**ECE374 Assignment 9**

Due 04/24/2023

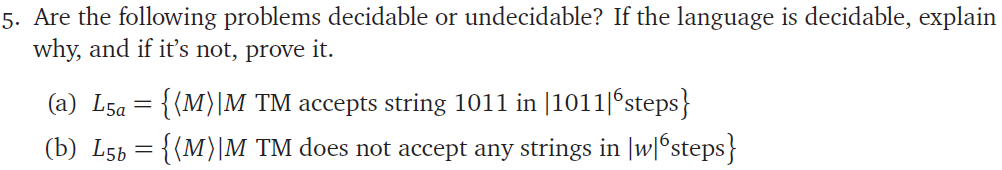
**Group & netid**

*Chen Si chensi3*

*Jie Wang jiew5*

*Shitian Yang sy39*

**Problem 5**



(a)

Solution:

This problem is decidable.

We could build the following decider:

It takes in a Turing Machine M, runs 1011 on it for |1011|6 steps, and check if it results on an accepting state. If M ends on an accepting state, it returns accept and if M doesn’t end on an accepting state, it returns reject.

<M>

Run 1011 on M for |1011|6 steps

<M>

accept

reject

Check if on accepting state

accept

reject

As the length of |1011| is fixed, we are supposed to finish this loop within |1011|6 steps and it’s thus decidable.

(b)

Solution:

This problem is undecidable.

We could construct a reduction from Halting to L5b:

Halting: {<M, w> | M is a TM and M halts on w}

We could construct the following reduction graph

<M’>

<M, w>

<M, w>

accept

reject

accept

reject

ORAC

L5b

Construct TM M’

(see details below)

We construct a Turing Machine M’ given <M, w>:

**M’** (M, w):

Run M on w for |w|6 steps

if M halts within |w|6 steps:

reject

else:

accept

Analysis:

(1) If M halts on w, M’ would halt on w within |w|6 steps, and M’ would reject.

In this case, M’ doesn’t accept any string within |w|6 steps, and ORAC\_L5b would return accept. Therefore, an Accepting case for Halting problem, where M halts on w, would be solved by calling ORAC\_L5b.

(2) If M doesn’t halt on w, M’ would not halt on w within |w|6 steps, and M’ would accept on any input string. In this case, ORAC\_L5b would return reject. Therefore, an Rejecting case for Halting problem, where M doesn’t halt on w, would also be solved by calling ORAC\_L5b.

Given that Halting is undecidable and we reduced Halting to L5b, we could prove that L5b is undecidable.