DECIDABLE AND UNDECIDABLE PROBLEMS

Dr. A. Beulah

AP/CSE

LEARNING OBJECTIVE

- To Design Turing machines for any Languages (K3)
  - To Understand the concept of Decidable and Undecidable Problems

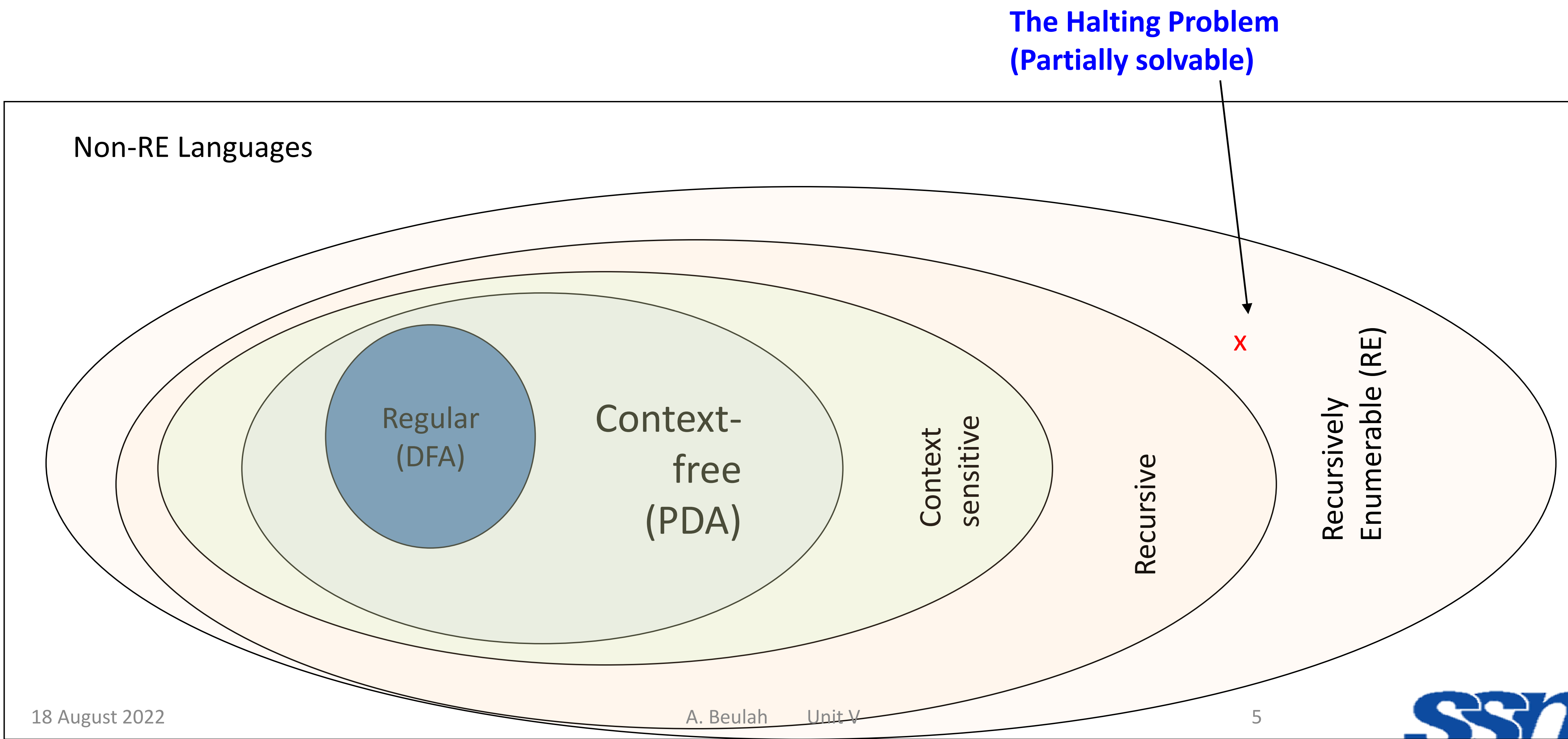
DECIDABLE PROBLEMS

- **Decidable problems about regular Languages**
  - Acceptance problem for DFAs
  - Acceptance problem for NFAs
  - Acceptance problem for Regular Expressions
  - Emptiness testing for DFAs
  - 2 DFAs recognizing the same language
- **Decidable problems about Context Free Languages**
  - Does a given CFG generate a given string?
  - Is the language of a given CFG empty?
  - Every CFL is decidable by a Turing machine

UNDECIDABLE PROBLEMS

- Halting Problem
- Post’s Correspondence problem
- Busy Beaver problem
- Whether the language accepted by a TM is empty
- Whether the language accepted by a TM is regular language
- Whether the language accepted by a TM is context free language

UNDECIDABLE PROBLEMS



THE HALTING PROBLEM

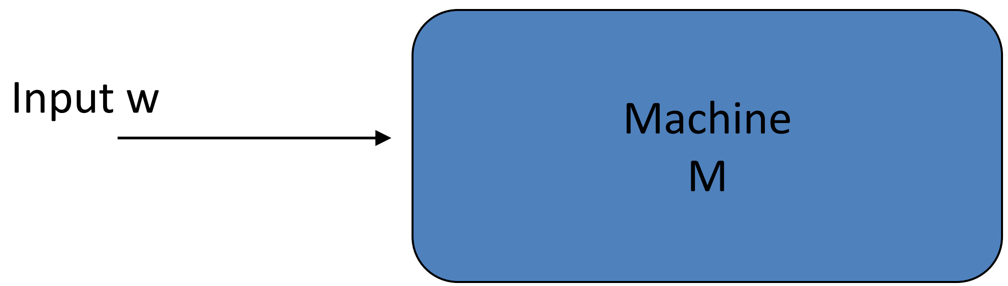
- An example of a recursive enumerable problem that is also undecidable

WHAT IS THE HALTING PROBLEM?

- Does a given Turing Machine M halt on a given input w?

or

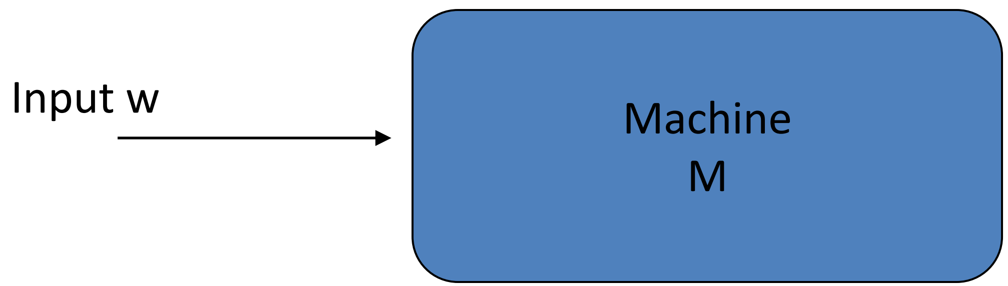
- Is it possible to tell whether a given machine will halt for some given input?



WHAT IS THE HALTING PROBLEM?

- Example: Given an arbitrary Turing machine M over alphabet Σ= { a , b } , and an arbitrary string w over Σ, does M halt when it is given w as an input ?

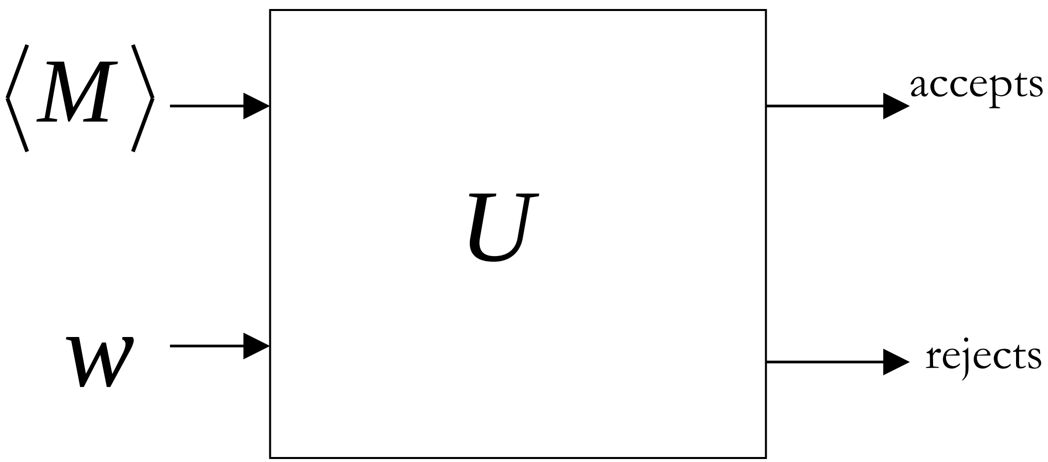
$$\mathrm{HALT}_{\mathrm{TM}} = \{ < M, w > | \text{ M is a TM which halts on } w \}$$



REVISIT UTM

Input string

$\langle M, w \rangle$



U accepts w

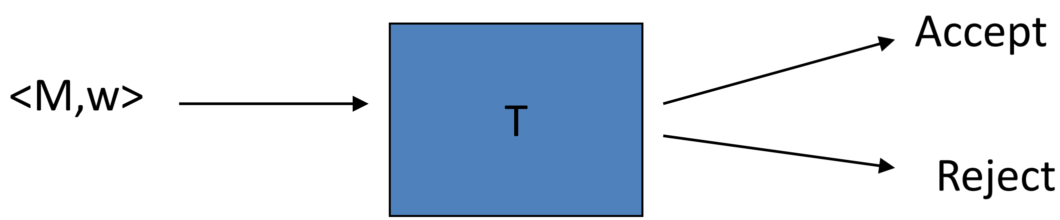
U rejects w

THEOREM: IS $HALT_{TM}$ DECIDABLE?

- Halting Problem is undecidable
- If there was such a Turing Machine
  - Its input will have two portions, M and w
  - It outputs either a YES or a NO depending on whether M halts on input w

PROOF BY CONTRADICTION

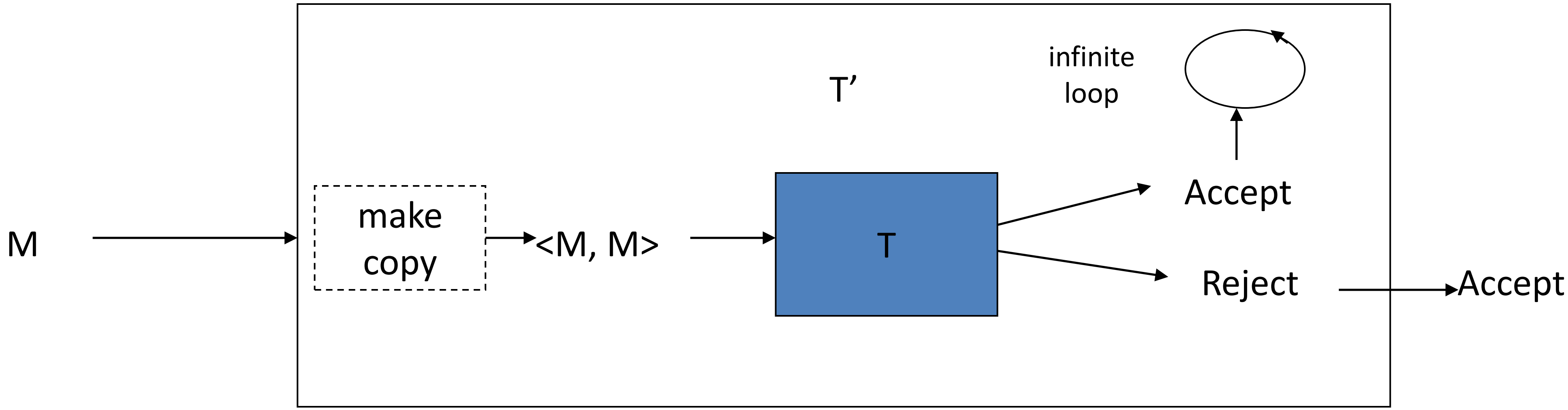
- Suppose Halting Problem is decidable
  - Plan: arrive at a contradiction
- If Halting Problem is decidable,
  - then there exists a TM T that decides Halting Problem



PROOF BY CONTRADICTION

- Create a TM T' based on T as follows:
  - T takes in a TM M
  - In T', M is duplicated so that there are now two portions on the input tape
  - Feed this new input into T
  - When it is about to print reject, print accept instead
  - When it is about to print accept, send the program to an infinite loop

PROOF BY CONTRADICTION

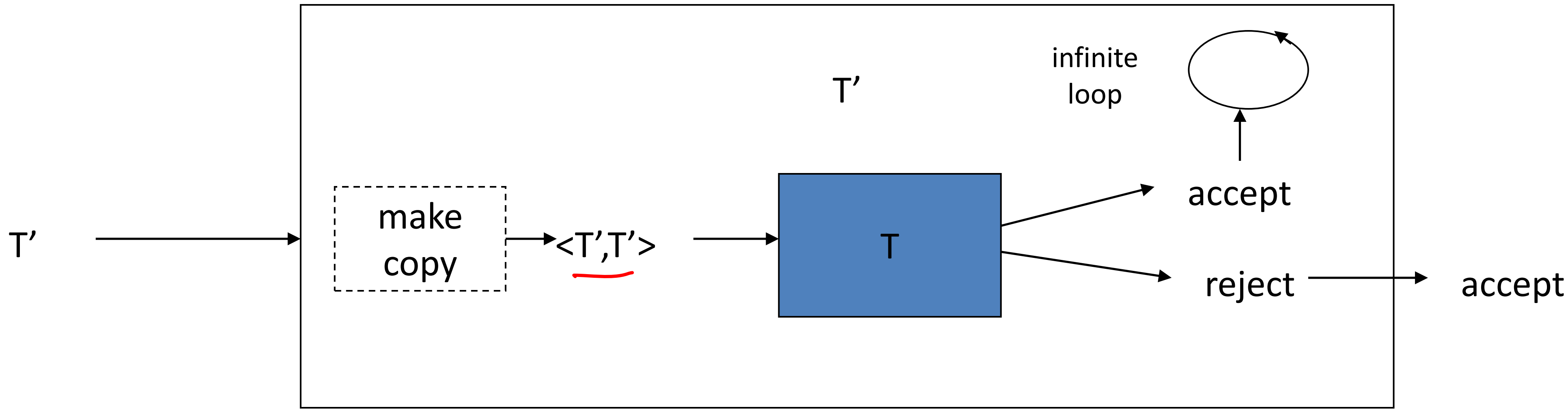


- Program T' takes a M as input, prints accept if M **does not halt** on input M, but goes into an infinite loop if M **halts** on input M

PROOF BY CONTRADICTION

- Consider feeding TM T’ to itself
- Consequence (two possibilities)
  - It prints accept
    - T’ halts on input T’
      if T’ does not halt on input T’ → a contradiction
  - It goes to an infinite loop
    - T’ does not halt on input T’
      if T’ halts on input T’ → a contradiction
- Therefore the supposition cannot hold, and Halting Problem is undecidable

PROOF BY CONTRADICTION

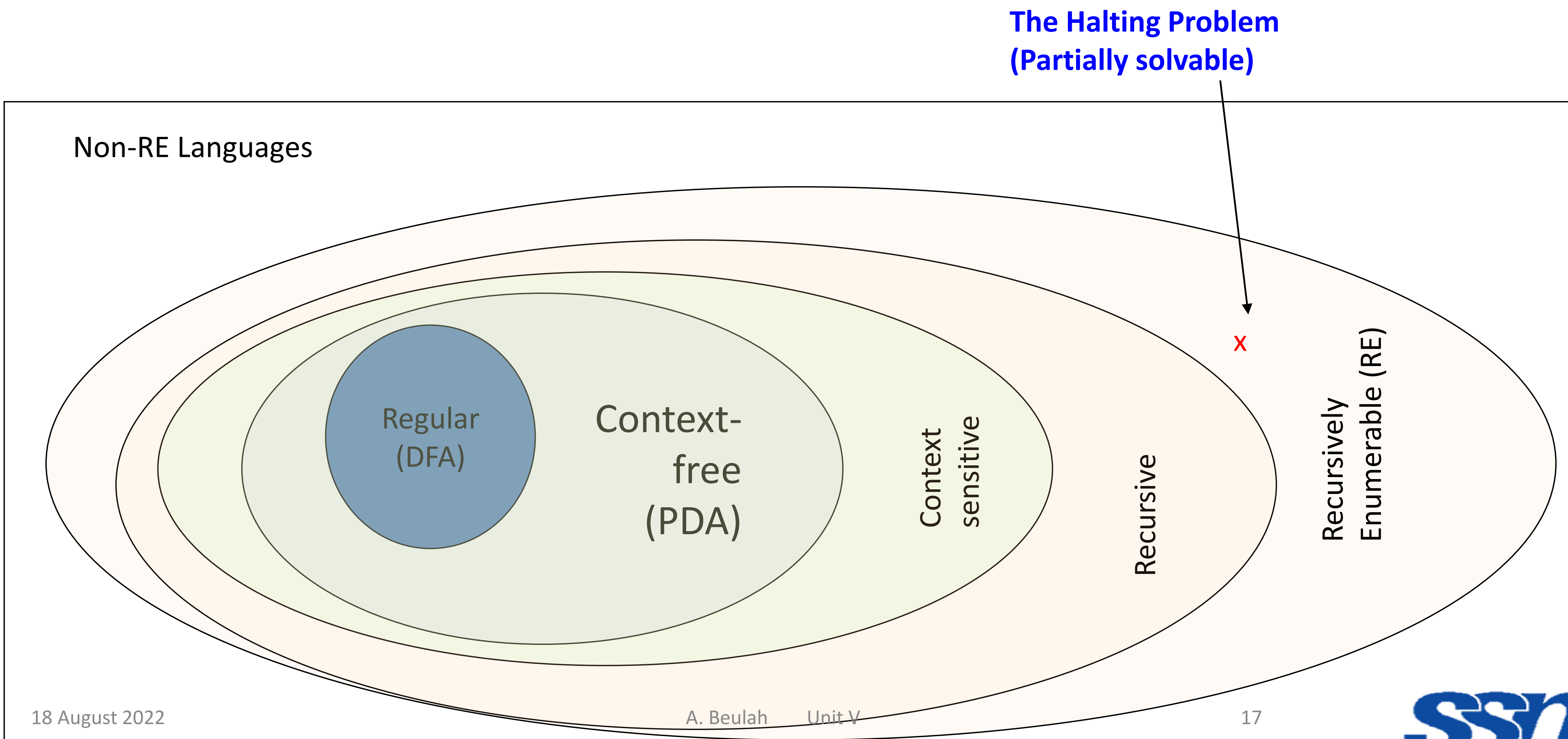


- T' halts on input T' (prints a accept, see outer box) if
- T' does not halt on input T' (T should yield a reject, see inner box)
- T' does not halt on input T' (infinite loop, see outer box) if
- T' halts on input T' (T should yield a accept, see inner box)

HP IS SEMIDECIDABLE

- There are problems such as HP that cannot be solved
- Actually, HP is semidecidable, that is if all we need is print accept when M on w halts, but not worry about printing reject if otherwise, a TM machine exists for the halting problem
  - Just simulate M on w, print accept (or go to a final state) when the simulation stops
  - This means that HP is not recursive but it is recursively enumerable

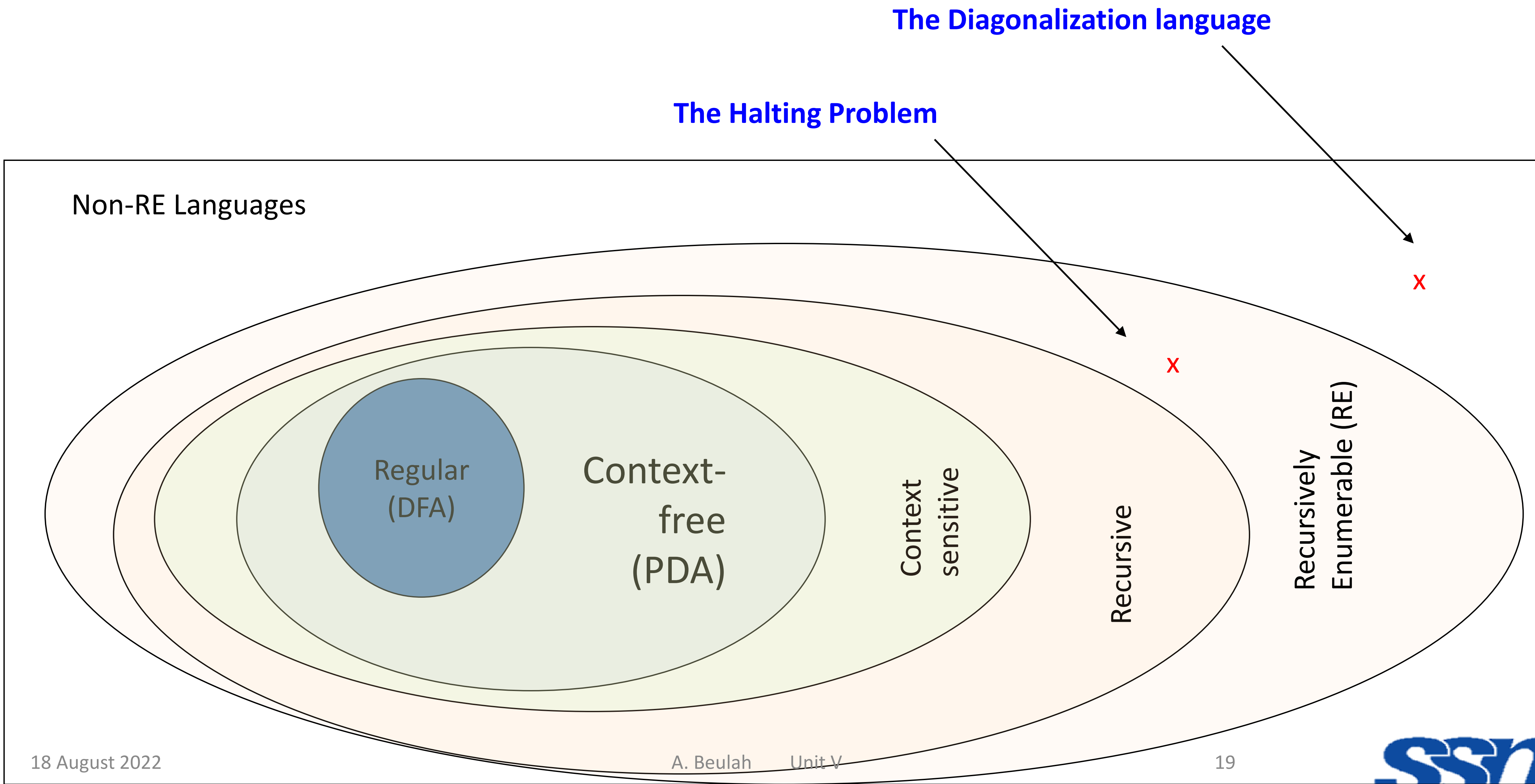
HP IS SEMIDECIDABLE



THE DIAGONALIZATION LANGUAGE

- **Example of a language that is <u>not recursive enumerable</u>**
- **(i.e, no TMs exist)**

THE DIAGONALIZATION LANGUAGE



A LANGUAGE ABOUT TMS & ACCEPTANCE

- Let L be the language of all strings <M,w> s.t.:
  1. M is a TM (coded in binary) with input alphabet also binary
  2. w is a binary string
  3. M accepts input w.

ANY TM M IN BINARY-CODED FORM

- $M = \{ Q, \{0,1\}, \Gamma, \delta, q_0, B, F \}$

  – Map all states, tape symbols and transitions to integers (→ binary strings)

  – $\delta(q_i, X_j) = (q_k, X_l, D_m)$ will be represented as:

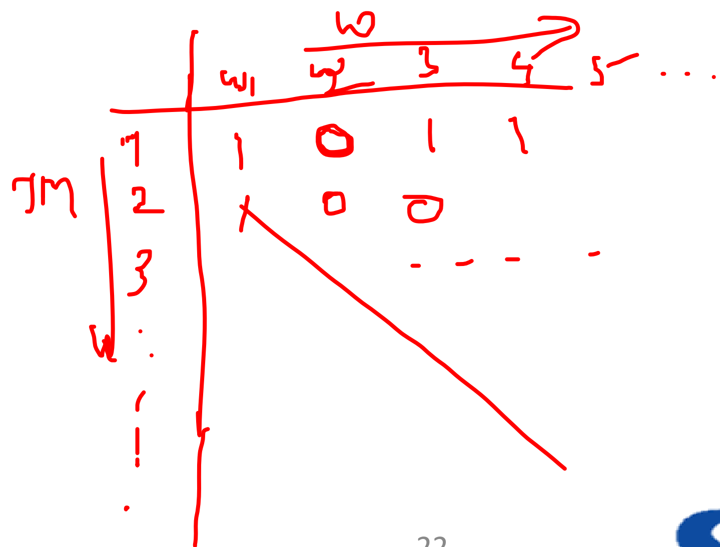
    → $0^i 1\ 0^j 1\ 0^k 1\ 0^l 1\ 0^m$

- Result: Each TM can be written down as a long binary string
- Canonical ordering of TMs:

  – $\{M_1, M_2, M_3, M_4, \ldots. M_i, \ldots \}$

ENUMERATING ALL BINARY STRINGS

- Let w be a binary string
- Then $1w \equiv i$, where i is some integer
  - E.g., If $w=\varepsilon$, then $i=1$;
  - If $w=0$, then $i=2$;
  - If $w=1$, then $i=3$; so on…
- If $1w \equiv i$, then call w as the i^{th} word or i^{th} binary string, denoted by w_i.
- ***A canonical ordering of all binary strings:***
  - ***$\{\varepsilon$, 0, 1, 00, 01, 10, 11, 000, 100, 101, 110, …..$\}$***
  - ***$\{w_1, w_2, w_3, w_4, \ldots.\ w_i, \ldots\}$***



THE DIAGONALIZATION LANGUAGE

- $L_d = \{ w_i \mid w_i \notin L(M_i) \}$
  - The language of all strings whose corresponding machine does *not* accept itself (i.e., its own code)

j → (input word w)

(TMs) i ↓

	1	2	3	4	…
1	0	1	0	1	…
2	1	1	0	0	…
3	0	1	0	1	…
4	1	0	0	1	…
⋮					⋱

diagonal

- Table: $T[i,j] = 1$, if M_i accepts w_j
  $= 0$, otherwise.

- Make a new language called
  $L_d = \{w_i \mid T[i,i] = 0\}$

L_D IS NOT RE (I.E., HAS NO TM)

<u>Proof (by contradiction):</u>

Let M be the TM for L_d

→ M has to be equal to some M_k s.t.

$$L(M_k) = L_d$$

→ Will w_k belong to $L(M_k)$ or not?

1. If $w_k \in L(M_k)$ ==> T[k,k]=1 ==> $w_k \notin L_d$
2. If $w_k \notin L(M_k)$ ==> T[k,k]=0 ==> $w_k \in L_d$

A contradiction either way!!

$L_d \rightarrow M \rightarrow M_k \rightarrow L(M_k)$

$\therefore L_d = L(M_k)$ ✗

w_k

POST'S CORRESPONDENCE PROBLEM

- Emil Post

DEFINITION

Given two lists A and B:

$A = w_1, w_2, \ldots, w_k \qquad B = x_1, x_2, \ldots, x_k$

The problem is to determine if there is a sequence of one or more integers $i_1, i_2, \ldots, i_m$ such that:

$$w_{i_1} w_{i_2} \ldots w_{i_m} = x_{i_1} x_{i_2} \ldots x_{i_m}$$

(w_i, x_i) is called a corresponding pair.

Indices may be repeated or omitted

EXAMPLE

	w_1	w_2	w_3
A:	100	11	111

	x_1	x_2	x_3
B:	001	111	11

PC-solution: 2,1,3 $\quad w_2 w_1 w_3 = x_2 x_1 x_3$

$(w_2, w_1, w_3, x_2, x_1, x_3)$

2, 1, 3

$$\underset{11\;100\;111}{w_2\;\; w_1\;\; w_3} = \underset{111\;001\;11}{x_2\;\; x_1\;\; x_3}$$

11100111

EXAMPLE

	w_1	w_2	w_3
$A:$	00	001	1000
	x_1	x_2	x_3
$B:$	0	11	011

- There is no solution
- Because total length of strings from B is smaller than total length of strings from A

MODIFIED POST CORRESPONDENCE PROBLEM (MPCP)

Given two lists A and B:

$A = w_1, w_2, \ldots, w_k \qquad B = x_1, x_2, \ldots, x_k$

The problem is to determine if there is a sequence of one or more integers $i_1, i_2, \ldots, i_m$ such that:

$$w_1 w_{i_1} w_{i_2} \ldots w_{i_m} = x_1 x_{i_1} x_{i_2} \ldots x_{i_m}$$

(w_i, x_i) is called a corresponding pair.

- Pair (w_1, x_1) is forced to be at the beginning of the two strings.

EXAMPLE

	A	B
i	w_i	x_i
1	11	1
2	1	111
3	0111	10
4	10	0

This MPCP instance has a solution: 3, 2, 2, 4:

$$w_1 w_3 w_2 w_2 w_4 = x_1 x_3 x_2 x_2 x_4 = 1101111110$$

EXAMPLE

	A	B
i	w_i	x_i
1	10	101
2	011	11
3	101	011

Does this MPCP instance have a solution?

MAPPING L_U TO MPCP

<M,w> → UTM → A / R

I/P O/P

L_U instance

Given:

(M,w)

Construct an MPCP instance →

MPCP instance

Two lists:

A and B

If M accepts w, the two lists can be matched.
Otherwise, the two lists cannot be matched.

MAPPING L_U TO MPCP

- Given M and w, there are five types of strings in list A and B:
- **Starting string (first pair):**

List A	**List B**
#	$\#q_0w\#$

where q_0 is the starting state of M.

$L_U = \{ \langle M, w \rangle / \}$

MAPPING L_U TO MPCP

$\Gamma = \{ \_\_ \}$

- **Strings from the transition function δ:**

List A	List B	
qX	Yp	from δ(q,X)=(p,Y,R)
ZqX	pZY	from δ(q,X)=(p,Y,L)
q#	Yp#	from δ(q,#)=(p,Y,R)
Zq#	pZY#	from δ(q,#)=(p,Y,L)

where Z is any tape symbol except the blank.

MAPPING L_U TO MPCP

- **Strings for copying:**

List A	**List B**
X	X

where X is any tape symbol (including the blank).

MAPPING L_U TO MPCP

- **Strings for consuming the tape symbols at the end:**

List A	List B
Xq	q
qY	q
XqY	q

where q is an accepting state, and each X and Y is any tape symbol except the blank.

MAPPING L_U TO MPCP

- **Ending string:**

List A	**List B**
q##	#

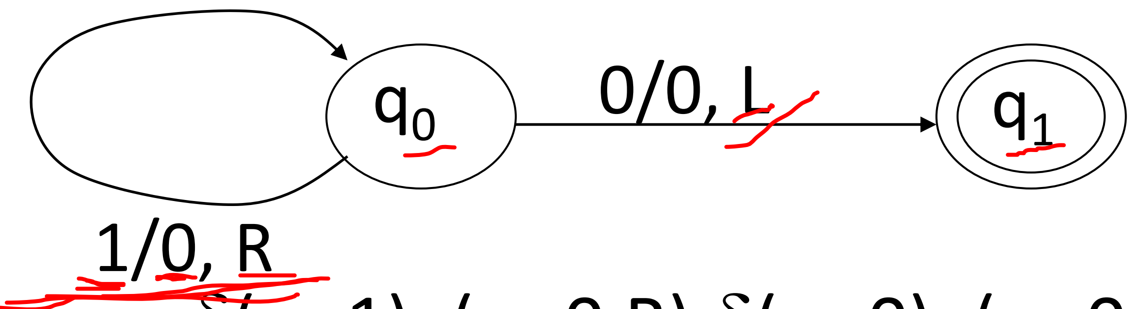
  where q is an accepting state.

- Using this mapping, we can prove that the original L_U instance has a solution if and only if the mapped MPCP instance has a solution.

EXAMPLE

- Consider the following Turing machine:

$$M = (\{q_0, q_1\},\{0,1\},\{0,1,\#\}, \delta, q_0, \#, \{q_1\})$$



$$\delta(q_0,1)=(q_0,0,R) \quad \delta(q_0,0)=(q_1,0,L)$$

- Consider input w=110.

$q_0 110 \vdash 0 q_0 10 \vdash 00 q_0 0 \vdash 0 q_1 00$

EXAMPLE

- Now we will construct an MPCP instance from M and w. There are five types of strings in list A and B:

A B

$\#$ $\# q_0 w \#$

- **Starting string (first pair):**

List A	**List B**
#	$\#q_0110\#$

EXAMPLE

- **Strings from the transition function δ:**

List A	**List B**	
$q_0 1$	$0 q_0$	(from $\delta(q_0,1)=(q_0,0,R)$)
$z=0$ $0 q_0 0$	$q_1 00$	(from $\delta(q_0,0)=(q_1,0,L)$)
$z=1$ $1 q_0 0$	$q_1 10$	(from $\delta(q_0,0)=(q_1,0,L)$)

$\Gamma = \{ \ldots \# \}$

$\delta(q,x) = (p,y,R)$

$qx \quad yp$

$\delta(q,x) = (p,y,L)$

$zqx \quad pzy$

EXAMPLE

- **Strings for copying:**

$\Gamma = \{\#, 0, 1\}$

List A	**List B**
#	#
0	0
1	1

EXAMPLE

$\Gamma = \{0, 1, \#\}$

- **Strings for consuming the tape symbols at the end:**

	List A	**List B**	**List A**	**List B**
$x=0$	$0q_1$	q_1	$0q_11$	q_1
$x=1$	$1q_1$	q_1	$1q_10$	q_1
$y=0$	q_10	q_1	$0q_10$	q_1
$y=1$	q_11	q_1	$1q_10$	q_1

EXAMPLE

- **Ending string:**

List A	**List B**
$q_1\#\#$	$\#$

Now, we have constructed an MPCP instance.

EXAMPLE

	List A	List B
1.	#	$\#q_0110\#$
2.	q_01	$0q_0$
3.	$0q_00$	q_100
4.	$1q_00$	q_110
5.	#	#
6.	0	0
7.	1	1
8.	$q_1\#\#$	#

	List A	List B
9.	$0q_1$	q_1
10.	$1q_1$	q_1
11.	q_10	q_1
12.	q_11	q_1
13.	$0q_11$	q_1
14.	$1q_10$	q_1
15.	$0q_10$	q_1
16.	$1q_10$	q_1

EXAMPLE OF ULP TO MPCP

M w

w

RY M

- This ULP instance has a solution:

$$q_0110 \xrightarrow{\#} 0q_010 \rightarrow 00q_00 \rightarrow 0q_100 \text{ (halt)}$$

- Does this MPCP instance has a solution?

List A:

$\# \; q_0 \; 1 \; 1 \; 0 \; \# \; 0 \; q_0 \; 1 \; 0 \; \# \; 0 \; 0 \; q_0 \; 0 \; \# \; 0 \; q_1 \; 0 \; 0 \; \# \; q_1 \; 0 \; \# \; q_1 \; \# \; \#$

List B:

$\# \; q_0 \; 1 \; 1 \; 0 \; \# \; 0 \; q_0 \; 1 \; 0 \; \# \; 0 \; 0 \; q_0 \; 0 \; \# \; 0 \; q_1 \; 0 \; 0 \; \# \; q_1 \; 0 \; \# \; q_1 \; \# \; \#$

The solution is the sequence of indices:

2, 7, 6, 5, 6, 2, 6, 5, 6, 3, 5, 15, 6, 5, 11, 5, 8

CLASS DISCUSSION

Consider the input w = 101. Construct the corresponding MPCP instance I and show that M will accept w by giving a solution to I.

CLASS DISCUSSION (CONT'D)

	List A	List B
1.	#	#$q_0$101#
2.	$q_0$1	0q_0
3.	0$q_0$0	$q_1$00
4.	1$q_0$0	$q_1$10
5.	#	#
6.	0	0
7.	1	1
8.	q_1##	#

	List A	List B
9.	0q_1	q_1
10.	1q_1	q_1
11.	$q_1$0	q_1
12.	$q_1$1	q_1
13.	0$q_1$1	q_1
14.	1$q_1$0	q_1
15.	0$q_1$0	q_1
16.	1$q_1$0	q_1

TEST YOUR KNOWLEDGE

- Which of the following problems is undecidable?
  - Deciding if a given context-free grammar is ambiguous.
  - Deciding if a given string is generated by a given context-free grammar.
  - Deciding if the language generated by a given context-free grammar is empty.
  - Deciding if the language generated by a given context-free grammar is finite.

SUMMARY

- What is undecidability
- Recursive and Recursive enumerable languages

REFERENCE

- Hopcroft J.E., Motwani R. and Ullman J.D, “Introduction to Automata Theory, Languages and Computations”, Second Edition, Pearson Education, 2008