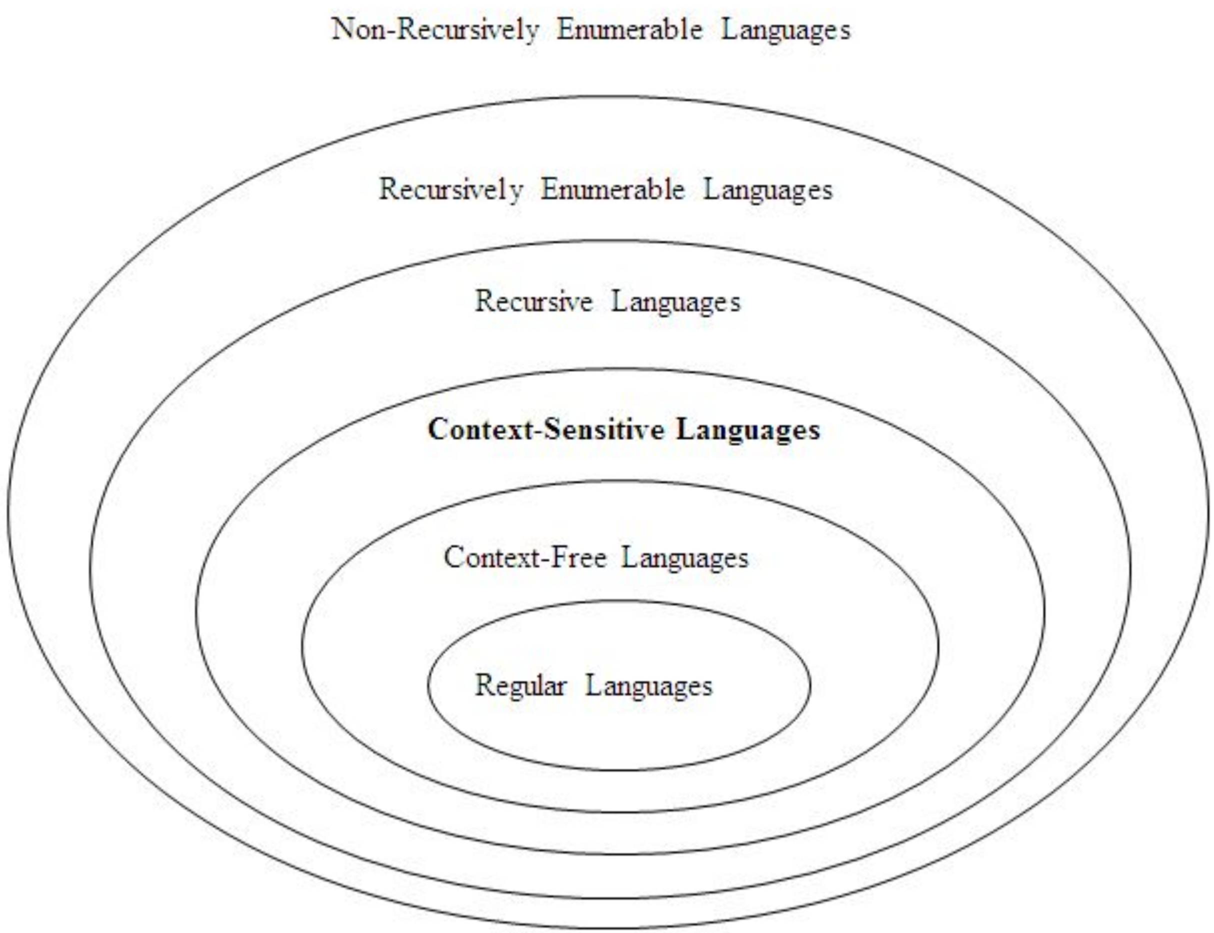
Turing Machines

Recursively Enumerable and Recursive Languages

- **The Hierarchy of Languages:**



- Recursively enumerable languages are also known as *type 0* languages.

- Context-sensitive languages are also known as *type 1* languages.

- Context-free languages are also known as *type 2* languages.

- Regular languages are also known as *type 3* languages.

- TMs model the computing capability of a general purpose computer, which informally can be described as:
  - Effective procedure
    - Finitely describable
    - Well defined, discrete, “mechanical” steps
    - Always terminates
  - Computable function
    - A function computable by an effective procedure

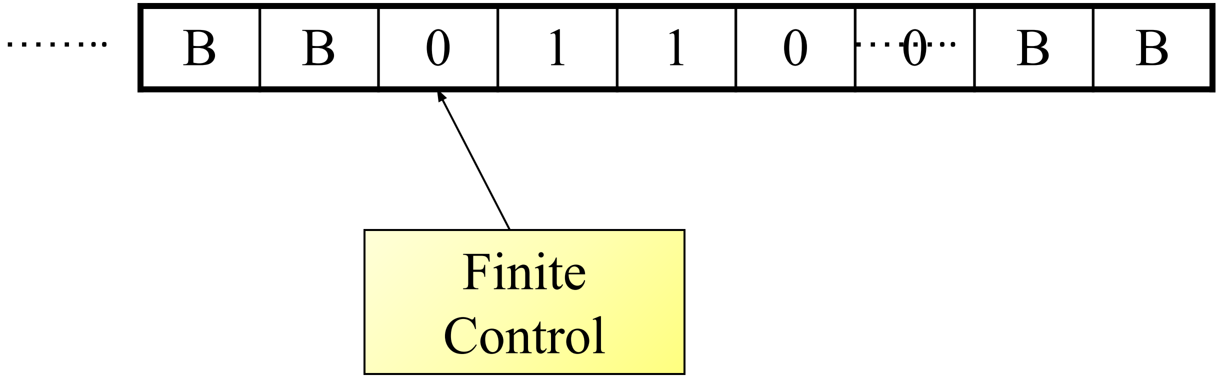
- TMs model the computing capability of a general purpose computer, which informally can be described as:
  - Effective procedure
    - Finitely describable
    - Well defined, discrete, “mechanical” steps
    - Always terminates
  - Computable function
    - A function computable by an effective procedure

- TMs formalize the above notion.

- **Church-Turing Thesis:** There is an effective procedure for solving a problem if and only if there is a TM that halts for all inputs and solves the problem.
  - There are many other computing models, but all are equivalent to or subsumed by TMs. *There is no more powerful machine* (Technically cannot be proved).

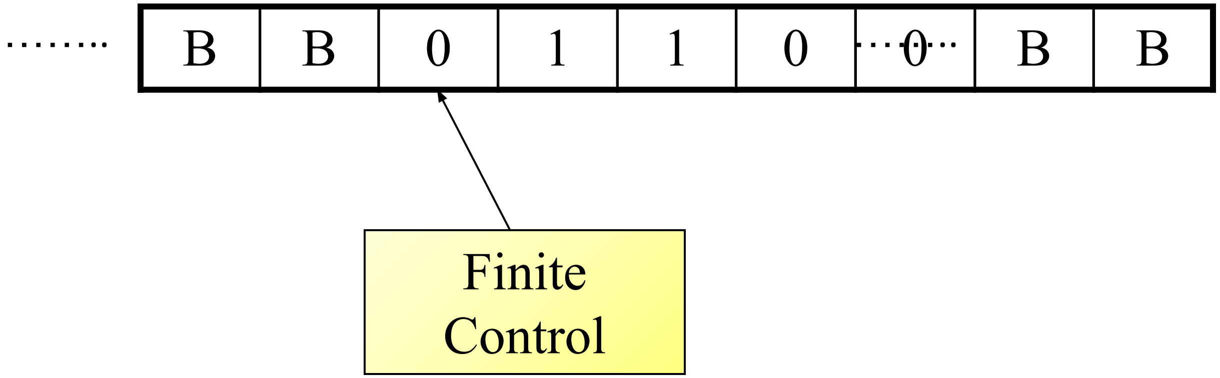
- DFAs and PDAs do not model all effective procedures or computable functions, but only a subset.

Deterministic Turing Machine (DTM)



- Two-way, infinite tape, broken into cells, each containing one symbol.
- Two-way, read/write tape head.
- An input string is placed on the tape, padded to the left and right infinitely with blanks, read/write head is positioned at the left most input character.
- Finite control (read/write head is part of this control), knows current symbol being scanned, and its current state.

Deterministic Turing Machine (DTM)



- In one move, depending on the current state and the current symbol being scanned, the TM does: (1) changes state, (2) prints a symbol over the cell being scanned, and (3) moves its' tape head one cell left or right.
- Many modifications possible, but Church-Turing declares equivalence of all.

Formal Definition of a DTM

- **A DTM is a seven-tuple:**

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Q A finite set of states
Σ A finite input alphabet, which is a subset of $\Gamma - \{B\}$
Γ A finite tape alphabet, which is a strict superset of Σ
B A distinguished blank symbol, which is in Γ
q_0 The initial/starting state, q_0 is in Q
FA set of final/accepting states, which is a subset of Q
δ A next-move function, which is a *mapping* (i.e., may be undefined)
$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$

Intuitively, $\delta(q,s)$ specifies the next state, symbol to be written, and the direction of tape head movement by M after reading symbol s while in state q.

- **Example #1:** {w | w is in {0,1}* and w ends with a 0}

  = {0, 00, 10, 10110, …}

  Note: ε is not in the language

- **Example #1:** {w | w is in {0,1}* and w ends with a 0}

  = {0, 00, 10, 10110, …}

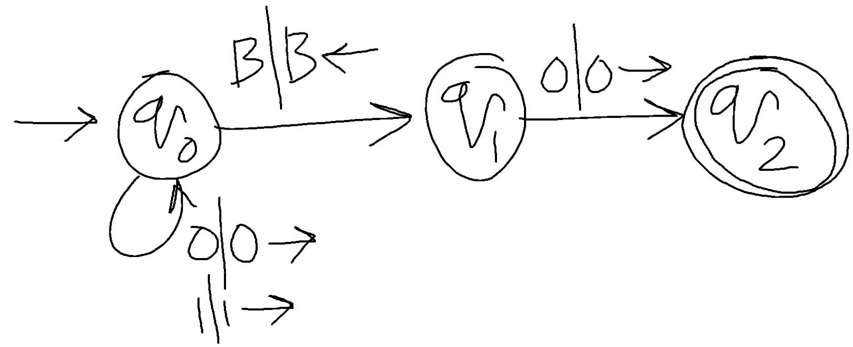
  Note: ε is not in the language

Q = {q_0, q_1, q_2}
Γ = {0, 1, B}
Σ = {0, 1}
F = {q_2}
δ:

	0	1	B
->q_0	(q_0, 0, R)	(q_0, 1, R)	(q_1, B, L)
q_1	(q_2, 0, R)	-	-
q_2^*	-	-	-

- q_0 is the start state and the "scan right" state, until hits B
- q_1 state is to begin the verification – last character is 0 or not
- q_2 is the final state

- **Example #1:** {w | w is in {0,1}* and w ends with a 0}



$Q = \{q_0, q_1, q_2\}$
$\Gamma = \{0, 1, B\}$
$\Sigma = \{0, 1\}$
$F = \{q_2\}$
δ:

	0	1	B
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, B, L)
q_1	$(q_2, 0, R)$	-	-
q_2^*	-	-	-

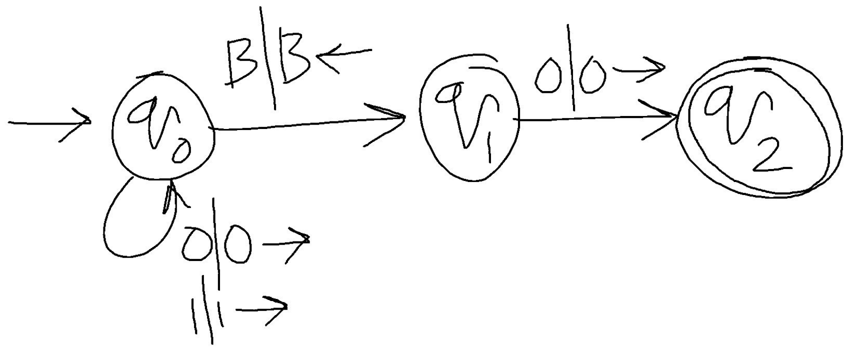
- **q_0 is the start state and the “scan right” state, until hits B**
- **q_1 state is to begin the verification – last character is 0 or not**
- **q_2 is the final state**

ID

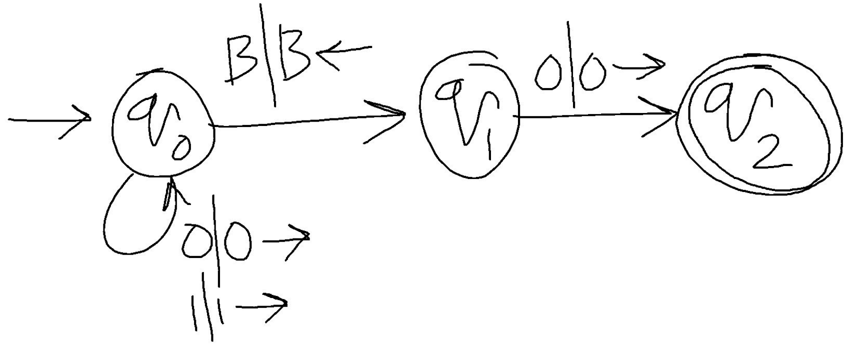
- To describe the steps we use IDs.
- An ID is $\alpha q \beta$
- $\alpha, \beta \in \Gamma^*$
- $\alpha\beta$ is on the tape. Left and right of this are blanks.
- Head is on the first character of β
- The TM is in state q

Step

- $id_1 \vdash id_2$ describes one step
- $\vdash^*$ is reflexive and transitive closure of $\vdash$



- For input 1010, computation steps…



- For input 1010, computation steps…

$$q_0 1010 \vdash 1 q_0 010 \vdash 10 q_0 10$$
$$\vdash 101 q_0 0 \vdash 1010 q_0 B$$
$$\vdash 101 q_1 0 \vdash 1010 q_2$$

Have you noticed

- Recognition is immediate.
  - TM, once hits one of the final states, it accepts and halts.
- No need for the input to be exhausted !!

- How TM rejects?
  - It gets stuck.
  - It goes on infinitely without ever hitting a final state.

- $\{0^n 1^n | n \geq 1\}$
- Idea??

- $\{0^n 1^n | n \geq 1\}$
- Idea??

- $\{0^n 1^n | n \geq 1\}$
- Idea??

B	B	0	0	0	1	1	1	B
		$\uparrow$						

q_0

- $\{0^n 1^n | n \geq 1\}$
- Idea??

B	B	0	0	0	1	1	1	B
		$\uparrow$ q_0						

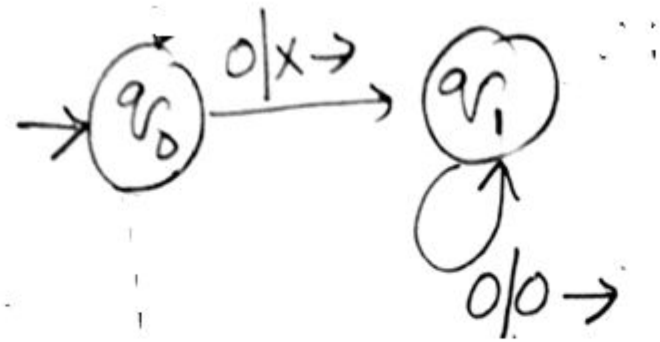
B	B	X	0	0	1	1	1	B
			$\uparrow$ q_1					

- We turn

0 in to X and

probe for 1.

We turn 1 to Y.



B	B	0	0	0	1	1	1	B
		↑						

q_0

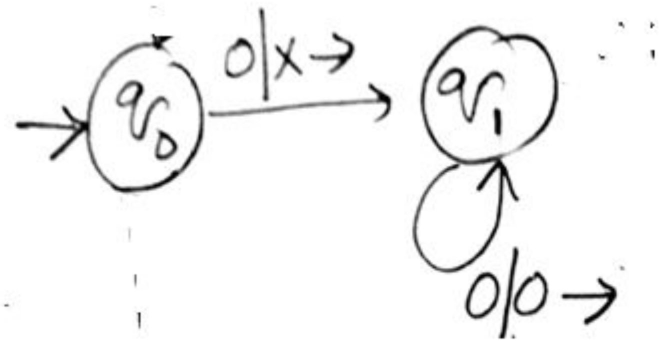
B	B	X	0	0	1	1	1	B
			↑					

q_1

⋮

B	B	X	0	0	1	1	1	B
					↑			

q_1



B	B	0	0	0	1	1	1	B
		↑						

q_0

B	B	X	0	0	1	1	1	B
			↑					

q_1

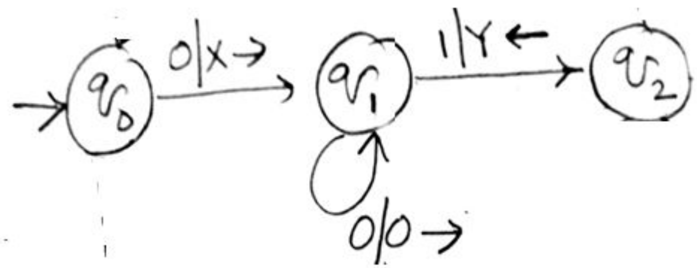
⋮

B	B	X	0	0	1	1	1	B
					↑			

q_1

B	B	X	0	0	Y	1	1	B
				↑				

q_2



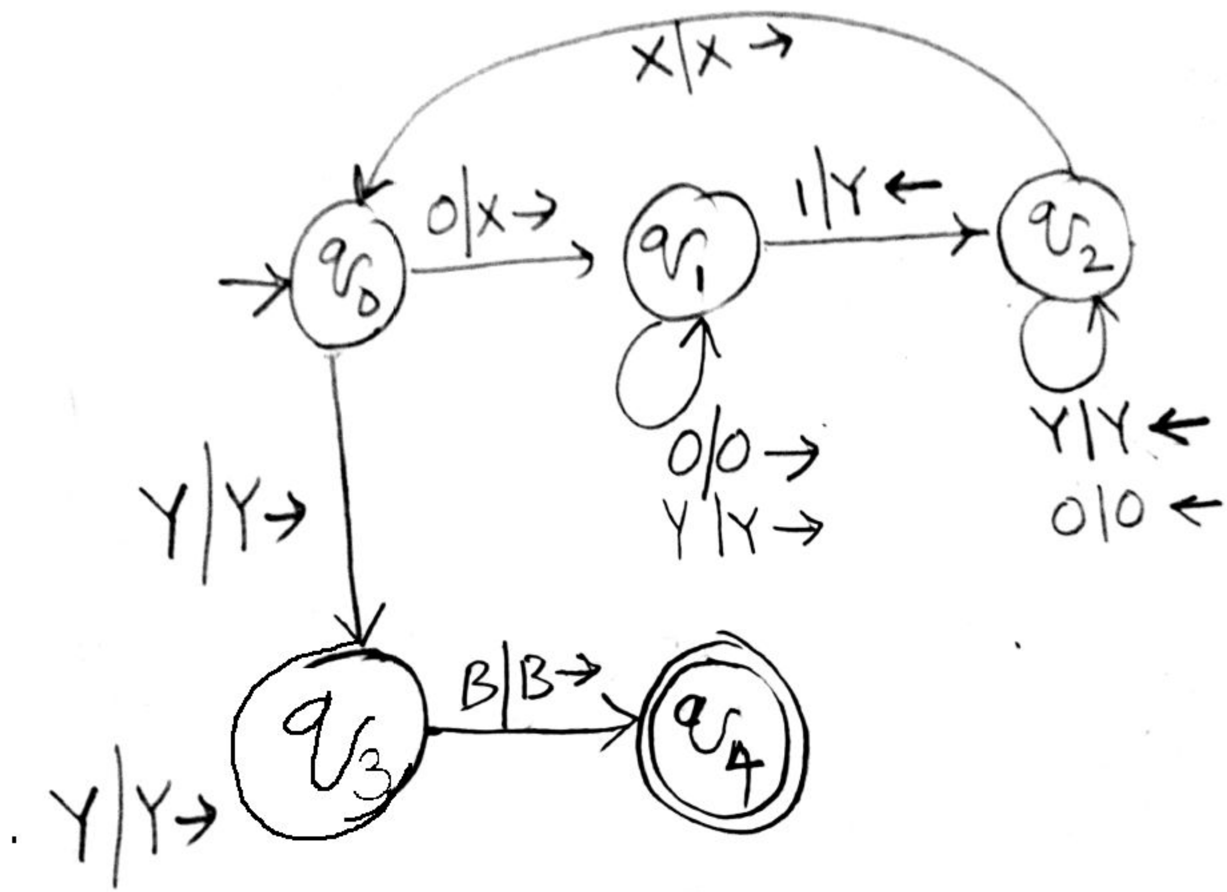
B	B	0	0	0	1	1	1	B
		↑ q_0						

B	B	X	0	0	1	1	1	B
			↑ q_1					

⋮

B	B	X	0	0	1	1	1	B
					↑ q_1			

B	B	X	0	0	Y	1	1	B
				↑ q_2				

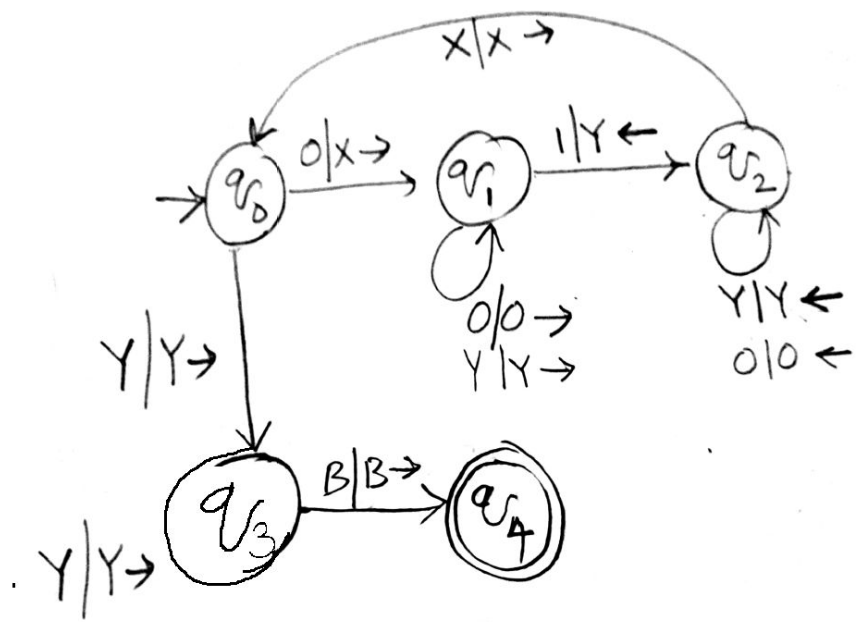

X|X →
0|X →
1|Y ←
q0
q1
q2
0|0 →
Y|Y →
Y|Y ←
0|0 ←
Y|Y →
B|B →
q3
q4
Y|Y →

- **Example #2:** $\{0^n1^n \mid n \geq 1\}$

	0	1	X	Y	B
$\to q_0$	(q_1, X, R)	-	-	(q_3, Y, R) ***0's finished***	-
q_1	$(q_1, 0, R)$ ***ignore1***	(q_2, Y, L)	-	(q_1, Y, R) ***ignore2***	- [more 0's]
q_2	$(q_2, 0, L)$ ***ignore2***	-	(q_0, X, R)	(q_2, Y, L) ***ignore1***	-
q_3	-	- [more 1's]	-	(q_3, Y, R) ***ignore***	(q_4, B, R)
q_4*	-	-	-	-	-

- **Sample Computation:** (on input: 0011),

$q_0 0011BB..$ $\vdash Xq_1011$
$\vdash X0q_111$
$\vdash Xq_20Y1$
$\vdash q_2X0Y1$
$\vdash Xq_00Y1$
$\vdash XXq_1Y1$
$\vdash XXYq_11$
$\vdash XXq_2YY$
$\vdash Xq_2XYY$
$\vdash XXq_0YY$
$\vdash XXYq_3Y\,B\ldots$
$\vdash XXYYq_3\,BB\ldots$
$\vdash XXYYBq_4$



TM

- TM can be seen as a recognizer.
- TM can be seen as a input-output mapper.
- TM can also be seen as an enumerator.
  - Generates the language one string after the other.

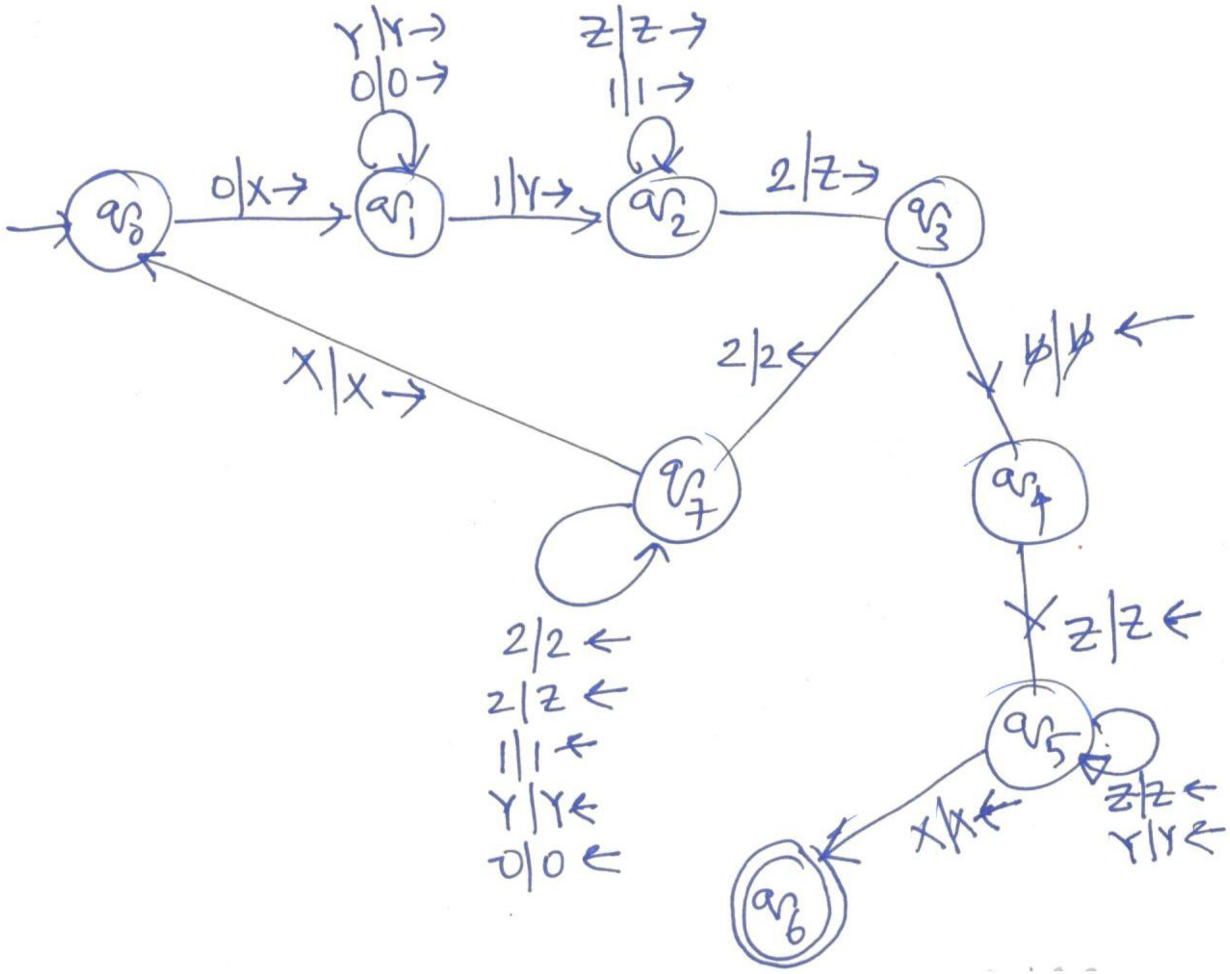
Eg: $\Sigma = \{0,1,2\}$, $\Gamma = \{X, Y, Z, B\}$ $L = \{0^n 1^n 2^n \mid n \geq 1\}$

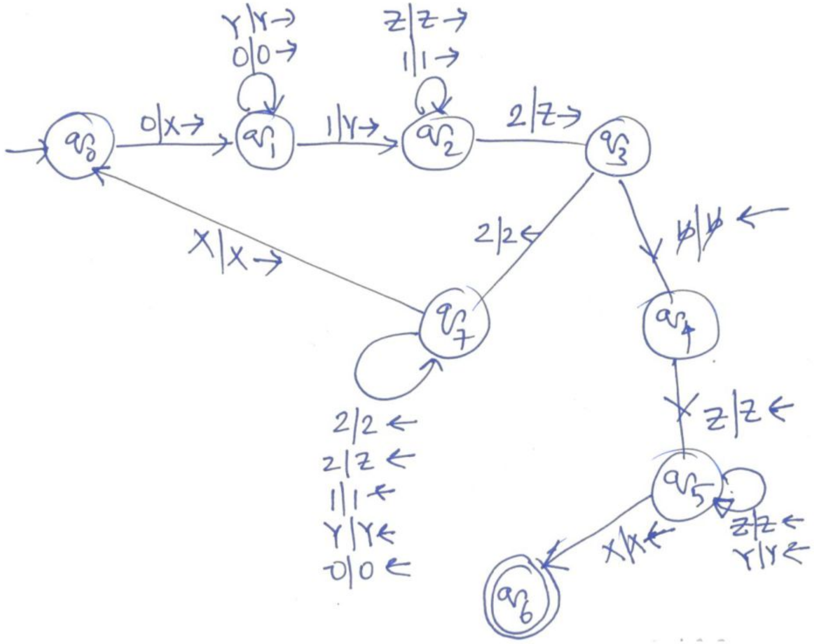
- But, there is a mistake in this.

Eg: $\Sigma = \{0,1,2\}$, $\Gamma = \{x, y, z, b\}$ $L = \{0^n 1^n 2^n \mid n \geq 1\}$

- But, there is a mistake in this.
- Γ has to be a superset of Σ

Eg: $\Sigma = \{0,1,2\}$, $\Gamma = \{X, Y, Z, b\}$ $L = \{0^n 1^n 2^n \mid n \geq 1\}$





$w = 001122$

$q_0\,001122 \vdash X\,q_1\,01122 \vdash X0\,q_1\,1122 \vdash X0Y\,q_2\,122$

$\vdash X0Y1\,q_2\,22 \vdash X0Y1Z\,q_3\,2 \vdash X0Y1\,q_7\,Z2$

$\vdash^* q_7\,X0Y1Z2 \vdash X\,q_0\,0Y1Z2 \vdash^* XXYYZZ\,q_3\,\not{b}$

$\vdash^* X\,q_5\,XYYZZ \vdash q_6\,XXYYZZ$

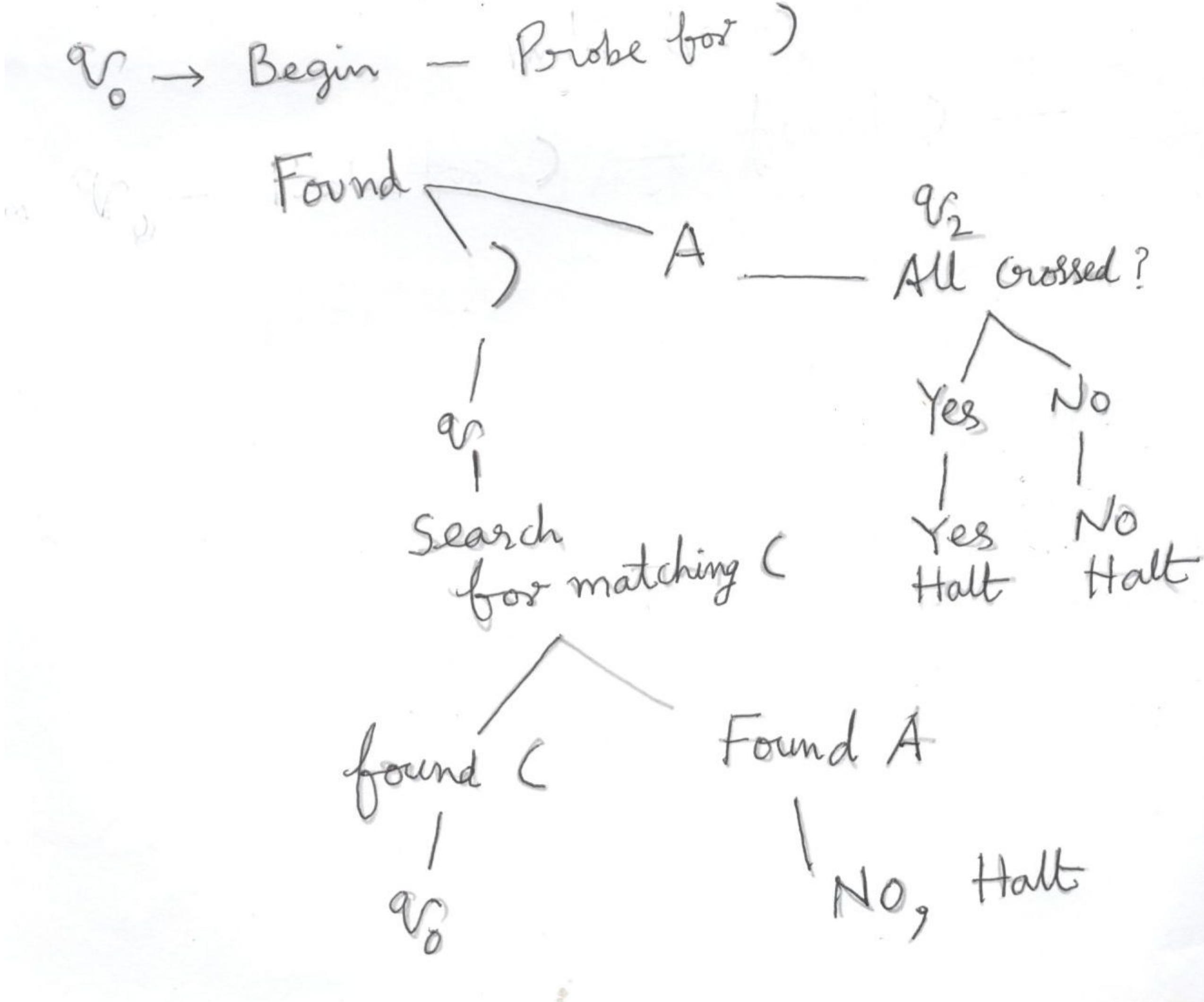
TM that gives output

- Example:
- $\Sigma = \{(,)\}$
- L is set of well formed parentheses.
- $\Gamma = \{A, X, B, Y, N\}$
- We use B or $\not{b}$ to mean the blank
- To simplify, we assume the given string is embedded between As

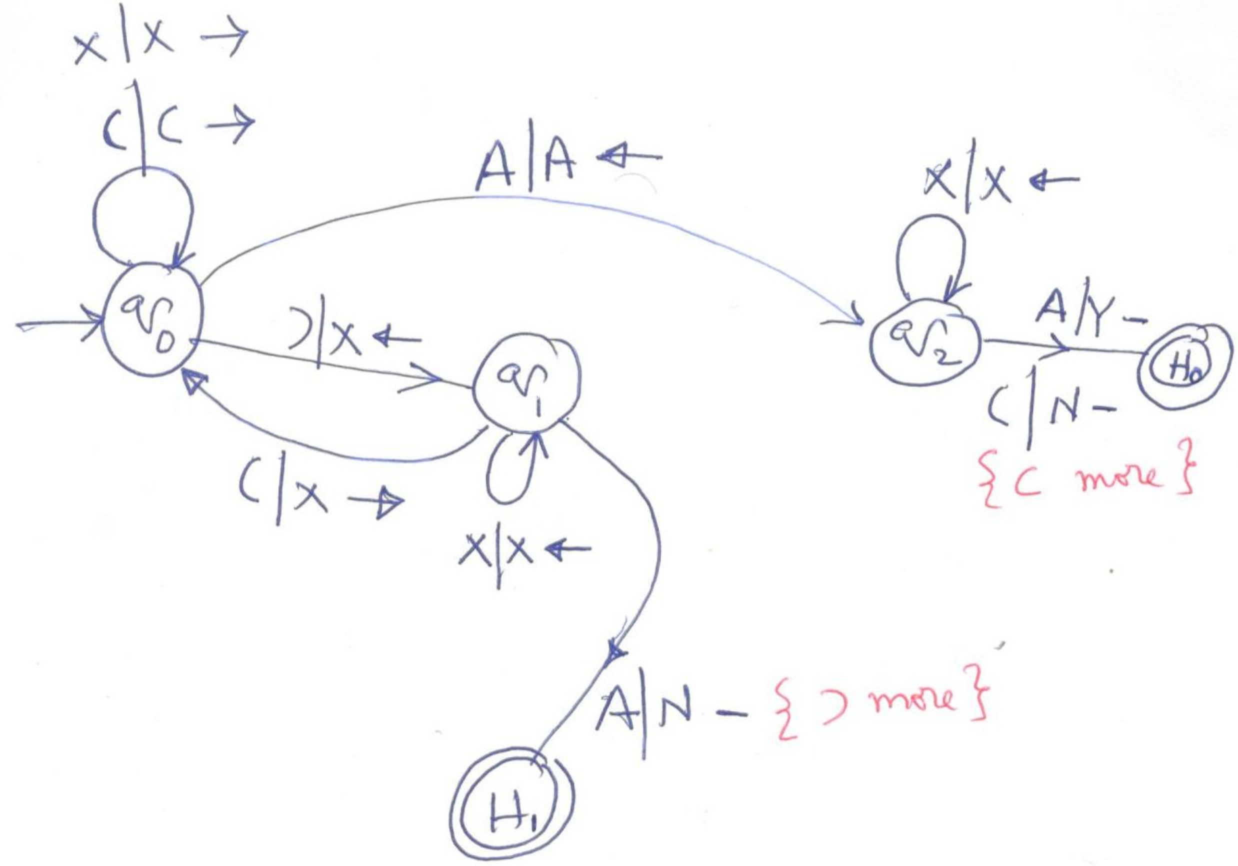
.	.	B	B	A	(	)	(	)	A	B	B	.	.

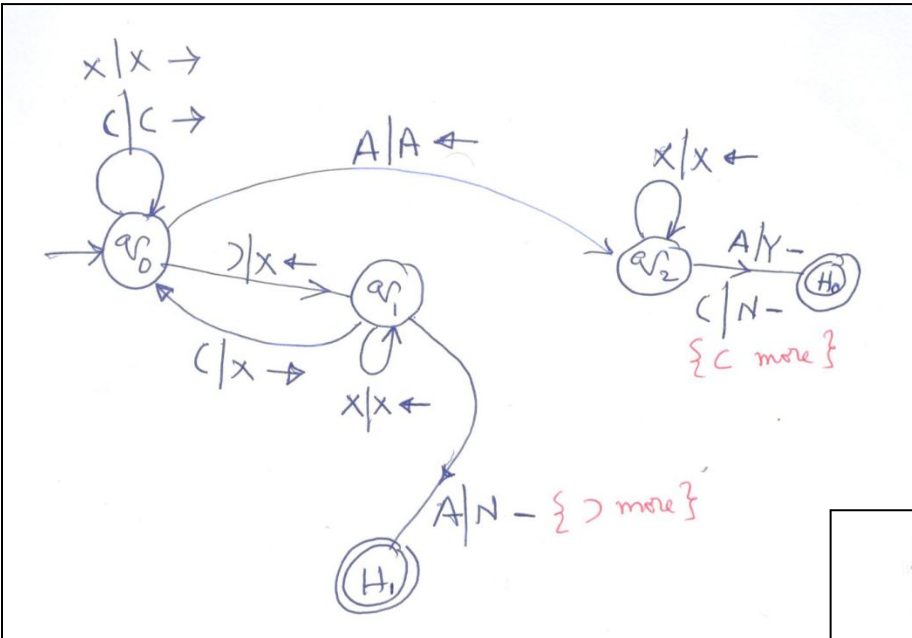
- The TM halts with a Y to mean Yes or a N to mean No.
- $Q = \{ q_0, q_1, q_2, H_0, H_1\}$
- $F = \{H_0, H_1 \}$

Idea


q_0 → Begin — Probe for)
Found
)
A
q_1
q_2
All crossed?
Yes
No
Yes Halt
No Halt
Search for matching (
found (
Found A
q_0
No, Halt

The transition diagram





$b\ b\ A\ (\ )\ (\ (\ )\ (\ )\ )\ A\ b\ b$

$\uparrow q_0$ (under the first "(")

$A\,q_0\,()(()())A \vdash A(\,q_0\,)(()())A$

$\vdash A\,q_1\,(X(()())A \vdash AX\,q_0\,X(()())A$

$\vdash AXX\,q_0\,(()())A \vdash AXX(\,q_0\,()())A$

$\vdash AXX((\,q_0\,)())A \vdash AXX(\,q_1\,(X())A$

$\vdash AXX(X\,q_0\,X())A \vdash^* AXX(XX(\,q_0\,))A$

$\vdash AXX(XX\,q_1\,(X)A \vdash AXX(XXX\,q_0\,X)A$

$\vdash AXX(XXXX\,q_0\,)A \vdash AXX(XXX\,q_1\,XXA$

$\vdash^* AXX\,q_1\,(XXXXXA \vdash AXXX\,q_0\,XXXXXA$

$\vdash^* AXXXXXXX\,q_0\,A \vdash AXX..X\,q_2\,XA$

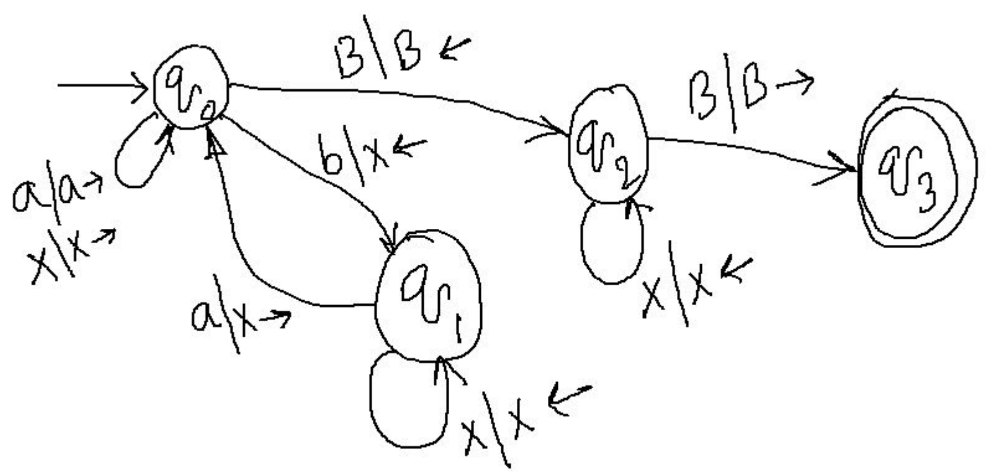
$\vdash^* q_2\,AXX\ldots XA \vdash H_0\,Y\,XX\text{--}XA$

Halt o/p

For the same problem (well balanced paranthesis) as a recognition problem

- Let (and) are represented as a and b, respectively. (B is the blank symbol)

	B	X	a	b
→ q0	(q2,B,L)	(q0,X,R)	(q0,a,R)	(q1,X,L)
q1		(q1,X,L)	(q0,X,R)	
q2	(q3,B,R)	(q2,X,L)		
*q3				



The language accepted by a TM M

- Let *M* be a TM, M = (Q, Σ, Γ, δ, q_0, B, F)
- $L(M) = \{w \in \Sigma^* | q_0 w \vdash^* \alpha p \beta$, where $p \in F$, and $\alpha, \beta \in \Gamma^*\}$.
- Note that, input string getting exhausted is not present (in contrast to the acceptance criterion by PDAs, DFAs, etc).
- The language accepted by a TM is also said the language recognized by a TM.

Recursively enumerable

- If $w \in L(M)$, then M accepts/recognizes the string w.
- If, $w \notin L(M)$, then M may or may not halt.
  - No transition means M halts and rejects.
- We say $L(M)$ for a given TM M is recursively enumerable (RE).

Recursive languages

- Recursive languages (R) is a subset of RE.
- We say $L(M)$ for a TM M is recursive, if for any given input string w, M halts.
- That is, if $w \in L(M)$, M halts in an accepting (final) state.
- Else, M halts in a non-final state.
  – i.e., It gets stuck in a non-final state.
- M never goes in to an infinite loop.

RE Vs. R

RE

- A TM M recognizes.

R

- A TM M decides.

RE Vs. R

RE

- A TM M recognizes.
- If $L \in RE - R$, then complement of L, that is $\bar{L} \notin RE$.

R

- A TM M decides.
- $L \in R \Leftrightarrow \bar{L} \in R$

RE Vs. R

RE

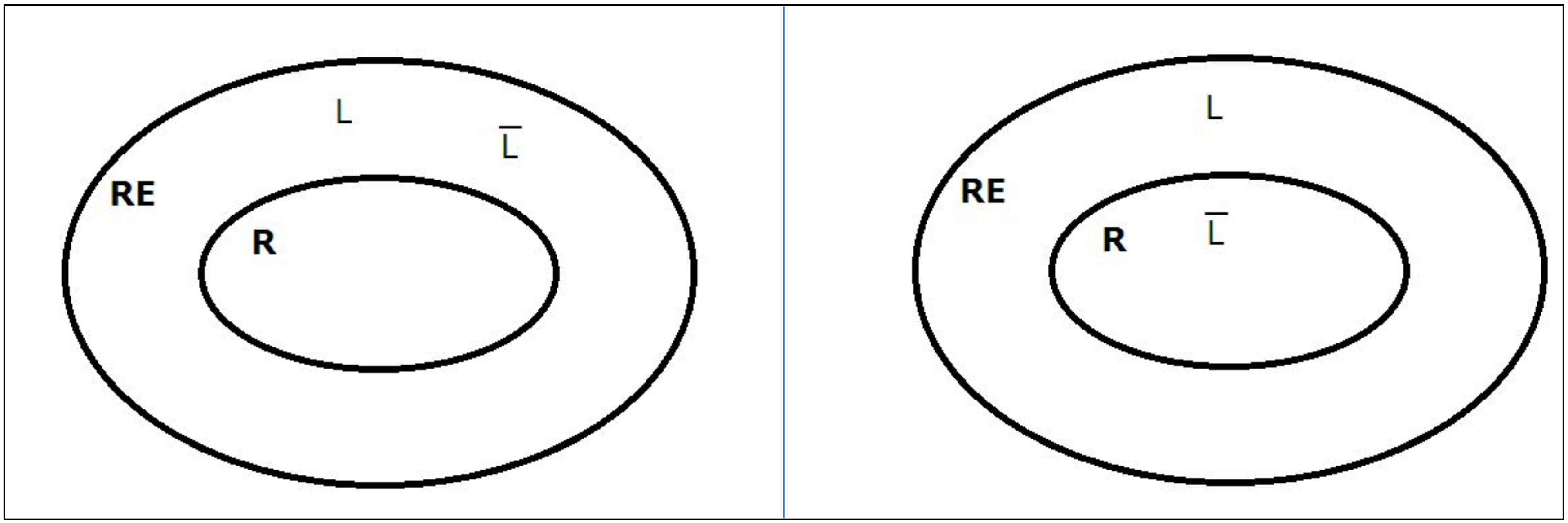
- A TM M recognizes.
- If $L \in RE - R$, then complement of L, that is $\overline{L} \notin RE$.

R

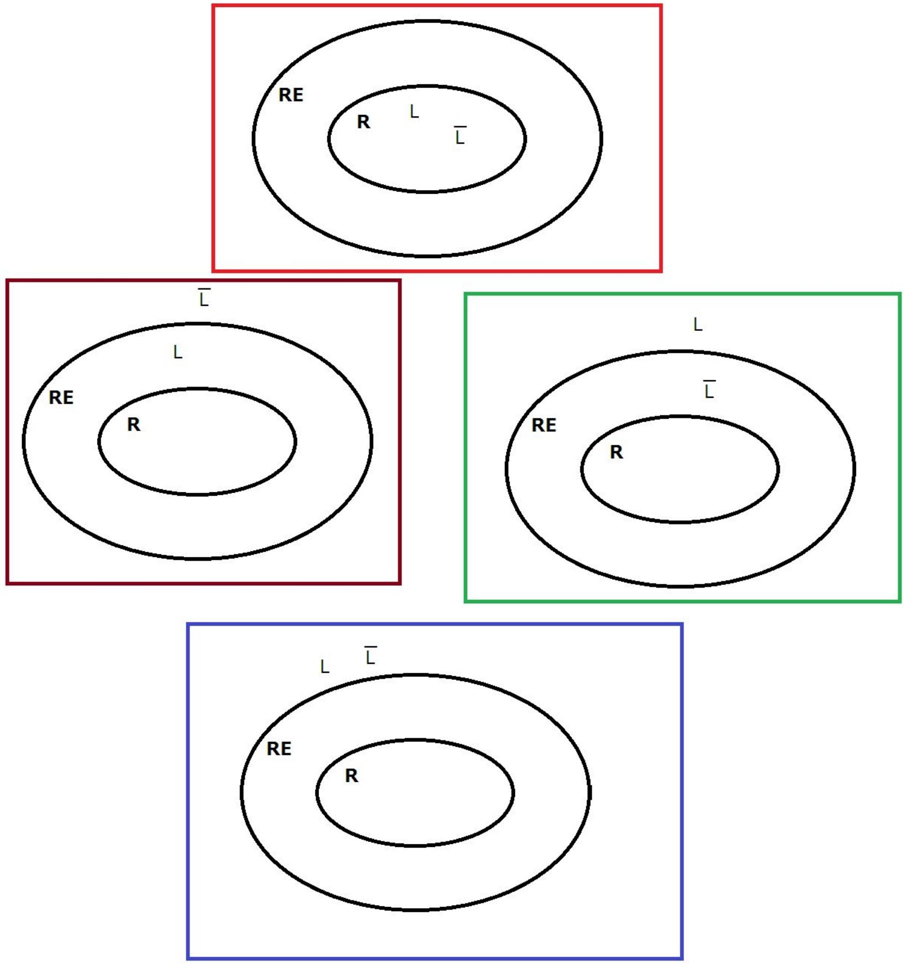
- A TM M decides.
- $L \in R \Leftrightarrow \overline{L} \in R$

$L \in RE \; and \; \overline{L} \in RE \quad \Rightarrow \quad L \in R$

Not Possible



Possible


RE
R
L
$\overline{L}$
$\overline{L}$
L
RE
R
L
$\overline{L}$
RE
R
L
$\overline{L}$
RE
R