to construct a closed χ

expression $\underline{halts}$ (written using a mixture of concrete syntax and meta-level notation):

$$\underline{halts} = \lambda\, e.\, \underline{f}\, \mathsf{Pair}(\ulcorner \lambda\_.\, (\lambda\_.\, \mathsf{True}())\ \llcorner e \lrcorner \urcorner, \ulcorner \lambda\_.\, \mathsf{True}() \urcorner).$$

Let us now verify that $\underline{halts}$ witnesses the decidability of the halting problem. For any $e \in \mathit{CExp}$ we have

$$\begin{aligned}
&[\![\underline{halts}\ \ulcorner e \urcorner]\!] &&= \\
&[\![\underline{f}\, \mathsf{Pair}(\ulcorner \lambda_.\, (\lambda_.\, \mathsf{True}())\ e \urcorner, \ulcorner \lambda_.\, \mathsf{True}() \urcorner)]\!] &&= \\
&[\![\underline{f}\ \ulcorner ((\lambda_.\, (\lambda_.\, \mathsf{True}())\ e), (\lambda_.\, \mathsf{True}())) \urcorner]\!] &&= \\
&\ulcorner f\, ((\lambda_.\, (\lambda_.\, \mathsf{True}())\ e), (\lambda_.\, \mathsf{True}())) \urcorner &&= \\
&\mathbf{if}\ \exists\, b \in \mathit{Bool}.\, [\![(\lambda_.\, (\lambda_.\, \mathsf{True}())\ e)\ \ulcorner b \urcorner]\!] = [\![(\lambda_.\, \mathsf{True}())\ \ulcorner b \urcorner]\!] && \\
&\quad \mathbf{then}\ \ulcorner \mathsf{true} \urcorner\ \mathbf{else}\ \ulcorner \mathsf{false} \urcorner &&= \\
&\mathbf{if}\ \exists\, b \in \mathit{Bool}.\, [\![(\lambda_.\, \mathsf{True}())\ e]\!] = \mathsf{True}()\ \mathbf{then}\ \ulcorner \mathsf{true} \urcorner\ \mathbf{else}\ \ulcorner \mathsf{false} \urcorner &&= \\
&\mathbf{if}\ [\![(\lambda_.\, \mathsf{True}())\ e]\!] = \mathsf{True}()\ \mathbf{then}\ \ulcorner \mathsf{true} \urcorner\ \mathbf{else}\ \ulcorner \mathsf{false} \urcorner.
\end{aligned}$$

If $[\![e]\!]$ is defined, then

$$\begin{aligned}
&\mathbf{if}\ [\![(\lambda_.\, \mathsf{True}())\ e]\!] = \mathsf{True}()\ \mathbf{then}\ \ulcorner \mathsf{true} \urcorner\ \mathbf{else}\ \ulcorner \mathsf{false} \urcorner &&= \\
&\mathbf{if}\ \mathsf{True}() = \mathsf{True}()\ \mathbf{then}\ \ulcorner \mathsf{true} \urcorner\ \mathbf{else}\ \ulcorner \mathsf{false} \urcorner &&= \\
&\ulcorner \mathsf{true} \urcorner,
\end{aligned}$$

and if $[\![e]\!]$ is undefined, then

$$\begin{aligned}
&\mathbf{if}\ [\![(\lambda_.\, \mathsf{True}())\ e]\!] = \mathsf{True}()\ \mathbf{then}\ \ulcorner \mathsf{true} \urcorner\ \mathbf{else}\ \ulcorner \mathsf{false} \urcorner &&= \\
&\mathbf{if}\ \mathsf{True}()\ \text{is undefined}\ \mathbf{then}\ \ulcorner \mathsf{true} \urcorner\ \mathbf{else}\ \ulcorner \mathsf{false} \urcorner &&= \\
&\ulcorner \mathsf{false} \urcorner.
\end{aligned}$$

Thus we get

$$[\![\underline{halts}\ \ulcorner e \urcorner]\!] = \ulcorner\ \mathbf{if}\ [\![e]\!]\ \text{is defined}\ \mathbf{then}\ \mathsf{true}\ \mathbf{else}\ \mathsf{false} \urcorner,$$

i.e. $\underline{halts}$ witnesses the decidability of the halting problem.

5. (a) If the machine is run with 1110 as the input string, then the following configurations are encountered:

   - $(s_0, [\,], [1, 1, 1, 0])$.
   - $(s_1, [1], [1, 1, 0])$.
   - $(s_1, [1, 1], [1, 0])$.
   - $(s_1, [1, 1, 1], [0])$.
   - $(s_2, [1, 1], [1, \sqcup])$.
   - $(s_3, [1], [1, 0, \sqcup])$.

   The last configuration above is a halting one, so the resulting string is 110.

(b) Yes, the machine implements a function that subtracts one from the input if the input is positive, and leaves the input unchanged if it is zero:

- If the input is $\ulcorner 0 \urcorner = 0$, then the machine halts right away.
- If the input is $\ulcorner 1 + n \urcorner = 1^{1+n}0$ for some $n \in \mathbb{N}$, then the machine will move to the right past all the ones, replace the zero with a blank, move left, replace the last one (there has to be at least one) with a zero, and halt (after potentially moving left). Thus the result is $1^n 0 = \ulcorner n \urcorner$.

6. No, the semantics is not total. Take the program $\mathsf{min}\ (\mathsf{proj}\ 1) \in RF_1^-$ and the vector $\mathsf{nil}, 1 \in \mathbb{N}^1$. There is no $m \in \mathbb{N}$ such that

$$\mathsf{min}\ (\mathsf{proj}\ 1)\, [\mathsf{nil}, 1] \Downarrow m,$$

because if there were, then $\mathsf{proj}\ 1\, [\mathsf{nil}, 1, n] \Downarrow 0$ would hold for some $n \in \mathbb{N}$, and it does not (because $\mathit{index}\ (\mathsf{nil}, 1, n)\ 1 = 1 \neq 0$).