Sample solutions for the examination of Computability (DAT415/DIT311/DIT312/TDA184) from 2021-01-13

Nils Anders Danielsson

1. (a) $A = \mathbb{N} \to \mathbb{N}$, $B = \{\, 0 \,\}$.

   (b) First consider the following lemma:

   Lemma. *If there is a surjection from B to A, then there is an injection from $A \to C$ to $B \to C$.*

   Proof. Take a surjection $f \in B \to A$. Define the function $g \in (A \to C) \to (B \to C)$ by $g\, h\, x = h\, (f\, x)$. This function is injective: Take h_1, $h_2 \in A \to C$. If $g\, h_1 = g\, h_2$, then, for every $x \in B$, we have $h_1\, (f\, x) = g\, h_1\, x = g\, h_2\, x = h_2\, (f\, x)$. Because f is surjective this means that we have $h_1\, y = h_2\, y$ for every $y \in A$, i.e. $h_1 = h_2$. □

   Note that there is a surjection from $\mathbb{N} \to \mathbb{N}$ to $\mathbb{N}$ (map f to $f\, 0$), so by the lemma above there is an injection from $\mathbb{N} \to \{\, 0, 1 \,\}$ to $(\mathbb{N} \to \mathbb{N}) \to \{\, 0, 1 \,\}$.

   Let us now prove that $(\mathbb{N} \to \mathbb{N}) \to \{\, 0, 1 \,\}$ is not countable. For this purpose, let us assume that the set is countable, i.e. that there is an injection from $(\mathbb{N} \to \mathbb{N}) \to \{\, 0, 1 \,\}$ to $\mathbb{N}$. The composition of two injections is injective, so this implies that there is an injection from $\mathbb{N} \to \{\, 0, 1 \,\}$ to $\mathbb{N}$, i.e. that $\mathbb{N} \to \{\, 0, 1 \,\}$ is countable. However, a minor variant of the diagonalisation argument that was used in a lecture to show that $\mathbb{N} \to \mathbb{N}$ is uncountable can be used to show that $\mathbb{N} \to \{\, 0, 1 \,\}$ is uncountable. Thus we have arrived at a contradiction, so $(\mathbb{N} \to \mathbb{N}) \to \{\, 0, 1 \,\}$ is not countable.

2. **case** $\mathsf{True}()$ **of** $\{\mathsf{True}(x) \to x\}$.

3. 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):[1]

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

For any $e \in \mathit{CExp}$ we have

$$\begin{array}{ll}
[\![\underline{halts}\ \ulcorner e \urcorner]\!] & = \\
[\![\underline{f}\ \mathsf{Pair}(\ulcorner \lambda\, _.\, \mathsf{False}() \urcorner, \ulcorner e \urcorner)]\!] & = \\
[\![\underline{f}\ \ulcorner ((\lambda\, _.\, \mathsf{False}()), e) \urcorner]\!] & = \\
\ulcorner f\, ((\lambda\, _.\, \mathsf{False}()), e) \urcorner & = \\
\ulcorner\ \textbf{if}\ [\![(\lambda\, _.\, \mathsf{False}())\ e]\!] = \ulcorner \mathsf{false} \urcorner\ \textbf{then}\ \mathsf{true}\ \textbf{else}\ \mathsf{false} \urcorner & = \\
\ulcorner\ \textbf{if}\ [\![e]\!]\ \text{is defined}\ \textbf{then}\ \mathsf{true}\ \textbf{else}\ \mathsf{false} \urcorner, &
\end{array}$$

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

4. Yes. The closed expression

$$\begin{array}{l}
\underline{f} = \lambda\, p.\, \textbf{case}\ p\ \textbf{of} \\
\qquad\qquad \{\, \mathsf{Pair}(e_1, e_2) \rightarrow \\
\qquad\qquad\quad \textbf{case}\ \mathit{equal}\ \mathsf{Pair}(\mathit{eval}\ \mathsf{Apply}(e_1, e_2), \ulcorner\ulcorner \mathsf{false} \urcorner\urcorner)\ \textbf{of} \\
\qquad\qquad\qquad \{\, \mathsf{True}() \rightarrow \mathsf{True}()\, \} \\
\qquad\qquad \}
\end{array}$$

(written using a mixture of concrete syntax and meta-level notation) witnesses the computability of f. Here eval is a self-interpreter and equal an equality test that satisfy the following properties:

$$\begin{array}{l}
\forall\ e \in \mathit{CExp}.\ [\![\mathit{eval}\ \ulcorner e \urcorner]\!] = \ulcorner [\![e]\!] \urcorner \\
\forall\ e_1, e_2 \in \mathit{CExp}. \\
\quad [\![\mathit{equal}\ \mathsf{Pair}(\ulcorner e_1 \urcorner, \ulcorner e_2 \urcorner)]\!] = \ulcorner\ \textbf{if}\ e_1 = e_2\ \textbf{then}\ \mathsf{true}\ \textbf{else}\ \mathsf{false} \urcorner
\end{array}$$

Let us prove that $\underline{f}$ is an implementation of f. Take two closed expressions $e_1, e_2 \in \mathit{CExp}$. We get that

$$\begin{array}{ll}
[\![\, \underline{f}\ \ulcorner (e_1, e_2) \urcorner\,]\!] & = \\
[\![\, \underline{f}\ \mathsf{Pair}(\ulcorner e_1 \urcorner, \ulcorner e_2 \urcorner)\,]\!] & = \\
[\![\, \textbf{case}\ \mathit{equal}\ \mathsf{Pair}(\mathit{eval}\ \mathsf{Apply}(\ulcorner e_1 \urcorner, \ulcorner e_2 \urcorner), \ulcorner\ulcorner \mathsf{false} \urcorner\urcorner)\ \textbf{of} & \\
\qquad \{\, \mathsf{True}() \rightarrow \mathsf{True}()\, \}\,]\!] & = \\
[\![\, \textbf{case}\ \mathit{equal}\ \mathsf{Pair}([\![\mathit{eval}\ \ulcorner \mathsf{apply}\ e_1\ e_2 \urcorner]\!], \ulcorner\ulcorner \mathsf{false} \urcorner\urcorner)\ \textbf{of} & \\
\qquad \{\, \mathsf{True}() \rightarrow \mathsf{True}()\, \}\,]\!] & = \\
[\![\, \textbf{case}\ \mathit{equal}\ \mathsf{Pair}(\ulcorner [\![\mathsf{apply}\ e_1\ e_2]\!] \urcorner, \ulcorner\ulcorner \mathsf{false} \urcorner\urcorner)\ \textbf{of} & \\
\qquad \{\, \mathsf{True}() \rightarrow \mathsf{True}()\, \}\,]\!]\, . &
\end{array}$$

We can conclude the proof by considering the following three, exhaustive cases:

[1] In the first version of these sample solutions I had written Apply instead of Pair. When I corrected the exams I encountered a solution that used Pair, and realised my mistake. Thanks!

- If $[\![\mathsf{apply}\ e_1\ e_2]\!]$ is equal to $\ulcorner \mathsf{false} \urcorner$, then we have

$$\begin{array}{ll} [\![\underline{f} \ulcorner (e_1, e_2) \urcorner]\!] & = \\ [\![\textbf{case}\ \mathit{equal}\ \mathsf{Pair}(\ulcorner \ulcorner \mathsf{false} \urcorner \urcorner, \ulcorner \ulcorner \mathsf{false} \urcorner \urcorner)\ \textbf{of} & \\ \quad \{ \mathsf{True}() \to \mathsf{True}() \}]\!] & = \\ [\![\textbf{case}\ \mathsf{True}()\ \textbf{of}\ \{ \mathsf{True}() \to \mathsf{True}() \}]\!] & = \\ \ulcorner \mathsf{true} \urcorner & = \\ \ulcorner f(e_1, e_2) \urcorner . & \end{array}$$

- If $[\![\mathsf{apply}\ e_1\ e_2]\!]$ is defined, but not equal to $\ulcorner \mathsf{false} \urcorner$, then

$$\begin{array}{ll} [\![\underline{f} \ulcorner (e_1, e_2) \urcorner]\!] & = \\ [\![\textbf{case}\ \mathit{equal}\ \mathsf{Pair}(\ulcorner [\![\mathsf{apply}\ e_1\ e_2]\!] \urcorner, \ulcorner \ulcorner \mathsf{false} \urcorner \urcorner)\ \textbf{of} & \\ \quad \{ \mathsf{True}() \to \mathsf{True}() \}]\!] & = \\ [\![\textbf{case}\ \mathsf{False}()\ \textbf{of}\ \{ \mathsf{True}() \to \mathsf{True}() \}]\!] , & \end{array}$$

  which is undefined, and thus equal to $\ulcorner f(e_1, e_2) \urcorner$.
- If $[\![\mathsf{apply}\ e_1\ e_2]\!]$ is undefined, then $[\![\underline{f} \ulcorner (e_1, e_2) \urcorner]\!]$ is also undefined, and thus equal to $\ulcorner f(e_1, e_2) \urcorner$.

5. (a) If the machine is run with 111 as the input string, then the following configurations are encountered:
   - $(s_0, [\,], [1, 1, 1])$.
   - $(s_0, [1], [1, 1])$.
   - $(s_0, [1, 1], [1])$.
   - $(s_0, [1, 1, 1], [\sqcup])$.
   - $(s_0, [\sqcup, 1, 1, 1], [\sqcup])$.
   - $(s_0, [\sqcup, \sqcup, 1, 1, 1], [\sqcup])$.
   - ...

   The machine stays in state s_0 forever: after the first couple of steps it will always read a blank. It does not halt.

   (b) No. If the machine is run with $0 = \ulcorner 0 \urcorner$ as the input string, then the following configurations are encountered:
   - $(s_0, [\,], [0])$.
   - $(s_1, [\,], [\sqcup])$.
   - $(s_1, [\,], [\sqcup])$.
   - ...

   The same configuration is encountered twice, so the machine is stuck in a loop and does not halt.

6. No. If we remove suc, proj or rec, then we can still construct the term $\mathsf{comp\ zero\ nil} \in PRF_1$, and the unary function represented by this term is not increasing:

$$\begin{array}{ll} [\![\mathsf{comp\ zero\ nil}]\!]\ (\mathsf{nil}, 1) & = \\ [\![\mathsf{zero}]\!]\ \mathsf{nil} & = \\ 0 & \ngeq \\ 1 & \end{array}$$

If we instead remove comp, then we can construct $\mathsf{rec\ zero}\ (\mathsf{proj}\ 1) \in PRF_1$, and

$$\begin{array}{ll} [\![\mathsf{rec\ zero}\ (\mathsf{proj}\ 1)]\!]\ (\mathsf{nil}, 1) & = \\ [\![\mathsf{proj}\ 1]\!]\ (\mathsf{nil}, 0, [\![\mathsf{rec\ zero}\ (\mathsf{proj}\ 1)]\!]\ (\mathsf{nil}, 0)) & = \\ 0 & \ngeq \\ 1. & \end{array}$$

Finally, if we remove zero, then we can construct the term

$$\mathsf{comp}\ (\mathsf{rec}\ (\mathsf{proj}\ 0)\ (\mathsf{proj}\ 1))\ (\mathsf{nil}, \mathsf{proj}\ 0, \mathsf{proj}\ 0) \in PRF_1,$$

and

$$\begin{array}{ll} [\![\mathsf{comp}\ (\mathsf{rec}\ (\mathsf{proj}\ 0)\ (\mathsf{proj}\ 1))\ (\mathsf{nil}, \mathsf{proj}\ 0, \mathsf{proj}\ 0)]\!]\ (\mathsf{nil}, 1) & = \\ [\![\mathsf{rec}\ (\mathsf{proj}\ 0)\ (\mathsf{proj}\ 1)]\!]\ (\mathsf{nil}, 1, 1) & = \\ [\![\mathsf{proj}\ 1]\!]\ (\mathsf{nil}, 1, 0, [\![\mathsf{rec}\ (\mathsf{proj}\ 0)\ (\mathsf{proj}\ 1)]\!]\ (\mathsf{nil}, 1, 0)) & = \\ 0 & \ngeq \\ 1. & \end{array}$$