Sample solutions for the examination of Models of Computation (DIT310/TDA183/TDA184) from 2018-01-10

Nils Anders Danielsson

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

   (b) It is not countable. This follows because *List* $\{0,1\} \to$ *List* $\{0,1\}$ is in bijective correspondence with $\mathbb{N} \to \mathbb{N}$, and $\mathbb{N} \to \mathbb{N}$ is not countable.

   We established in the lectures that *List* A is countable for countable sets A, and $\{0,1\}$ is countable. Thus there is an injection from *List* $\{0,1\}$ to $\mathbb{N}$. There is also an injection *bits* from $\mathbb{N}$ to *List* $\{0,1\}$ that, for a natural number n, returns the binary number (without redundant zeros) that represents n, seen as a list of bits (with the most significant bit first).

   The Schröder–Bernstein theorem states that if there are injections from A to B and from B to A, then there is a bijection from A to B. Thus we have established that *List* $\{0,1\}$ is in bijective correspondence with $\mathbb{N}$, from which it follows that *List* $\{0,1\} \to$ *List* $\{0,1\}$ is in bijective correspondence with $\mathbb{N} \to \mathbb{N}$.

   However, the Schröder–Bernstein theorem was not covered in the lectures, so I provide an alternative proof of uncountability, based on the following two lemmas:

   Lemma 1. *For any sets A, B and C, if $f \in B \to C$ is injective, then there is an injection from $A \to B$ to $A \to C$.*

   Proof. Let us define the function $g \in (A \to B) \to (A \to C)$ by $g\ h\ x = f\ (h\ x)$. This function is injective: Take any $h_1, h_2 \in A \to B$ for which $g\ h_1 = g\ h_2$. For any $x \in A$ we have $g\ h_1\ x = g\ h_2\ x$, i.e. $f\ (h_1\ x) = f\ (h_2\ x)$. Because f is injective this implies that, for any $x \in A$, $h_1\ x = h_2\ x$. Thus $h_1 = h_2$. □

   Lemma 2. *For any sets A, B and C, if $f \in A \to B$ is surjective, then there is an injection from $B \to C$ to $A \to C$.*

   Proof. Let us define the function $g \in (B \to C) \to (A \to C)$ by $g\ h\ x = h\ (f\ x)$. This function is injective: Take any $h_1,\ h_2 \in B \to C$ for

which $g\ h_1 = g\ h_2$. For any $x \in A$ we have $g\ h_1\ x = g\ h_2\ x$, which implies that $h_1\ (f\ x) = h_2\ (f\ x)$. Because f is surjective we get that, for any $y \in A$, $h_1\ y = h_2\ y$. Thus $h_1 = h_2$. □

As discussed above there is an injection *bits* from $\mathbb{N}$ to *List* $\{0,1\}$. Lemma 1 thus gives us an injection f from $\mathbb{N} \to \mathbb{N}$ to $\mathbb{N} \to$ *List* $\{0,1\}$. We can also construct a surjection from *List* $\{0,1\}$ to $\mathbb{N}$, for instance the function that gives the sum of all elements in the list. Lemma 2 thus gives us an injection g from $\mathbb{N} \to$ *List* $\{0,1\}$ to *List* $\{0,1\} \to$ *List* $\{0,1\}$. By composing g and f we get an injection from $\mathbb{N} \to \mathbb{N}$ to *List* $\{0,1\} \to$ *List* $\{0,1\}$. Now, if *List* $\{0,1\} \to$ *List* $\{0,1\}$ were countable, then $\mathbb{N} \to \mathbb{N}$ would also be countable (because compositions of injections are injective). Thus *List* $\{0,1\} \to$ *List* $\{0,1\}$ is not countable.

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

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

   If *is-total* is χ-decidable, then there is a closed χ expression $\underline{\textit{is-total}}$ witnessing the computability of *is-total*. We can use this expression to construct a closed χ expression $\underline{\textit{halts}}$ (written using a mixture of concrete syntax and meta-level notation):

   $$\underline{\textit{halts}} = \lambda\, e.\, \underline{\textit{is-total}}\ \ulcorner\, \lambda\_.\, (\lambda\_.\, \ulcorner\, 0\, \urcorner)\ \llcorner\, e\, \lrcorner\, \urcorner.$$

   This expression witnesses the decidability of the halting problem: for any closed expression $e \in \mathit{Exp}$ we have

   $$\begin{array}{ll}
   [\![\underline{\textit{halts}}\ \ulcorner\, e\, \urcorner]\!] & = \\
   [\![\underline{\textit{is-total}}\ \ulcorner\, \lambda_.\, (\lambda_.\, \ulcorner\, 0\, \urcorner)\ e\, \urcorner]\!] & = \\
   \ulcorner\, \textit{is-total}\ (\lambda_.\, (\lambda_.\, \ulcorner\, 0\, \urcorner)\ e)\, \urcorner & = \\
   \textbf{if}\ \forall\, m \in \mathbb{N}.\ \exists\, n \in \mathbb{N}.\, [\![(\lambda_.\, (\lambda_.\, \ulcorner\, 0\, \urcorner)\ e)\ \ulcorner\, m\, \urcorner]\!] = \ulcorner\, n\, \urcorner & \\
   \quad \textbf{then}\ \ulcorner\, \mathsf{true}\, \urcorner\ \textbf{else}\ \ulcorner\, \mathsf{false}\, \urcorner & = \\
   \textbf{if}\ \exists\, n \in \mathbb{N}.\, [\![(\lambda_.\, \ulcorner\, 0\, \urcorner)\ e]\!] = \ulcorner\, n\, \urcorner\ \textbf{then}\ \ulcorner\, \mathsf{true}\, \urcorner\ \textbf{else}\ \ulcorner\, \mathsf{false}\, \urcorner & = \\
   \textbf{if}\ [\![(\lambda_.\, \ulcorner\, 0\, \urcorner)\ e]\!] = \ulcorner\, 0\, \urcorner\ \textbf{then}\ \ulcorner\, \mathsf{true}\, \urcorner\ \textbf{else}\ \ulcorner\, \mathsf{false}\, \urcorner & = \\
   \textbf{if}\ [\![e]\!]\ \text{is defined}\ \textbf{then}\ \ulcorner\, \mathsf{true}\, \urcorner\ \textbf{else}\ \ulcorner\, \mathsf{false}\, \urcorner. &
   \end{array}$$

4. Yes. The statement $\forall m \in \mathbb{N}.\ \exists\, n \in \mathbb{N}.\, [\![e\ \ulcorner\, m\, \urcorner]\!] = \ulcorner\, n\, \urcorner$ is vacuously true: e is assumed to be a witness of computability for $f \in \mathbb{N} \to \mathbb{N}$, so we have $\forall m \in \mathbb{N}.\, [\![e\ \ulcorner\, m\, \urcorner]\!] = \ulcorner\, f\ m\, \urcorner$ (and f is total, so $\ulcorner\, f\ m\, \urcorner$ is well-defined). Thus the χ program $\lambda\, e.\, \mathsf{True}()$ witnesses the χ-decidability of *is-total*.

5. (a) Denote the argument to the outermost occurrence of min,

   $$\mathsf{rec}\ (\mathsf{proj}\ 0)\ (\mathsf{comp}\ (\mathsf{min}\ \mathsf{suc})\ \mathsf{nil}),$$

   by q. We have

$$[\![q]\!]\,(\mathsf{nil}, 0, 0) = [\![\mathsf{proj}\ 0]\!]\,(\mathsf{nil}, 0) = 0.$$

Thus, by the semantics of min,

$$[\![p]\!]\,(\mathsf{nil}, 0) = [\![\mathsf{min}\ q]\!]\,(\mathsf{nil}, 0) = 0.$$

(b) No, this partial function is undefined for 1. If $f\ 1 = [\![p]\!]\,(\mathsf{nil}, 1)$ were defined, then $p\,[\mathsf{nil}, 1] \Downarrow m$ would be defined for some $m \in \mathbb{N}$. This would mean that $q\,[\mathsf{nil}, 1, n] \Downarrow 0$ would be defined for some $n \in \mathbb{N}$. However, we can prove that $q\,[\mathsf{nil}, 1, n] \Downarrow 0$ does not hold for any $n \in \mathbb{N}$. Consider the following two exhaustive cases:

- $n = 0$: In this case we have $q\,[\mathsf{nil}, 1, n] \Downarrow 1$, and because the semantics is deterministic we cannot also have $q\,[\mathsf{nil}, 1, n] \Downarrow 0$.
- $n = 1 + n'$ for some $n' \in \mathbb{N}$: If $q\,[\mathsf{nil}, 1, 1 + n'] \Downarrow 0$ holds, then $\mathsf{comp}\ (\mathsf{min}\ \mathsf{suc})\ \mathsf{nil}\,[\mathsf{nil}, 1, k, n'] \Downarrow 0$ holds for some $k \in \mathbb{N}$, and thus we have $\mathsf{min}\ \mathsf{suc}\,[\mathsf{nil}] \Downarrow 0$. This implies that $\mathsf{suc}\,[\mathsf{nil}, i] \Downarrow 0$ holds for some $i \in \mathbb{N}$. However, zero is not the successor of any natural number.

6. The (obviously Turing-computable) function $f \in \mathbb{N} \to \mathbb{N}$ that maps 1 to 0 and all other natural numbers to 1 cannot be implemented on this kind of machine.

   Let us assume that the machine *tm* implements this function, and let us consider what happens when *tm* is given the input $\ulcorner n \urcorner$ for some natural number $n \in \{1, 2\}$. Note that $\ulcorner 1 \urcorner = 10$ and $\ulcorner 2 \urcorner = 110$ start with the same symbol, so the machine's actions are identical for these inputs up to and including the first action (if any) that moves the head to the right. Note also that, by the definition of computability, the machine must terminate for these inputs. Thus, when the machine moves the head away from the left-most square on the tape (or if this never happens, when the machine terminates), the symbol present in this square must be the same for these two inputs. Let us denote this symbol by c. Note that when the machine terminates the symbol in the left-most square must still be c, because the head cannot be moved to the left, and consider the following two exhaustive cases:

   - $c = 0$. In this case we get the wrong result for $n = 2$, because $\ulcorner f\ 2 \urcorner = \ulcorner 1 \urcorner = 10$.
   - $c \neq 0$. In this case we get the wrong result for $n = 1$, because $\ulcorner f\ 1 \urcorner = \ulcorner 0 \urcorner = 0$.

   Thus the assumption that *tm* implements f is incorrect, and we can conclude that f cannot be implemented on this kind of machine.