Homework 4

Problem 1. Give a model for the sentence

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

Solution. Consider the model $\mathcal{U}$ as such:

1. Universe $U = \mathbb{Z}$;
2. Relation $R_1 = \{(x,y)|x=y\}$;
3. Relation $R_2 = \{(x,y)|x<y\}$.

Then the first three lines of the condition are satisfied by the reflexive, symmetric and transitive properties of "$=$". The fourth and fifth lines are satisfied by the trichotomy of the strict total order "$<$". The sixth line is the transitive property of "$<$". The seventh line is satisfied by the fact that there is no biggest integer.

Problem 2. Prove that the Halting problem with empty input

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

is undecidable.

Solution. Suppose that HALT_ε is decidable, and TM H is the always-halting Turing Machine that recognize HALT_ε

Construct another TM B, which, on any input:

1. Obtain its own code $\langle B \rangle$;
2. Run $H(\langle B \rangle)$;

   (a) if $H(\langle B\rangle)$ accepts, loop;
   (b) if $H(\langle B\rangle)$ rejects, halt.

Because H always halts we can always get the answer of $H(\langle B\rangle)$ in finite time, So the behavior of B is well-defined.

Then we can consider the question of whether B halts on empty input. If B halts, then $\langle B\rangle \in \text{HALT}_\varepsilon$, so $H(\langle B\rangle)$ should accept. This lead $B(\varepsilon)$ to case 2.(a), and so $B(\varepsilon)$ should loop, which is a contradiction.

If B loops on empty input, then $H(\langle B\rangle)$ should reject, and $B(\varepsilon)$ should go into case 2.(b) and halt, where contradiction also occurs.

Thus we can conclude that such always-halting H does not exist, and HALT_ε is undecidable.

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

Solution. Suppose $S \subset \text{MIN}_{\text{TM}}$ is an infinite TM-recognizable subset, and TM R is the machine that can recognize strings in S. Now we will construct an enumerator E that enumerates all TMs in S. Because the set of all TMs are countable (since they are all finite-length strings), we can denote them as $M_1, M_2, M_3, \ldots$. The enumerator E can be constructed as follows:

Algorithm 1: Enumerator for S

```
for i = 1, 2, 3, ... do
    Run R on M_1, M_2, ..., M_i for i steps;
    Print all M_j that R accepts within i steps and hasn't been printed
      before.
end
```

As all strings in S can be accepted by R in finite steps, say $M_k \in S$ is accepted by R in m steps, then M_k will be enumerated by E in the max m, k-th iteration. Thus all TMs in S will be enumerated by E in finite steps.

Based on the enumerator E, we can construct a TM C as such:

On input w:

1. obtain its own code $\langle C\rangle$;
2. run E until a machine D appears such that $|\langle D\rangle| > |\langle C\rangle|$;
3. simulate D on w.

The TM D with a longer description is can always be found in finite steps by the enumerator because S is infinite. If the description length of TMs in S has an upper bound then the number of elements in S will be limited. So we can ensure the TM C is well-defined.

Meanwhile we can see that TM C is equivalent to D and has a shorter description, which means that D isn't the minimal machine that conduct what it does. This raises a contradiction to $D \in S \subset \mathrm{MIN}_{\mathrm{TM}}$. Thus we can conclude that S 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) Consider TM M as such:
On input $\langle\langle T\rangle, x\rangle$:

  (a) Construct TM S: "On input y, simulate T on input $\langle x, y\rangle$";

  (b) Print $\langle S\rangle$.

(b) We want to prove that for any computable function $t : \Sigma^* \times \Sigma^* \to \Sigma^*$, there exists a TM R for computable function $r : \Sigma^* \to \Sigma^*, r(w) = t(\langle R\rangle, w)$. For any such t, by item (a) we have TM S_1^1 $s.t.$ $S_1^1(\langle T\rangle, x) = S$ where $S(y) = T(x, y)$. Since S_1^1 itself is computable we can construct, by item (a), another TM S_T $s.t.$ $S_T(w) = S_1^1(\langle T\rangle, w)$.
By the Roger's fixed-point theorem, there exists TM R such that $S_T(\langle R\rangle)$ describes a TM equivalent to R, which means R is equivalent to $S_T(\langle R\rangle) = S_1^1(\langle T\rangle, \langle R\rangle) = S$ where $S(y) = T(\langle R\rangle, y)$. So for any input w, $R(w) = S(w) = T(\langle R\rangle, w)$, which is the desired result.