CSC320 Assignment Four Report

2018-07-22

Name: Zhaocheng Li

ID: V00832770

Question 1:

*Proof idea: Proving from two directions, applying the proof of definition on the Turing-recognizability and enumerator shown on slide 11th.*

*Assumption: Suppose we hypothetically have a TM M that decides L, and an enumerator E for L, such that E respects ∝ ordering.*

*PROOF:*

*(>>>). Given the TM M that decides L, we can try to construct such E for L.*

1. *E ignores any input*
2. *For I = 1, 2, 3, …:*

* *For each possible string s in {0,1}\* of length less than or equal to I:*

1. *Run M on s for I steps.*
2. *If M accepts, print s.*

*Then such E exists with all outputs is printed in ∝ order, and they belong to L.*

*(<<<). Given the enumerator E for L and it respect the ∝ ordering. We can try to construct a TM M that decides L.*

*Here there are two cases: L is finite, or L is infinite.*

*Case 1: suppose L is finite. Then it is decidable because all finite languages are decidable, by theorem. Such TM M that decides L obviously exists.*

*Case 2: If A is infinite, a decider for L works as follow:*

1. *Take w as the input of M, then the decider enumerates all strings in L in order until some string in order after w appears. This is because L is infinite.*
2. *If w has shown in the enumeration already, accept; if it has not appeared yet, it never will, reject.*

*Then the TM M for L is made.*

*Proof done.*

Question 2:

Proof idea: following the definition of undecidability of ATM and the hint.

PROOF:

We know that

L = {<M> | M when started on the blank tape, eventually writes a $ somewhere on the tape}

And by the hint (i.e., give a computable reduction f, such that f (<M, w>) = <M1>.

For any M, string w we can let M1 be the TM which takes as input string x:

1. If w = x, M1 runs M on input w and accepts if M accepts;
2. If w ≠ x, M1 rejects;

Now we construct TM S to decide ATM. Let R be a hypothetical TM which decides L:

S has input <M, w>

1. Use <M, w> to construct <M1> as described above;
2. Run R on <M1> and accept if and only if R accepts.

If R decides L, then S decides ATM. Therefore, R cannot exist and L is undecidable.

Question 3:

Proof Idea: Hint on the computable reduction f

PROOF:

Given the HALTTM and L, construct f that HALTTM ≤m L

For any M, string w we can let M1 be the TM:

1. If w is less than j 1’s, M1 runs M on input w and accepts if M accepts;
2. If w is more than j 1’s, M1 rejects;

Now we construct TM S to decide HALTTM. Let R be a hypothetical TM which decides L:

S has input <M, w>

1. Use <M, w> to construct <M1> as described above;
2. Run R on <M1> and accept if and only if R accepts.

If R decides L, then S decides HALTTM. Therefore, R cannot exist and L is undecidable.

Question 4:

PROOF:

Since ETM = {<M> | M is a TM and L(M) = φ},

then we ETM\* (meaning complement) = {<M> | M is a TM and L(M) ≠ φ}.

For a nondeterministic TM H for ETM\*,

For i = 1, 2, 3, …:

1. Run M for i steps, non-deterministically choose an input x1 ∈ ETM\*,
2. H accepts if M accepts on any input.

Question 5:

Proof Idea: applying reduction with ETM

PROOF:

Suppose H = {<M1, M2> | L(M1) ∩ L(M2) = φ}

We show ETM ≤m H.

Define f as follow: f(<M, w>) = <M1, M2>, where M1, M2 are machines such that:

1. M1 accepts all inputs;
2. For any input x, M2 runs M on w and rejects if M accepts.

If <M, w> ∈ ETM then M accepts w. But then L(M1) = φ and L(M2) = Σ**\***, so <M1, M2> ∈ H.

If <M, w> ∉ ETM then L(M1) ∩ L(M2) ≠ φ, so <M1, M2> ∉ H.

So, f is a mapping reduction from ETM to H, and H is not recognizable since ETM isn’t.

Question 6:

Proof Idea: Using the hint.

PROOF:

Now we know that

T = {<M> | M is a reversible TM}

And by the hint (i.e., give a computable reduction f, such that f (<M, w>) = <M1>.

For any M, string w we can let M1 be the TM which takes as input string x:

1. If w = x = xR, M1 runs M on input w and accepts if M accepts;
2. If x ≠ xR, M1 runs M on input w and accepts x if M accepts xR;
3. If w ≠ x, M1 rejects;

Now we construct TM S to decide ATM. Let R be a hypothetical TM which decides T:

S has input <M, w>

1. Use <M, w> to construct <M1> as described above;
2. Run R on <M1> and accept if and only if R accepts.

If R decides T, then S decides ATM. Therefore, R cannot exist, and T is undecidable.