ITCS 532: W4 Homework Solutions

Rob Egrot

Q1

Let L_1 and L_2 be disjoint r.e. languages. Suppose $L_1 \cup L_2$ is recursive. Prove that L_1 and L_2 are both recursive.

We will describe an algorithm for deciding L_1.

1. Given a string x we can decide if $x \in L_1 \cup L_2$, as this language is recursive.
2. If $x \notin L_1 \cup L_2$ then $x \notin L_1$, so reject.
3. If $x \in L_1 \cup L_2$ then it must be in either $L_1 \setminus L_2$, or $L_2 \setminus L_1$. Use dovetailing to simultaneously run the algorithms that semidecide L_1 and L_2 on x.
4. If $x \in L_1$ then accept.
5. If $x \in L_2$ then reject.

We can decide L_2 similarly.

Q2

Let D be the decision problem "Given a Turing machine T and input I, does $T(I)$ halt within 100 steps?". Then there is an associated formal language

$$L_D = \{\mathbf{code}(T, I) : T \text{ halts on } I \text{ within 100 steps}\}$$

Which of the following is true? i) L_D is recursive, ii) L_D is r.e but not recursive, iii) L_D is not r.e.

- L_D is recursive.
- To decide L_D we use a Turing machine that, given input $\mathbf{code}(T, I)$ simulates $T(I)$, and also maintains a counter track of the number of steps that have been simulated.
- If this simulation halts before the counter reaches 100 then the input is accepted.
- If it does not (or if the input is not in the correct format), then it rejects.

Q3

Let D be the decision problem "Given a Turing machine T, does T halt on every input I within 100 steps?". What is the formal language L_D associated with D?

$\{\textbf{code}(T) : T$ is a Turing machine and $T(I)$ halts within 100 steps for all $I\}$.

Q4

With D as in Q3 prove that L_D is recursive.

- Note that $T(I)$ halts for all I within 100 steps if and only if $T(I)$ halts for all I of length ≤ 100 within 100 steps, as T can never read past the first 100 symbols within 100 steps.
- Now, the number of strings over a finite alphabet whose length is ≤ 100 is finite, so we can check $T(I)$ for each such string I using the algorithm from Q2.
- If the answer is no for any I we reject $\mathbf{code}(T)$, and if the answer is yes for all I we accept $\mathbf{code}(T)$.

Q5

Let HAI be the decision problem "Given T does T halt for all inputs?". Then an instance of HAI is a Turing machine T.

a) What is an 'instance' of the Halting Problem?

b) If M is a Turing machine and I is an input for M let M_I be a machine that first erases its input then simulates $M(I)$. Show that $M(I)$ halts if and only if $M_I(J)$ halts for all inputs J, and $M(I)$ runs forever if and only if $M_I(J)$ runs forever for all J.

c) Prove that the Halting Problem reduces to HAI.

d) What does this tell us about the decidability of HAI?

Q5

a) What is an 'instance' of the Halting Problem?

- A pair (M, I) where M is a Turing machine and I is a finite string over its alphabet.

b) If M is a Turing machine and I is an input for M let M_I be a machine that first erases its input then simulates $M(I)$. Show that $M(I)$ halts if and only if $M_I(J)$ halts for all inputs J, and $M(I)$ runs forever if and only if $M_I(J)$ runs forever for all J.

- By definition $M(I)$ halts if and only if $M_I(J)$ halts for all J.
- So the contrapositive statement says that $M(I)$ runs forever if and only if $M_I(J)$ runs forever for some J.
- But $M_I(J)$ does the same thing for all J.
- So $M(I)$ runs forever if and only if $M_I(J)$ runs forever for all J.

Q5

c) Prove that the Halting Problem reduces to HAI.

- Given an instance (M, I) of HP we construct an instance M_I of HAI as described.
- We have just proved that (M, I) is a yes instance of HP if and only if M_I is a yes instance of HAI.

d) What does this tell us about the decidability of HAI?

- As $HP \leq HAI$, and HP is undecidable, it follows that HAI is undecidable.