Computability

Assignment 5 Kacper Korban

1. has-fixpoint is not χ-computable. Using Rice's theorem.

$$\text{has-fixpoint}\ (\lambda n.\ \text{Suc}(n)) = \text{false}$$
$$\text{has-fixpoint}\ (\lambda n.\ n) = \text{true}$$

$$\forall e \in \mathbb{N}.\ [\![e_1\ e]\!] = [\![e_2\ e]\!] \Rightarrow$$
$$\forall n \in \mathbb{N}.\ [\![e_1\ \ulcorner n \urcorner]\!] = [\![e_2\ \ulcorner n \urcorner]\!] \Rightarrow$$
$$\left(\exists n \in \mathbb{N}.\ [\![e_1\ \ulcorner n \urcorner]\!] = \ulcorner n \urcorner \iff \exists n \in \mathbb{N}\ [\![e_2\ \ulcorner n \urcorner]\!] = \ulcorner n \urcorner\right) \Rightarrow$$
$$\text{has-fixpoint}\ e_1 = \text{has-fixpoint}\ e_2$$

2. A function $f \in \mathbb{N} \to \mathbb{N}$ is Turing-computable when there exists a Turing machine tm, such that:

- $\Sigma_{tm} = \Sigma$ if $\ulcorner \urcorner$ encodes natural numbers in Σ
- $\forall n \in \mathbb{N}.\ [\![tm]\!]\ \ulcorner n \urcorner = \ulcorner f\ n \urcorner$

3. The given Turing machine implements incrementation function. That is a function that for a given number (n) returns $n+1$.

4.

Input alphabet: $\{0,1\}$

Tape alphabet: $\{0,1,\sqcup,C\}$

States: $\{acc, rej, cd, gr1, gl, gr0, cdr1, gll\}$

Initial state: cd

Accepting states: $\{acc\}$

Transition function:

$\delta(cd,1) = (gr1, C, R)$
$\delta(cd,0) = (gr0, 0, R)$
$\delta(gr0,C) = (gr0, C, R)$
$\delta(gr0,1) = (rej, 1, R)$
$\delta(gr0,0) = (acc, 0, R)$
$\delta(gr1,1) = (gr1, 1, R)$
$\delta(gr1,0) = (cdr1, 0, R)$

$\delta(cdr1,C) = (cdr1, C, R)$
$\delta(cdr1,0) = (rej, 0, R)$
$\delta(cdr1,1) = (gl, C, L)$
$\delta(gl,C) = (gl, C, L)$
$\delta(gl,0) = (gll, 0, L)$
$\delta(gll,1) = (gll, 1, L)$
$\delta(gll,C) = (cd, C, R)$

The Turing machine crosses a 1 on the left and then on the right. If it doesn't find a corresponding 1 on the right it rejects. If it finds a 0 on the left, it looks for a 0 on the right: if there is a 1 that is not crossed out yet, it rejects otherwise it accepts.

5. Abstract syntax:

S is a finite set $\quad s_0 \in S$

Σ is a finite set $\quad \sqcup \notin \Sigma$

Γ is a finite set $\quad \Sigma \cup \{\sqcup\} \subseteq \Gamma$

$$\frac{\delta \in S \times \Gamma \times \Gamma \to S \times (\Gamma \times \{L,R\}) \times (\Gamma \times \{L,R\})}{(S, s_0, \Sigma, \Gamma, \delta) \in TM_2}$$

Representation of a single tape is the same as the representation of the tape of a normal Turing machine, so:

$$Tape = List\,\Gamma \times List\,\Gamma$$

Small step operational semantics (assuming the implementation of $head_\Gamma$ and act from TM):

$$Configuration = State \times Tape \times Tape$$

$$\frac{\delta(s, head_\Gamma\, t_1, head_\Gamma\, t_2) = (s', a_1, a_2)}{(s, t_1, t_2) \to (s', act\, a_1\, t_1, act\, a_2\, t_2)}$$

Result (assuming implementation of list and remove for TM):

$$\frac{(s_0, [], s, [], ys) \to^* (s, t_1, t_2) \quad \nexists c.\ (s, t_1, t_2) \to c}{(xs, ys) \Downarrow (remove\,(list\,t_1), remove\,(list\,t_2))}$$

Result as partial function:

$$[\![\_]\!] \in \forall tm \in TM_2.\ List\,\Sigma_{tm} \times List\,\Sigma_{tm} \to List\,\Gamma_{tm} \times List\,\Gamma_{tm}$$

$$[\![tm]\!]\,(xs, ys) = (as, bs)\ if\ (xs, ys) \Downarrow (as, bs)$$

6. We define remove-stay:

remove-stay $\in$ TMS $\to$ TM

remove-stay $(S, s_0, \Sigma, \Gamma, \delta) = (\text{extendStates } S,\ s_0,\ \Sigma,\ \Gamma,\ \text{remove-S } \delta S \Gamma)$

extendStates $S = S \cup \{ m(s),\ s \in S \}$

(where m is an injection : $S \to A$ where $A \cap S = \emptyset$)

remove-S δ S Γ = f, such that $\forall s \in S, e \in \Gamma$.

if $\delta(s,e) = (s', e', S)$

then $f(s,e) = (m(s'), e', R)$ and

$\forall e'' \in \Gamma.\ f(m(s'), e'') = (s', e'', L)$

else $f(s,e) = \delta(s,e)$