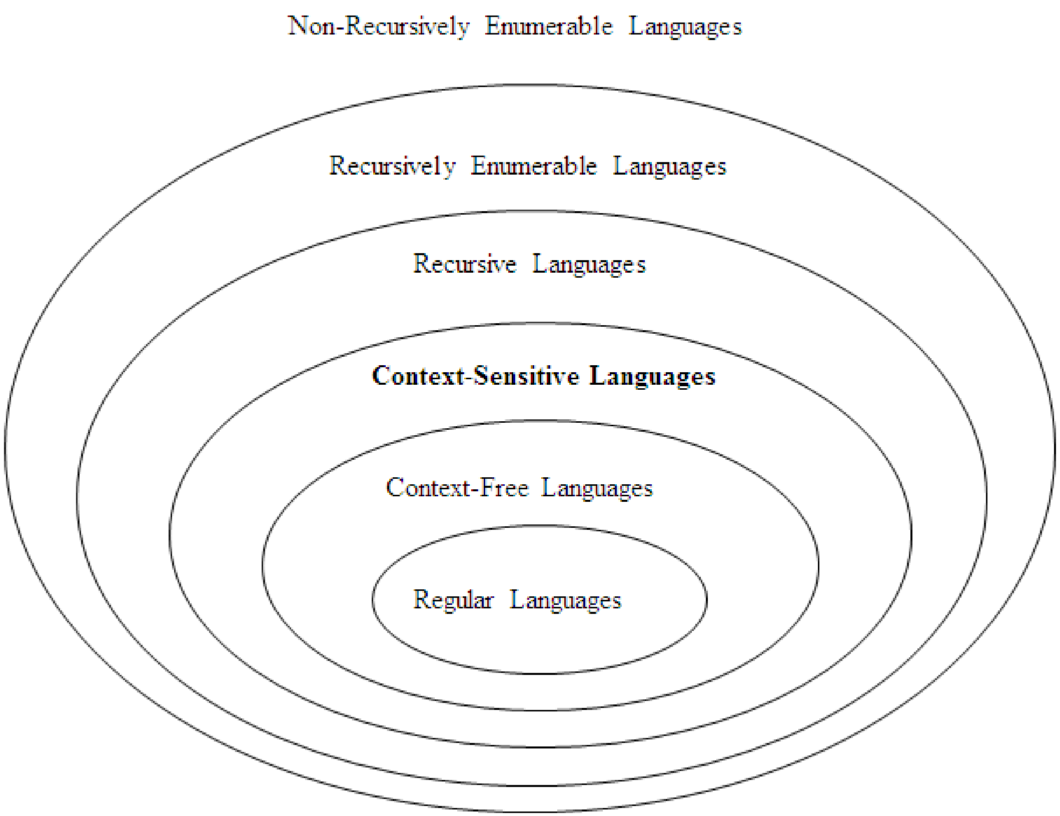
Turing Machines

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

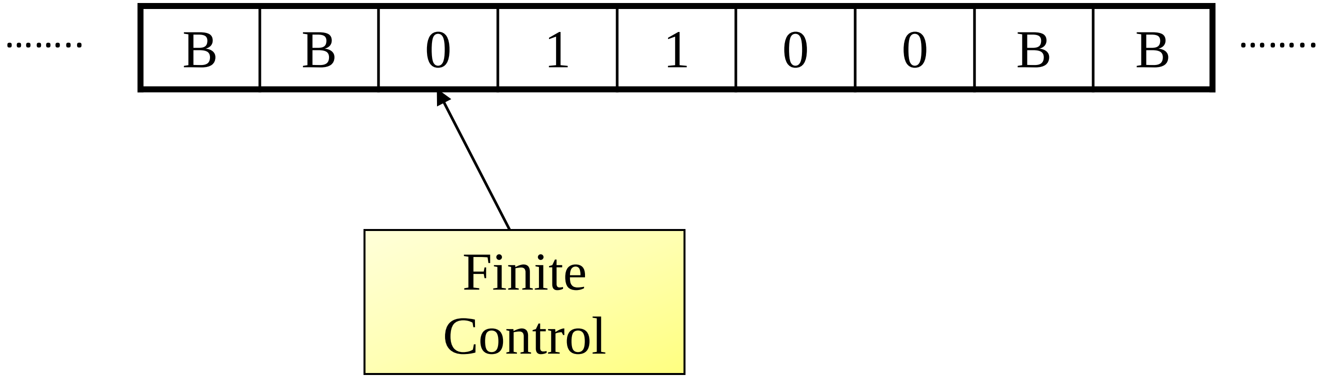
- TMs formalize the above notion.

- **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.

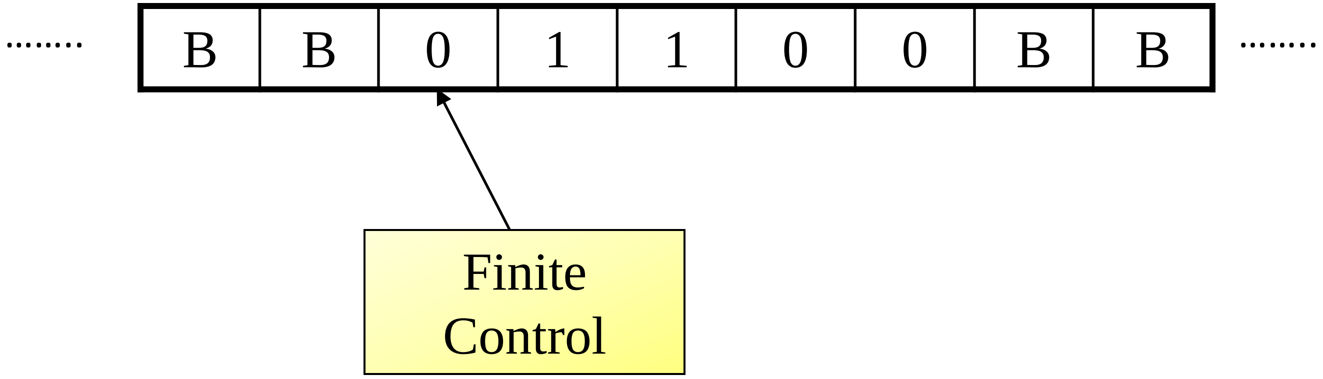

Universal
Turing Machine
I fear no man.
But that thing...
Halting
Problem
it scares me.

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**
F	**A 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

$Q = \{q_0, q_1, q_2\}$

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

$\Sigma = \{0, 1\}$

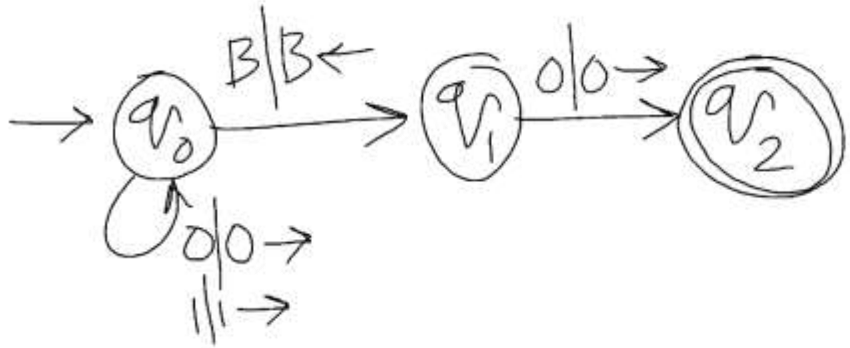
$F = \{q_2\}$

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}

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^*	-	-	-

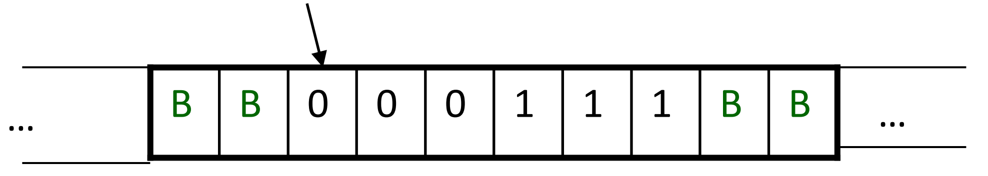
- **q_0 is the start state and the "scan right" state, until hits B**
- **q_1 state is to begin the verification – last character is 0 or not**
- **q_2 is the final state**

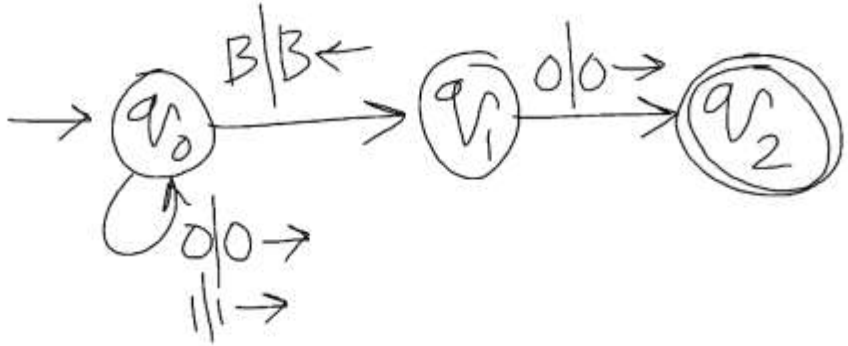
ID

- To describe the steps we use Instantaneous Descriptions.
- 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…

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

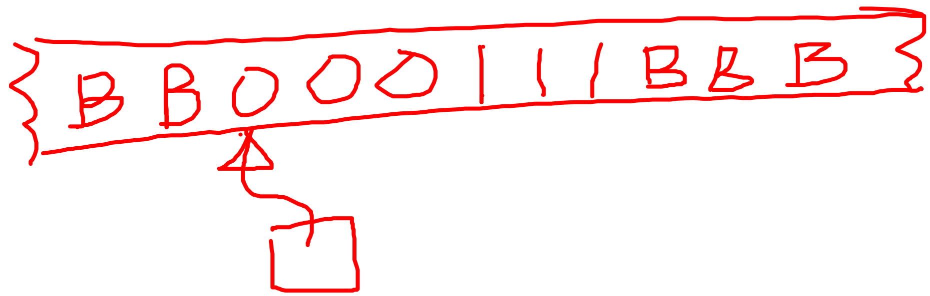
Have you noticed

- 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.

- $\{0^n1^n | n \geq 1\}$
- 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

- $\{0^n1^n | n \geq 1\}$
- Idea??

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

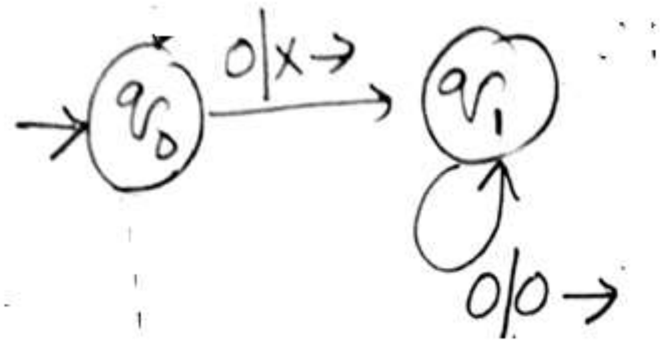
B	B	X	0	0	1	1	1	B
			$\uparrow$ q_1					

- 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

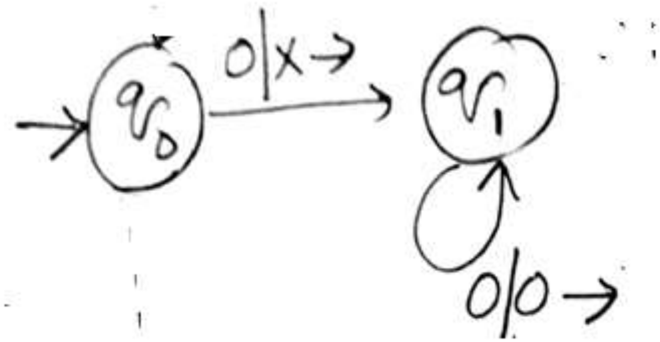
B	B	X	0	0	1	1	1	B
			↑					

q_1

⋮

B	B	X	0	0	1	1	1	B
					↑			

q_1



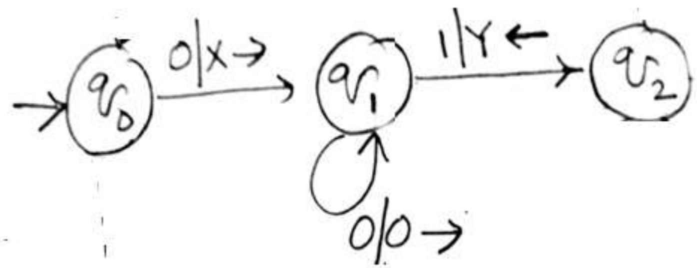
B	B	0	0	0	1	1	1	B
		↑ q_0						

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
					↑			

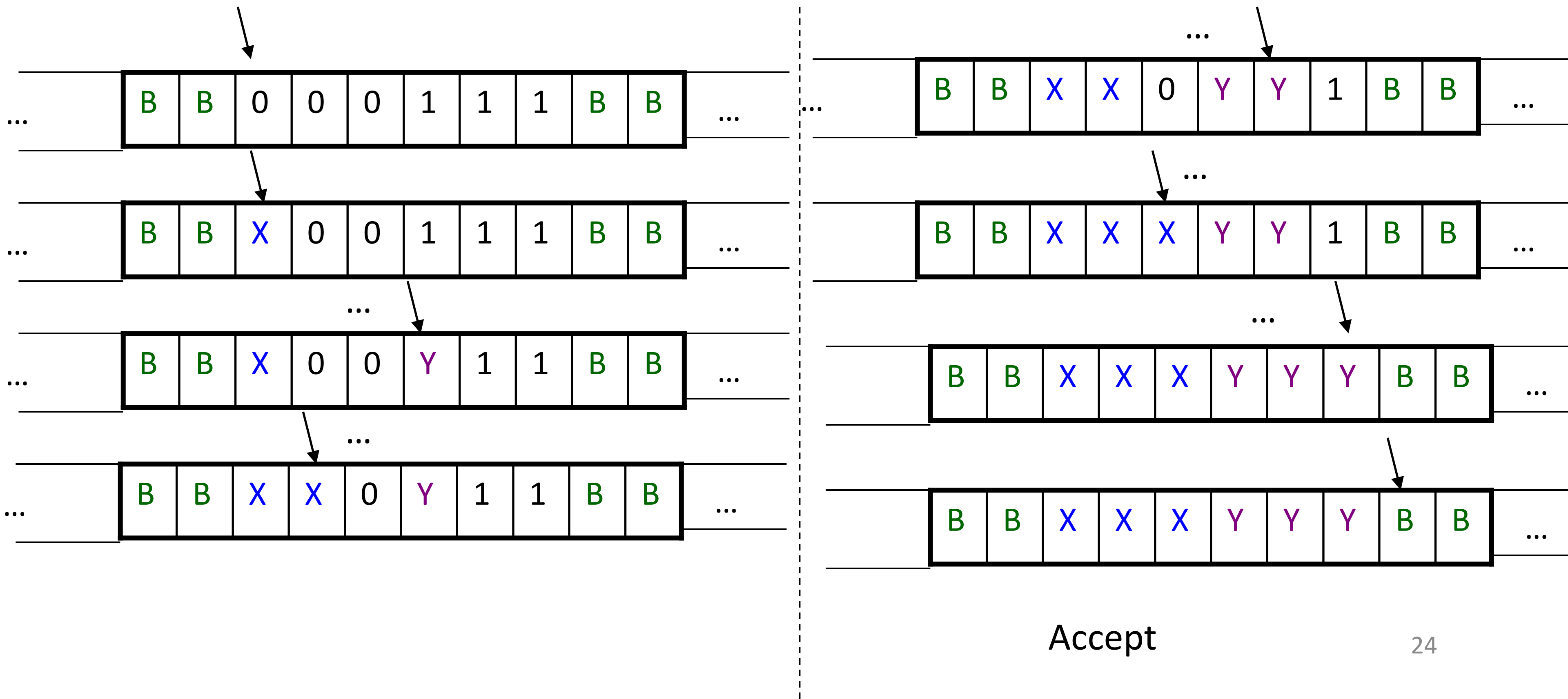
q_1

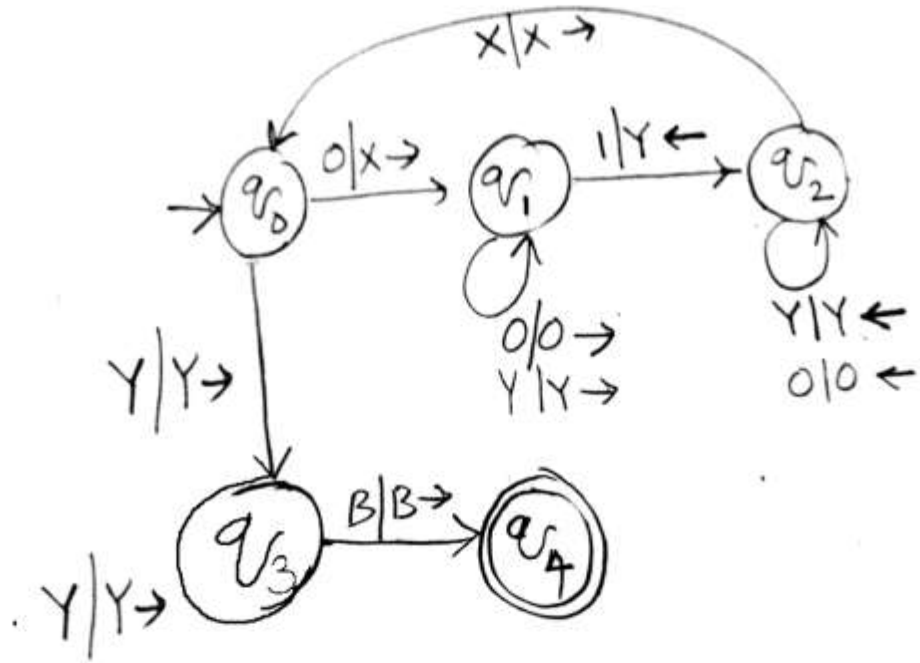
B	B	X	0	0	Y	1	1	B
				↑				

q_2

Example: L = $\{0^n1^n \mid n \geq 1\}$

- Strategy: w = 000111



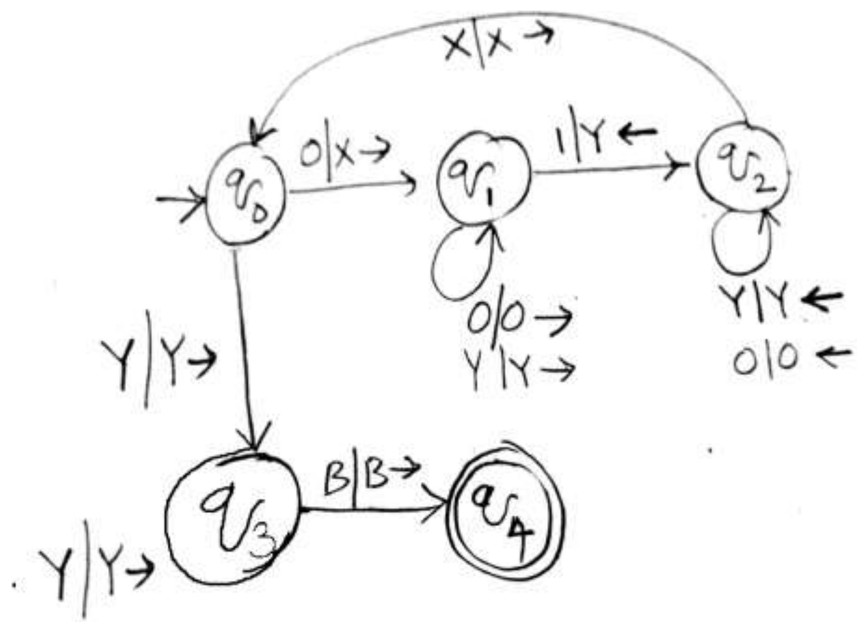

X|X →
0|X →
1|Y ←
q0
q1
q2
0|0 →
Y|Y →
Y|Y ←
0|0 ←
Y|Y →
q3
B|B →
q4
Y|Y →

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

	0	1	X	Y	B
$\rightarrow q_0$	(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.. |— Xq_1011
|— $X0q_111$
|— Xq_20Y1
|— q_2X0Y1
|— Xq_00Y1
|— XXq_1Y1
|— $XXYq_11$
|— XXq_2YY
|— Xq_2XYY
|— XXq_0YY
|— $XXYq_3Y$ B…
|— $XXYYq_3$ BB…
|— $XXYYBq_4$



TMs for calculations

- TMs can also be used for calculating values
  - Like arithmetic computations
  - Eg., addition, subtraction, multiplication, etc.

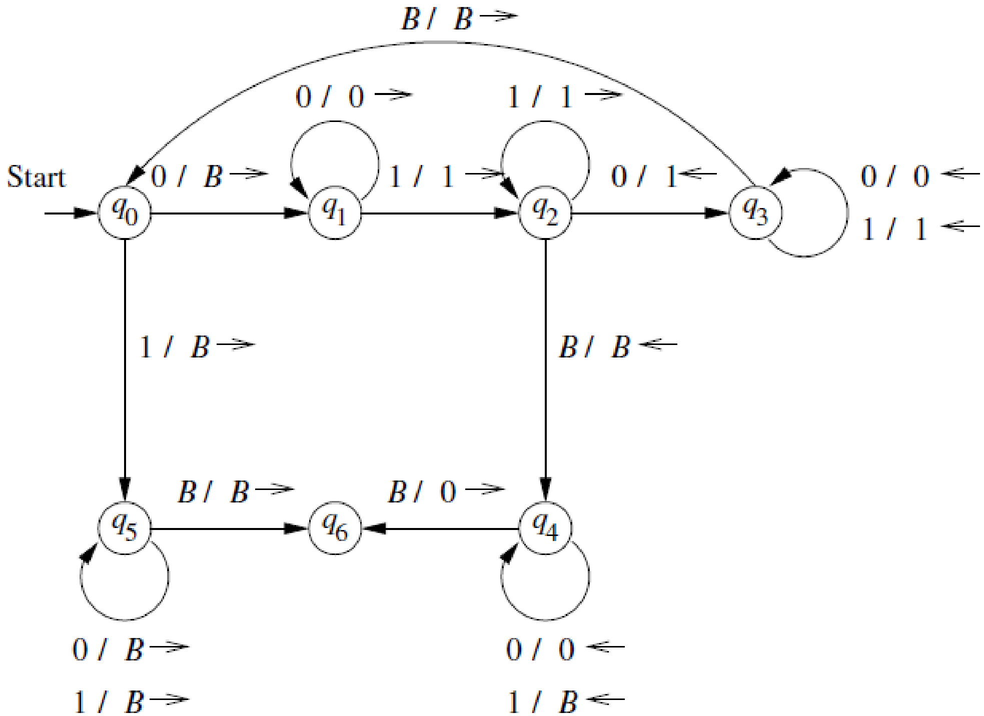
Example 2: monus subtraction

"m -- n" = max{m-n,0}

0^m10^n ➔ ...B 0^{m-n} B.. (*if m>n*)
...BB...B.. (*otherwise*)

1. For every 0 on the left (mark B), mark off a 0 on the right (mark 1)
2. Repeat process, until one of the following happens:
    1. // No more 0s remaining on the left of 1
       Answer is 0, so flip all excess 0s on the right of 1 to Bs (and the 1 itself) and halt
    2. //No more 0s remaining on the right of 1
       Answer is m-n, so simply halt after making 1 to B

Give state diagram



B / B →
0 / 0 →
1 / 1 →
Start
0 / B →
1 / 1 →
0 / 1 ←
0 / 0 ←
1 / 1 ←
q_0
q_1
q_2
q_3
1 / B →
B / B ←
B / B →
B / 0 →
q_5
q_6
q_4
0 / B →
1 / B →
0 / 0 ←
1 / B ←