Sample solutions for the examination of Models of Computation (DIT310/TDA183/TDA184) from 2017-01-10

Nils Anders Danielsson

1. (a) $\mathit{Bool} \rightarrow \mathit{Bool}$ is countable, $\mathbb{N} \rightarrow \mathbb{N}$ is not countable.

   (b) The given set—let us denote it by A—consists of χ-*computable* functions. For every such function there is at least one χ expression that witnesses its computability, and no expression is a witness for two different functions. The set of all χ expressions is countable, and thus A is also countable.

   In more detail: The set Exp consisting of (the abstract syntax of) every χ expression is countable, i.e. there is an injection $f \in \mathit{Exp} \rightarrow \mathbb{N}$. If we can construct an injection from A to Exp, then we are done, because the composition of two injections is also injective. We can construct a function $g \in A \rightarrow \mathit{Exp}$ in the following way: for a χ-computable function $h \in \mathbb{N} \rightarrow \mathbb{N}$, $g\ h$ is one of the χ expressions that witnesses the computability of h (the one for which the injective function f returns the smallest number). To see that the function g is injective, consider two functions h_1, $h_2 \in A$ for which $g\ h_1 = g\ h_2 = e$, where e witnesses the computability of both h_1 and h_2. We get that, for all $n \in \mathbb{N}$, $\ulcorner h_1\ n \urcorner = [\![e \ulcorner n \urcorner]\!] = \ulcorner h_2\ n \urcorner$. The function $\ulcorner \_ \urcorner$ is injective, so we get that, for all $n \in \mathbb{N}$, $h_1\ n = h_2\ n$, and thus h_1 and h_2 are equal.

2. $\mathsf{Lambda}(\mathsf{Zero}(), \mathsf{Apply}(\mathsf{Var}(\mathsf{Zero}()), \mathsf{Var}(\mathsf{Zero}())))$.

3. Yes. If f and g are both χ-computable, then there are closed χ expressions e_f and e_g witnessing the computability of f and g, respectively. For any variable x the closed expression $\mathsf{lambda}\ x\ (\mathsf{apply}\ e_g\ (\mathsf{apply}\ e_f\ (\mathsf{var}\ x)))$ witnesses the computability of $g \circ f$, because for any $n \in \mathbb{N}$ we have

$$\begin{array}{ll}
[\![\mathsf{apply}\ (\mathsf{lambda}\ x\ (\mathsf{apply}\ e_g\ (\mathsf{apply}\ e_f\ (\mathsf{var}\ x))))\ \ulcorner n \urcorner]\!] & = \\
[\![\mathsf{apply}\ e_g\ (\mathsf{apply}\ e_f\ \ulcorner n \urcorner)]\!] & = \\
[\![\mathsf{apply}\ e_g\ \ulcorner f\ n \urcorner]\!] & = \\
\ulcorner g\ (f\ n) \urcorner & = \\
\ulcorner (g \circ f)\ n \urcorner. &
\end{array}$$

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

   If f is χ-computable, 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 $\underline{\mathit{halts}}$:[1]

   $$\underline{\mathit{halts}} = \lambda\, p.\ \underline{f}\ \mathsf{Apply}(\mathsf{Lambda}(\mathsf{Zero}(), \ulcorner\ulcorner 35 \urcorner\urcorner), p)$$

   (For brevity $\underline{\mathit{halts}}$ is expressed using a mixture of concrete syntax and meta-level notation.) This expression witnesses the computability of the halting problem. Note that, for any closed expression $e \in \mathit{Exp}$,

   $$\begin{aligned}
   &[\![\underline{\mathit{halts}}\ \ulcorner e \urcorner]\!] &=\\
   &[\![\underline{f}\ \mathsf{Apply}(\mathsf{Lambda}(\mathsf{Zero}(), \ulcorner\ulcorner 35 \urcorner\urcorner), \ulcorner e \urcorner)]\!] &=\\
   &[\![\underline{f}\ \ulcorner (\lambda\, x.\ \ulcorner 35 \urcorner)\ e \urcorner]\!] &=\\
   &\ulcorner\ \mathbf{if}\ [\![(\lambda\, x.\ \ulcorner 35 \urcorner)\ e]\!] = \ulcorner 35 \urcorner\ \mathbf{then}\ \mathsf{true}\ \mathbf{else}\ \mathsf{false} \urcorner
   \end{aligned}$$

   (for some variable x). We have two cases to consider:

   - If e is a closed χ expression that terminates with a value, then $[\![(\lambda\, x.\ \ulcorner 35 \urcorner)\ e]\!] = \ulcorner 35 \urcorner$, and thus $[\![\underline{\mathit{halts}}\ \ulcorner e \urcorner]\!] = \ulcorner \mathsf{true} \urcorner$.
   - If e is a closed χ expression that does not terminate with a value, then $[\![(\lambda\, x.\ \ulcorner 35 \urcorner)\ e]\!] \neq \ulcorner 35 \urcorner$, and thus $[\![\underline{\mathit{halts}}\ \ulcorner e \urcorner]\!] = \ulcorner \mathsf{false} \urcorner$.

5. (a) If the machine is run with 000 as the input string, then the following configurations are encountered:

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

   The last configuration above is a halting one, with the head over the leftmost square, so the resulting string is $\square$.

   (b) No. If the machine is run with 00 as the input string, then the following configurations are initially encountered:

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

[1] In the first version of this document there was an error in the following expression (I used $\mathsf{Var}(p)$ instead of p at the end of the expression). I realised this while correcting exams. Thanks to all the students who got this right.

- $(s_1, [], [\square, \sqcup, \sqcup])$.
- $(s_4, [], [\sqcup, \sqcup, \sqcup])$.

  Every subsequent transition goes from the last configuration above to the same configuration, so the machine does not halt.

6. The operational semantics is extended with the following two inference rules (where CExp_χ is a set consisting of the abstract syntax of every closed expression from the unmodified language χ, and $\_ \Downarrow_\chi \_$ refers to the unmodified semantics):

$$\frac{e', v \in \mathit{CExp}_\chi \qquad e \Downarrow \ulcorner e' \urcorner \qquad e' \Downarrow_\chi v}{\mathsf{halts}\ e \Downarrow \ulcorner \mathsf{true} \urcorner}$$

$$\frac{e' \in \mathit{CExp}_\chi \qquad e \Downarrow \ulcorner e' \urcorner \qquad \forall v \in \mathit{CExp}_\chi.\ \neg\,(e' \Downarrow_\chi v)}{\mathsf{halts}\ e \Downarrow \ulcorner \mathsf{false} \urcorner}$$

   Properties:

   - The expression $\underline{\mathit{halts}}$ can be taken to be $\mathsf{lambda}\ e\ (\mathsf{halts}\ (\mathsf{var}\ e))$ (for some variable e). This expression is closed. Note that, for any closed χ expression e, $\ulcorner e \urcorner \Downarrow \ulcorner e \urcorner$, because $\ulcorner e \urcorner$ is a value. Thus

     $$[\![\mathsf{apply}\ \underline{\mathit{halts}}\ \ulcorner e \urcorner]\!] = [\![\mathsf{halts}\ \ulcorner e \urcorner]\!].$$

     Furthermore, if e is a closed χ expression that terminates with a value (according to $\_ \Downarrow_\chi \_$), then, by the first inference rule above,

     $$[\![\mathsf{halts}\ \ulcorner e \urcorner]\!] = \ulcorner \mathsf{true} \urcorner$$

     (again using the fact that $\ulcorner e \urcorner \Downarrow \ulcorner e \urcorner$.) Similarly, if e is a closed χ expression that does *not* terminate with a value (according to $\_ \Downarrow_\chi \_$), then, by the second inference rule above,

     $$[\![\mathsf{halts}\ \ulcorner e \urcorner]\!] = \ulcorner \mathsf{false} \urcorner.$$

   - Given that the semantics is still deterministic, the proof (sketch) given in the lectures showing that the halting problem for χ is not χ-decidable can be used almost unchanged to show that the halting problem for $\overline{\chi}$ is not $\overline{\chi}$-decidable.