CSE211-Formal Languages and Automata Theory

U4L6_Universal Language Part 2

Dr. P. Saravanan

School of Computing

SASTRA Deemed University

Agenda

- Recap of previous class: Diagonalization
- Converting the Problem to a Language
- Binary-Strings from TM's
- TM Representation by binary
- Language representation by binary
- Universal Language
- Universal Turing Machine

TM Representation by binary

- Represent a TM by concatenating the codes for each of its moves, separated by **11** as punctuation.
  - That is: $Code_1 11 Code_2 11 Code_3 11$ …

	0	1	B
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	(q_2, B, L)
q_2	$(q_3, 0, R)$	-	-
q_3	-	-	-

What is the L(M)?

Coding for the above table:

1110101010100**11**0100101001001**11**0100010010010**11**001010001010***111***

Are the followings correct encoding of a TM?

0110000111000 1

111111

Language Representation by binary string

- **Definition:**

$$L_t = \{x \mid x \text{ is in } \{0, 1\}^* \text{ and } x \text{ encodes a TM}\}$$

- Question: Is L_t recursive?
- Answer: Yes. [Check only for format, i.e. the order and number of 0's and 1's, syntax checking]

- Question: Is L_t decidable:
- Answer: Yes (same question).

The Universal Language

- An example of a recursively enumerable, but not recursive language is the language L_u of a *universal Turing machine*.
- That is, the UTM takes as input the code for some TM M and some binary string w and accepts if and only if M accepts w.

The Universal Language

- We can define the language L_u as follows:

  L_u = {x | x is in {0, 1}* and x = <M,w> where *M* is a TM encoding and *w* is in L(M)}

- Let *x* be in {0, 1}*. Then either:
  1. *x* doesn't have a TM prefix, in which case *x* is **not** in L_u
  2. *x* has a TM prefix, i.e., x = <M,w> and either:
     a) *w* is not in L(M), in which case *x* is **not** in L_u
     b) *w* is in L(M), in which case *x* is in L_u

Universal TM (UTM)

- A limitation of Turing Machines:

Turing Machines are "hardwired"

they execute
only one program

- Real Computers are re-programmable
- Solution: Universal Turing Machine

Universal TM (UTM)

- Attributes of UTM
  - Reprogrammable machine
  - Simulates any other Turing Machine
- Universal TM simulates Turing Machine M
- Inputs are
  - Description of TM M and
  - Input String of M

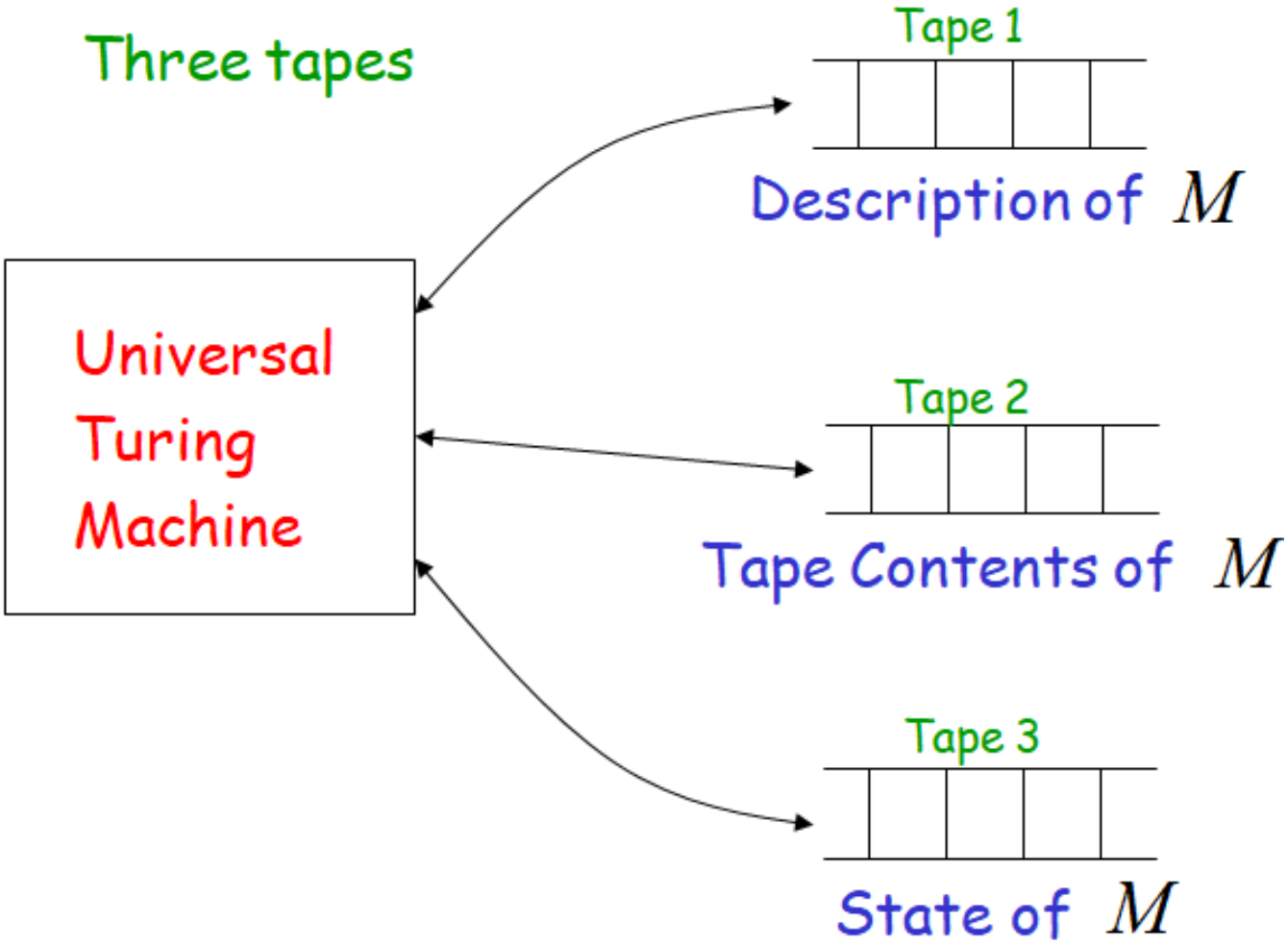
Designing the UTM

- Inputs are of the form:

  Code for M 111 w

- Note: A valid TM code never has 111, so we can split M from w.
- The UTM must accept its input if and only if M is a valid TM code and that TM accepts w.

Universal TM (UTM)



The UTM – (2)

- The UTM will have several tapes.
  - Tape 1 holds the input M111w
  - Tape 2 holds the tape of M.
    - Mark the current head position of M.
  - Tape 3 holds the state of M.

The UTM – (3)

- Step 1: The UTM checks that M is a valid code for a TM.
  - E.g., all moves have five components, no two moves have the same state/symbol as first two components.
- If M is not valid, its language is empty, so the UTM immediately halts without accepting
- Step 2: The UTM examines M to see how many of its own tape squares it needs to represent one symbol of M.

The UTM – (4)

- Step 3: Initialize Tape 2 to represent the tape of M with input w, and initialize Tape 3 to hold the start state
- Step 4: Simulate M.
  - Look for a move on Tape 1 that matches the state on Tape 3 and the tape symbol under the head on Tape 2.
  - If found, change the symbol and move the head marker on Tape 2 and change the State on Tape 3.
  - If M accepts, the UTM also accepts.

A Question

- Do we see anything like universal Turing machines in real life?

Proof That L_u is Recursively Enumerable, but not Recursive

- We designed a TM for L_u, so it is surely RE.
- Suppose it were recursive; that is, we could design a UTM U that always halted.
- Then we could also design an algorithm for L_d, as follows.

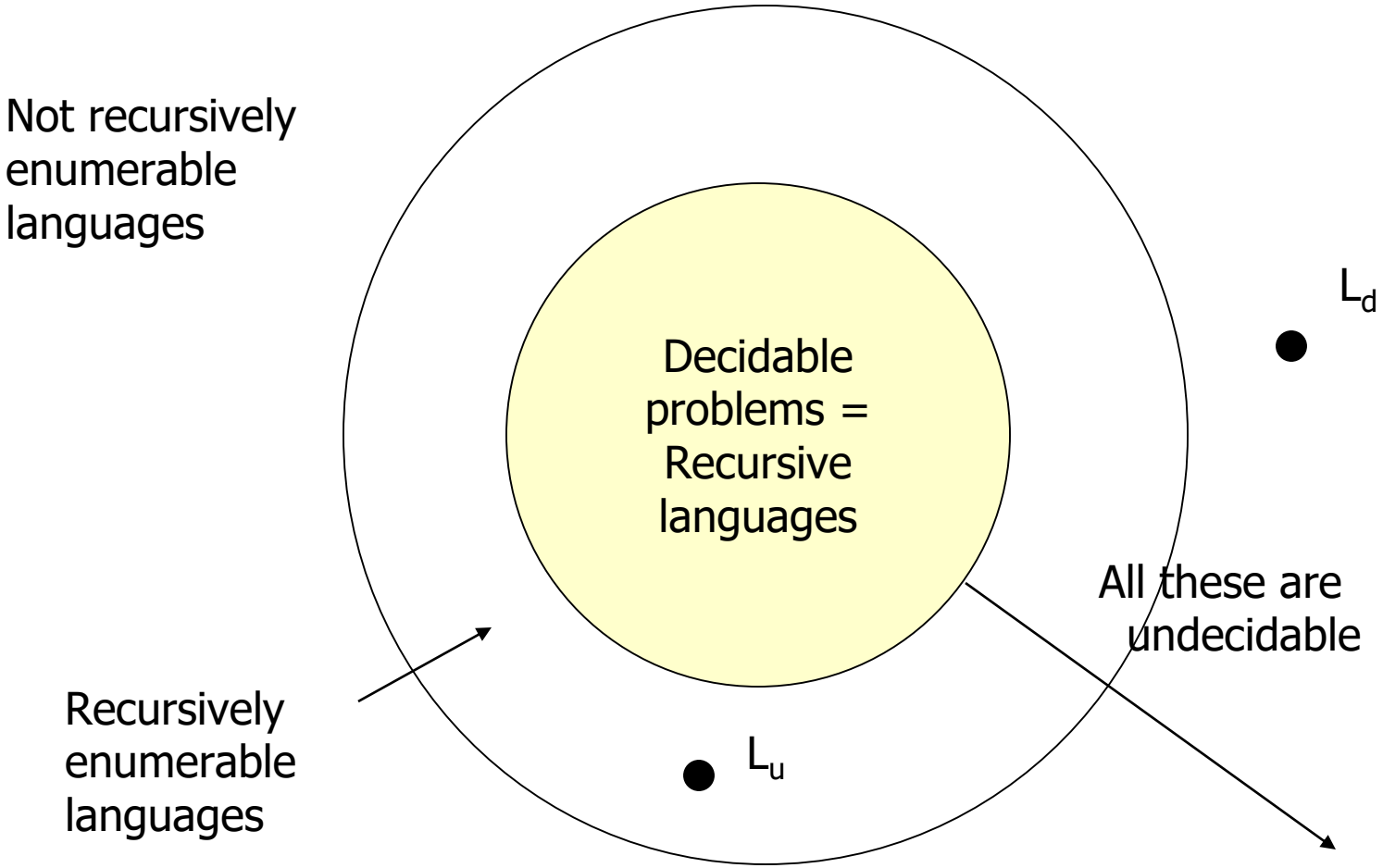
Proof – (2)

- Given input w, we can decide if it is in L_d by the following steps.
  1. Check that w is a valid TM code.
     - If not, then its language is empty, so w is in L_d.
  2. If valid, use the hypothetical algorithm to decide whether w111w is in L_u.
  3. If so, then w is not in L_d; else it is.

Proof – (3)

- But we already know there is no algorithm for L_d.
- Thus, our assumption that there was an algorithm for L_u is wrong.
- L_u is RE, but not recursive.

Bullseye Picture



References

John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Introduction to Automata Theory*, Languages, and Computation, Pearson, 3rd Edition, 2011.

Peter Linz, An Introduction to Formal Languages and Automata, Jones and Bartle Learning International, United Kingdom, 6th Edition, 2016.

Next Class: Unit IV

Reducibility and Rice's Theorem

Thank you.