Homework 4

Problem 1. Give a model for the sentence

$$\begin{aligned}\phi_{\text{lt}} = {} & \forall x\,[R_1(x,x)] \\ & \wedge \forall x,y\,[R_1(x,y) \leftrightarrow R_1(y,x)] \\ & \wedge \forall x,y,z\,[(R_1(x,y) \wedge R_1(y,z)) \to R_1(x,z)] \\ & \wedge \forall x,y\,[R_1(x,y) \to \neg R_2(x,y)] \\ & \wedge \forall x,y\,[\neg R_1(x,y) \to (R_2(x,y) \oplus R_2(y,x))] \\ & \wedge \forall x,y,z\,[(R_2(x,y) \wedge R_2(y,z)) \to R_2(x,z)] \\ & \wedge \forall x \exists y[R_2(x,y)].\end{aligned}$$

Solution. We describe the model $\mathcal{U}$ as follows:

1. The universe of $\mathcal{U}$ is an infinite set U, with a certain partial order R defined on it, and any of its subset has no maximum element.

2. There are two predicates: R_1 and R_2. We assign a binary relation $R_1^{\mathcal{U}}$ and a binary relation $R_2^{\mathcal{U}}$ to them respectively. Both $R_1^{\mathcal{U}}$ and $R_2^{\mathcal{U}}$ belong to $\mathcal{U}$, and are denoted and defined as follows: for all $x, y \in U$,

$$\begin{aligned} xR_1^{\mathcal{U}}y \iff x =_R y \iff xRy \wedge yRx, \\ xR_2^{\mathcal{U}}y \iff x <_R y \iff \neg(yRx). \end{aligned}$$

Therefore, the model $\mathcal{U}$ satisfies ϕ_{lt}, because we can verify as follows:

1. For all $x \in U$, $x =_R x$ holds.

2. For all $x, y \in U$, $x =_R y$ if and only if $y =_R x$.

3. For all $x, y, z \in U$, if $x =_R y$ and $y =_R z$, then $x =_R z$.

4. For all $x, y \in U$, if $x =_R y$, then $x <_R y$ does not hold.

5. For all $x, y \in U$, if $x =_R y$ does not hold, then either $x <_R y$ or $y <_R x$.

6. For all $x, y, z \in U$, if $x <_R y$ and $y <_R z$, then $x <_R z$.

7. For all $x \in U$, there exists $y \in U$ such that $x <_R y$, since any subset of U has no maximum element.

Problem 2. Prove that the Halting problem with empty input

$$\mathrm{HALT}_\varepsilon = \{\langle M \rangle \mid M \text{ halts on empty input.}\}$$

is undecidable.

Solution. We prove it by contradiction. Suppose that $\mathrm{HALT}_\varepsilon$ is decidable, then there exists a Turing machine H that decides $\mathrm{HALT}_\varepsilon$. Construct a new Turing machine D as follows:

1. Obtain its own description $\langle D \rangle$ by recursion theorem.

2. Run H on input $\langle D \rangle$.

3. If H accepts, then loop; otherwise, halt.

You can see that D halts on empty input if and only if H accepts $\langle D \rangle$, which leads to D looping at step 3, meaning D does not halt on empty input. This is a contradiction and thus $\mathrm{HALT}_\varepsilon$ is undecidable.

Problem 3. Show that any infinite subset of $\mathrm{MIN}_{\mathrm{TM}}$ is not Turing-recognizable where $\mathrm{MIN}_{\mathrm{TM}}$ is a language defined in the class.

Solution. We prove it by contradiction. Suppose that there is an infinite subset $\mathrm{SMIN}_{\mathrm{TM}}$ of $\mathrm{MIN}_{\mathrm{TM}}$ that is Turing-recognizable, then there exists a Turing machine E that enumerates $\mathrm{SMIN}_{\mathrm{TM}}$. We construct the following Turing machine C:

1. Obtain its own description $\langle C \rangle$ by recursion theorem.

2. Run the enumerator E until machine D appears with a longer description than $\langle C \rangle$.

3. Simulates D on input w.

Because $\mathrm{SMIN}_{\mathrm{TM}}$ is infinite, E's list must contain a longer description than $\langle C\rangle$. Therefore, C eventually terminates at step 2 with some machine D that has a longer description. And C simulates D on input w, so they are equivalent. Because C is shorter than D and is equivalent to it, so D cannot be minimal; but D appears on E's list.

This is a contradiction and thus $\mathrm{SMIN}_{\mathrm{TM}}$ is not Turing-recognizable.

Problem 4.

(a) Prove a special case of the S-m-n theorem, the Currying technique for Turing machines. That is, show that there is a computable function $S_1^1 : \Sigma^* \times \Sigma^* \to \Sigma^*$ mapping the description of Turing machine T and input x to the description of a Turing machine S such that (1) S on input y computes the same output as T on input $\langle x, y\rangle$ if T halts; and (2) S loops on input y if T loops on input $\langle x, y\rangle$.

(b) Prove Kleene's recursion theorem by item (a) and Roger's fixed-point theorem.

Solution. (a) Given a Turing machine T and an input x, we can construct a Turing machine S that $S_1^1(\langle T\rangle, x) = S$ as follows:

1. On input y, S simulates T on input $\langle x, y\rangle$.

If the simulation of T on input $\langle x, y\rangle$ halts, then S halts and outputs the same result as T. Otherwise, S loops on input y as well. S satisfies the requirements, and S_1^1 is a computable function. Therefore the Currying technique is proved.

(b) Let T be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \to \Sigma^*$. From item (a), we can construct a Turing machine $S_w = S_1^1(\langle T\rangle, w)$ that computes a function $s_w : \Sigma^* \to \Sigma^*$, such that $t(x, w) = s_w(x)$.

From Roger's fixed-point theorem, since $s_w : \Sigma^* \to \Sigma^*$ is a computable function, then there is a Turing machine R for which $s_w(\langle R\rangle)$ describes a machine equivalent to R. Note that $s_w(\langle R\rangle) = t(\langle R\rangle, w)$, and s_w is a function that implicitly relies on w, so s_w could be seen as a function of w, denoted as $r(w)$.

Therefore, we can construct a Turing machine R that has a computing function $r : \Sigma^* \to \Sigma^*$, such that for every w, $r(w) = t(\langle R\rangle, w)$.