3004
Sample Problems and Solutions

Mark Herbster
m.herbster@cs.ucl.ac.uk

This document contains problems similar to Hw #1 and sample solutions.

1. *The "character-position" problem*
   Given the decision problem,

   "Given a TM M and input I and a character σ', and tape position k does M ever write character σ' at tape cell k when processing I?"

   $$L = \{\text{code}(M)\text{code}(I)\text{code}(\sigma')\text{code}(k) : M \text{ writes } \sigma' \text{ at tape cell } k \text{ while processing } I\}$$

   Which of three possibilities holds

   (a) L is recursive.
   (b) L is recursively enumerable but not recursive.
   (c) L is not recursively enumerable.

2. *The "move left" problem*

   Given the decision problem,

   "Given a TM M and input I does M ever move the tape head left"

   $$L = \{\text{code}(M)\text{code}(I) : M \text{ moves the tape head left while processing input } I\}$$

   Which of three possibilities holds

   (a) L is recursive.
   (b) L is recursively enumerable but not recursive.
   (c) L is not recursively enumerable.

3. *The "finite" problem*

   Given the decision problem,

   "Given a TM M is the language that it semi-decides finite?"

   Let $LA(M)$ denote the language a TM M semi-decides (i.e., $LA(M)$ is simply the set of strings that M accepts) then consider a corresponding language L_{fin}, which is

   $$L_{\text{fin}} = \{\text{code}(M) : LA(M) \text{ is a finite set.}\}$$

   Which of three possibilities holds

   (a) L is recursive.
   (b) L is recursively enumerable but not recursive.

(c) L is not recursively enumerable.

Solutions:

1. **Problem 1:**
The language,

$$L = \{\text{code}(M)\text{code}(I)\text{code}(\sigma')\text{code}(k) : M \text{ writes } \sigma' \text{ at tape cell } k \text{ while processing } I\}$$

is recursively enumerable but not recursive (b).

Proof The proof consists of two parts, first we prove that L is recursively enumerable then we prove it is not recursive.

The language L is recursively enumerable. Consider the following machine M' which semi-decides L. Construct M' as follows: when given the input code(M)code(I)code(σ')code(k), the machine M' then simulates M on input I; if the simulated M ever writes σ' at tape cell k (simulated tape) then M' halts itself; otherwise M' continues the simulation. TM M' thus semi-decides L, therefore L is r.e.

The language L is not recursive. We suppose L is recursive and we show that this leads to a contradiction (we "solve" the halting problem).

Suppose L is recursive then there exists a machine M_{cp} which decides L.

Consider the following construction of a TM M'', which will be used below. The machine M'' takes an instance of the halting problem, machine M and input I (code(M)code(I)). Assume without loss of generality that for machine M, $0 \in \Sigma$. Machine M'' then simulates the machine M on input I (a nonproblematic requirement of the "simulator" is that it never writes in cell position 1 of its own tape). If the simulation of M on I halts, then the machine M'' moves its own tape head to cell position 1 and writes 0 and halts itself. Otherwise, the simulation of M on I continues indefinitely.

Now we construct a machine M_{hp} which when given an instance of the halting problem M and input I (code(M)code(I)) operates as follows: first it constructs a machine M'' with M and I as described above. Then M_{hp} pipes $[M'', MI, 0, 1]$ to a simulation of machine M_{cp}, i.e. code(M'')code(code(M)code(I))code(0)code(1). Then if the simulation of M_{cp} halts in the Y state, M_{hp} halts itself in the Y state and if the simulation of M_{cp} halts in the N state then M_{hp} halts itself in the N state. Notice that by the detailed construction, that M_{hp} halts in Y if M would halt on I and M_{hp} halts in N if M does not halt on I. Thus we have shown that the halting problem language is recursive. This is a contradiction, thus the supposition L is recursive is false. □

2. **Problem 2:**
The language L,

$$L = \{\text{code}(M)\text{code}(I) : M \text{ moves the tape head left while processing input } I\}$$

is recursive (a).

Proof First some notation, let k be the length of input I, let ℓ the number of 4-tuples in TM M's transition function.

We prove L is recursive by designing a TM M' which can determine (in a finite number of steps) if the "tape head" of machine M will ever move left while processing I. The operation of M' can be split into two distinct phases: in the first phase M is "processing" the k input characters; in the second phase M has moved to the first blank past the input.

The TM M' which takes as input machine M and string I (code(M)code(I)) is constructed to operate as follows. In *Phase 1* TM M' simulates the machine M on input I for $(k+1)\ell$ steps after which one of the four following conditions will be true of the simulation of machine M on I,

(a) Machine M will have moved left while processing I.

(b) Machine M will have halted.

(c) Machine M will have entered an infinite loop.

(d) Machine M will have moved to the first blank character to the right of the input.

If either option a or b has occured then M' halts the simulation of M and halts in either **Y** or **N** respectively. Let's examine the simulation in more detail now. If option a or b haven't occurred, notice that while processing a single tape cell we must either move right after ℓ processing steps or the simulation is in an infinite loop, i.e., there are only ℓ tuples and if we do not move left or right, after ℓ steps we must "match" a tuple that we have already seen; therefore the simulation will enter an infinite loop, hence M' can halt its simulation of M and halt itself in **N**. If the tape head moves right, notice that that it can only do this $k+1$ times (input length k plus initial blank) until it has moved past the input I. Thus if option a,b, or c do not hold within $(k+1)\ell$ steps, option d holds and we move into *phase 2* of our processing.

Phase 2: The machine M' continues simulating M on I. After ℓ^2 steps (in phase 2) one of the three following conditions will be true of the simulation of machine M on I,

(a) Machine M will have moved left while processing I.

(b) Machine M will have halted.

(c) Machine M will have entered an infinite *pattern*.

If either a or b occurs then M' halts in **Y** or **N** respectively. Otherwise option c holds and M' halts the simulation of M and halts itself in **N**. Let's examine why option c will occur after ℓ^2 steps.

Let q'_i denote the state of M when it first moves to the ith cell after the input. When first moving into the ith cell notice that it is always blank. By our argument for phase 1, after ℓ steps we are either in an infinite loop or we will have moved right to the $i+1$th cell following the input. Notice that after we have moved right ℓ times (past the input) the next time the tape head moves right we necessarily enter a blank cell in a state $q'_{\ell+1}$ which has already been entered before on a blank cell, which means that M has entered a simple pattern. Thus if after $l \times l$ steps, M has not moved left or halted, then M will never move left or halt, thus M' halts the simulation of M and halts itself in the **N** state. □

3. **Problem 3:**

The language,

$$L_{\text{fin}} = \{\text{code}(M) : LA(M) \text{ is a finite set.}\}$$

is not recursively enumerable (c).

Definition 1.1 *The "not halting" language is defined to be,*

$$L_{nhp} = \{code(M)code(x) : \textit{Turing machine } M \textit{ does not halt on input } x\}.$$

Lemma 1.2 *The language L_{nhp} is not r.e.*

Proof This proof is left for the reader. Hint: The proof is analogous to the proof that the halting problem is unsolvable by reducing the self-acceptance problem to the halting problem (lecture 10, slide 3) except that "non-self acceptance (NSA)" reduces to not halting problem. □

Proof Suppose L_{fin} is recursively enumerable. Then there exists a TM M_{fin} that semi-decides L_{fin}. We proceed by showing that there then exists a TM M_{nhp} which semi-decides L_{nhp}, which contradicts the lemma above, thus our supposition is false.

Given an instance of the not halting problem M and x, construct a machine $M'_{(M,x)}$ that operates as follows. The machine $M'_{(M,x)}$ ignores any input and simulates machine M on input x, only halting if M halts on x. Hence the language of $M'_{(M,x)}$ is

$$LA(M'_{(M,x)}) = \begin{cases} \Sigma^* & \text{if } M \text{ halts on } x \\ \emptyset & \text{if } M \text{ does not halt on } x \end{cases}$$

Now we construct the machine M_{nhp}. TM M_{nhp} takes machine M and input x as input (code(M)code(x)). It then constructs TM $M'_{(M,x)}$ and "pipes" M' (code(M')) as the input to TM M_{fin}. Notice that M_{nhp} halts if M would not halt on x and M_{nhp} fails to halt if M would halt on x. Hence we've constructed a TM that semi-decides L_{nhp}. This contradicts the lemma above, hence our supposition that a TM M_{fin} exists is false, thus L_{fin} is not r.e. □