# **University of Calgary**

# 

# **CPSC 313: Introduction to Computability, Winter 2018**

# 

# **Assignment #3**

**For**

**Samin Sabokrohiyeh, Dr. Wayne Michael Eberly**

**By**

**Jesus Cuadra (ID: 30002376 | Email:** [**jesus.cuadrakaram@ucalgary.ca**](mailto:jesus.cuadrakaram@ucalgary.ca)**)**

**BenKun Chen (ID: 30005337 | Email:** [**benkun.chen@ucalgary.ca**](mailto:benkun.chen@ucalgary.ca)**)**

**Adrienne Lee (ID: 10087046 | Email:** [**adrienne.lee1@ucalgary.ca**](mailto:adrienne.lee1@ucalgary.ca)**)**

**1. A RAM Turing Machine is a Turing machine that has random access memory, along with one or more work tapes.**

1. **Show that if there is a (standard) multi-tape Turing machine *M* with a given language then there is a RAM Turing machine with language *L* as well. Furthermore, if *M* decides *L* then decides *L* as well.**

Let *M* be a (standard) multi-tape Turing machine with a given language then there is a RAM Turing machine with language *L* as well.

As a RAM turing machine is described above in the question, the machine described:

* **Accepts** a string , when **.**
* **Rejects** a string , when **.**
* **Loops** on any other string.

As such, decides *L*.

1. **Conversely, show that if there is a (RAM Turing machine *M* with a given language then there is a (standard) multi-tape Turing machine with language *L* as well. Furthermore, if *M* decides *L* then decides *L* as well.**

In order to show that if M decides L then decides L as well, we need to prove that ifthenand if thenWe first need to design a Turing machine which achieves the functionalities that Ram Turing machine has.

Suppose that is the language of the Turing machine M, where

Consider a Multi-Tape Turing machine

* where k = total number of tapes

If the the machine enters state , then

Tape ‘A’(stores all the values from array A, with length n)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | …... | |  | ப | ... |

**-- size of tapes that splits each digit of the unpadded decimal represented integer --**

Tape that stores the last significant number (x) of ‘\*’s which read from

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ப | ப | ப | ப | ப | ப | ப | ப | ப | ... |

Tape #1

Tape that stores the second last significant number (x10) of ‘\*’s which read from

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ப | ப | ப | ப | ப | ப | ப | ப | ப | ... |

Tape #2

Tape that stores the third last significant number (x) of ‘\*’s which read from

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ப | ப | ப | ப | ப | ப | ப | ப | ப | ... |

Tape #3

…….

Tape that stores the first significant number (x) of ‘\*’s which read from

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ப | ப | ப | ப | ப | ப | ப | ப | ப | ... |

Tape #

To illustrate, if the machine reads . The tapes will look like as follow:

Tape that stores the last significant number 3 ‘\*’s which read from

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | \* | ப | ப | ப | ப | ப | ப | ... |

Tape #1

Tape that stores the second last significant number (110) of ‘\*’s which read from

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | \* | \* | \* | \* | \* | \* | \* | \* | ப | ... |

Tape #2

Tape that stores the third last significant number (1) of ‘\*’s which read from

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* | \* | \* | \* | …... | | \* | ப | ப | ... |

Tape #3

**Case 1:** If the non-blank part of the address tape stores a string , we create the work tape:

Work tape(designated the address tape)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | R | ப | ப | ப | ப | ப | ப | …... | |

On an input

1. Scans through the work tape, record the value into n tapes that splits each digit of the unpadded decimal represented integer and erase at the same time.
2. Locates the entry A[i], write onto the leftmost cell of the work tape.

**The details of step 1 are as follows:**

**1(a).** First, we need to create size of tapes, in which is the total number of elements in the tape from the leftmost cell to the left of ‘R’ symbol. When reading in tape ‘A’, if is visible on the tape, for a symbol then create new tape, moving right, and change to a state . Go to step 1(b).

If ப is visible on the tape, stop reading. Change to state . Go to step 1(d).

**1(b).** After we have all the tapes ready, we now need to scan the Work tape. Read and skip through all the numbers from until we see ‘R’. If R is visible on the tape, for a symbol then replace it with ப, moving left. Now, if is visible on the tape, for a symbol then replace it with ப and record it into the first tape by initializing (digit read) of ‘\*’s in the tape, moving left. Go to step 1(c).

**1(c).** If is visible on the tape, for a symbol then replace it with ப and record it into the following tape by initializing (digit read ) of ‘\*’s in the tape, moving left. Repeat the step until reach the leftmost cell, stop reading.

**1(d).** Moving left until reach the leftmost cell, stop reading. Go to step 2(a).

**The details of step 2 are as follows:**

**2(a).** Move tape heads from the start configurations of tape ‘A’ and the first tape created which stores the significant number (x) of ‘\*’s which read from . If ‘\*’ is visible on the first tape, move tape heads of tape ‘A’ and second tape to the right. Go to

**2(b).** Keep scanning. When ‘\*’ is visible on the tape, move tape heads of tape ‘A’ and second tape to the right.

If ப is visible on the tape, stop reading. Change to state . Go to step 2(c).

**2(c)**. Jump to the next tape which stores the significant number (x) of ‘\*’s which read from . Go to step 2(b).

If ப is visible on the start configuration of the tape, go to step 2(c).

Else if ப is visible on the start configuration of the tape and it is the last tape, stop reading. Go to step 2(d).

**2(d).** Write the value in tape ‘A’ in which it’s tape head is pointing to the leftmost cell of the work tape. Go to state .

**Case 2:** If the non-blank part of the address tape stores a string , we create the work tape:

Work tape(designated the address tape)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | W |  | ப | ப | ப | ப | ப | …... | |

On an input

1. Scans through the work tape, record the value into n tapes that splits each digit of the unpadded decimal represented integer and erase at the same time.
2. Locates the entry A[i], write found on the work tape onto the entry A[i]

**The details of step 1 are as follows:**

**1(a).** First, we need to create size of tapes, in which is the total number of elements in the tape from the leftmost cell to the left of ‘R’ symbol. When reading in tape ‘A’, if is visible on the tape, for a symbol then create new tape, moving right, and change to a state . Go to step 1(b).

If ப is visible on the tape, stop reading. Change to state . Go to step 1(d).

**1(b).** After we have all the tapes ready, we now need to scan the Work tape. Read and skip through all the numbers from until we see ‘W’. If W is visible on the tape, for a symbol then replace it with ப, moving left. Now, if is visible on the tape, for a symbol then replace it with ப and record it into the first tape by initializing (digit read) of ‘\*’s in the tape, moving left. Go to step 1(c).

**1(c).** If is visible on the tape, for a symbol then replace it with ப and record it into the following tape by initializing (digit read ) of ‘\*’s in the tape, moving left. Repeat the step until reach the leftmost cell, stop reading.

**1(d).** Moving right and skip through all ப s until we see , go to step 2(a)

**The details of step 2 are as follows:**

**2(a).** Move tape heads from the start configurations of tape ‘A’ and the first tape created which stores the significant number (x) of ‘\*’s which read from . If ‘\*’ is visible on the first tape, move tape heads of tape ‘A’ and second tape to the right. Go to

**2(b).** Keep scanning. When ‘\*’ is visible on the tape, move tape heads of tape ‘A’ and second tape to the right.

If ப is visible on the tape, stop reading. Change to state . Go to step 2(c).

**2(c)**. Jump to the next tape which stores the significant number (x) of ‘\*’s which read from . Go to step 2(b).

If ப is visible on the start configuration of the tape, go to step 2(c).

Else if ப is visible on the start configuration of the tape and it is the last tape, stop reading. Go to step 2(d).

**2(d).** Write the value in the Work tape onto the cell where the tapehead of tape ‘A’ is pointing. Replace in the Work tape with ப, and move the tapehead to the leftmost cell. Go to state .

**Case 3:** If the non-blank part of the address tape stores a string different than the previous 2 cases:

On an input

1. Scans through the work tape, replacing all symbols with a ப.
2. Moves the tapehead to the leftmost cell of the Work tape, then go to state .

From the description above, we have designed the multi-tape turing machine which shows that if there is a multi-tape Turing machine M with a given language then there is a RAM Turing machine with language *L* as well. Ifthenand if then

**2.** **Consider the problem of determining whether a given single-tape Turing machine *M* (with some input alphabet ) ever writes a blank symbol over a non-blank symbol during the course of its computation on a given input string This can be formulated as a language . Prove that is undecidable.**

Given the claim in *Lecture #25* on *Claim #3*, it proves that . Given the claim in *Lecture #25*, we can show that .

Given the proof on is undecidable in *Tutorial Exercise #17*, it is proven that is undecidable.

A many-one reduction will be given to show that is many-one reducible to . Since is undecidable it will follow that is undecidable as well.

Consider a total function :

* If but then , (assuming that is not in ).
* Suppose that , so that is the encoding of some Turing machine . Now consider another Turing machine, .

that follows:

* for some integer
  + includes three new states:
    - : this new state writes a blank symbol, replace the blank symbol with a nonblank symbol that is not in
    - : this new state reads a blank symbol, replace the blank symbol with a nonblank symbol
    - : this new state overwrites the previous blank symbol and writes a nonblank symbol
* for some integer and for for some integer
* , where

given the three new states the transitions will be as follows:

* for all
* for all
* for all

Now, we need to show that if and then .

Suppose that . Then encodes a Turing machine *M* that accepts at least one string

Now, we need to show that if and then .

Using the fact that .

Suppose that and , then either or , where encodes a Turing machine *M*:

Given two cases:

* Case 1: where
  + Suppose Then , so that Since it follows
* Case 2: where
  + Suppose where encodes a Turing machine M such that

Now, we need to show that *f* is a computable function.

Since the language is decidable it is possible to ask whether in an algorithm that computes from . Now if then this establishes the claim that can be computed.

It shows that *f* is a computable function, where *f* has all the properties of a many-one function, which follows that .

Since is undecidable, it follows that is undecidable as well.