Theory of Computing:

10. Turing Machine - 2

Professor Imed Bouchrika
National School of Artificial Intelligence
imed.bouchrika@ensia.edu.dz

Outline :

- TM as
  - Language Acceptor/Recognizer
  - Transducer and Examples
  - Problem Solver
- Variations of Turing Machines
- Universal Turing Machine
- Algorithms and Church's Thesis
- Turing-Complete Systems

Formalism of Turing Machine

A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta \colon Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Classes of Languages for Turing Machine

- The collection of strings that M **<u>accept</u>**s is the language of M , or the language recognized by M
    - A language is called **Turing-recognizable** if some Turing machine recognizes it
    - Mainly : Accepting words that belong to the language.
    - For words not in the language:
        - Reject or Loop

Classes of Languages for Turing Machine

- **Turing-decidable** language or simply decidable if some Turing machine decides it
  - Halts and Accepts for words in the language
  - Halts and Reject for words not in the language
- Every Decidable language is also recognizable.

Classes of Languages for Turing Machine

- **Terminologies:**
    - Turing Recognizable is called a **recursively enumerable language** in some other textbooks.
    - For turing decidable is called a **recursive language**

TM as Language Recognizer/Acceptor

- What about the following language :
  - Even length palindromes
  - Algorithm ?
  - Turing Machine ?

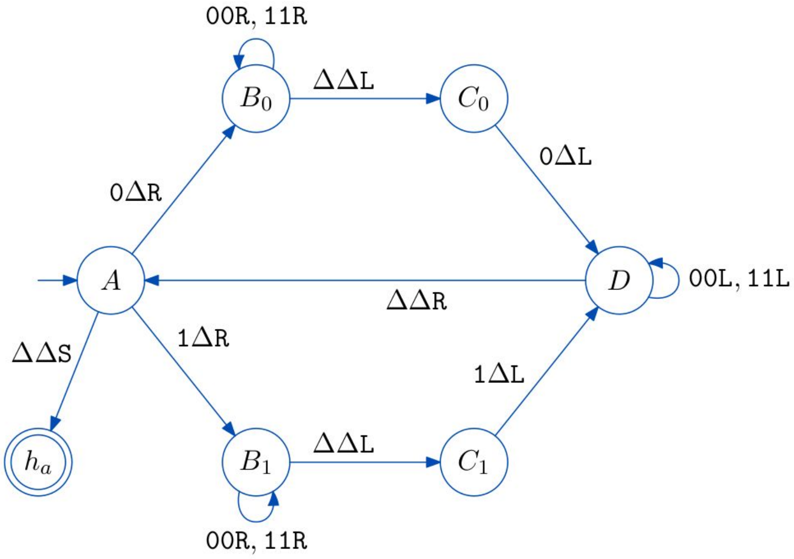
TM as Language Recognizer/Acceptor

- What about the follo
  - Even length pali
  - Algorithm :

1. **If first letter is :**
   1.1. **0 → Remove and Skip to Blank at extreme Right, Move Left Remove 0**
   1.2. **1 → Remove and Skip to Blank at extreme Right, Move Left Remove 1**
   1.3. **Blank → Accept**
2. **Skip to Blank at extreme left.**
3. **Go to Step 1**

TM as Language Recognizer/Acceptor

- What about the following language :
  - Even length palindromes
  - Turing Machine

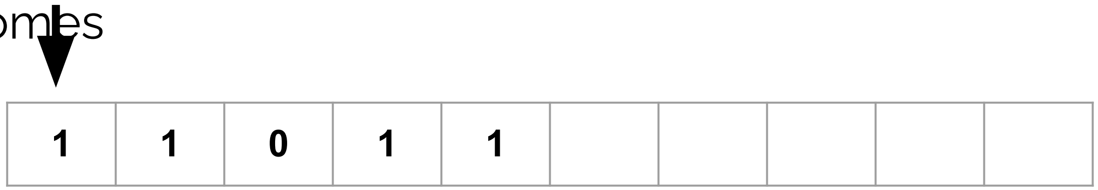


TM as Language Recognizer/Acceptor

- What about the following language :
  - Odd-length palindromes
  - Algorithm ?
  - Turing Machine ?

TM as Language Recognizer/Acceptor

- What about the following language :
  - Odd-length palindromes
  - Algorithm :



TM as Language Recognizer/Acceptor

- What about the following language :
  - Odd-length palindromes
  - Algorithm :

1	1	0	1	1					

	1	0	1	1					

TM as Language Recognizer/Acceptor

- What about the following language :
  - Odd-length palindromes
  - Algorithm :

1	1	0	1	1					

	1	0	1	1					

	1	0	1	X					

TM as Language Recognizer/Acceptor

- What about the following language :
  - Odd-length palindromes
  - Algorithm :

1	1	0	1	1					

	1	0	1	1					

	1	0	1	X					

		0	X	X					

TM as Language Recognizer/Acceptor

- What about the following language :

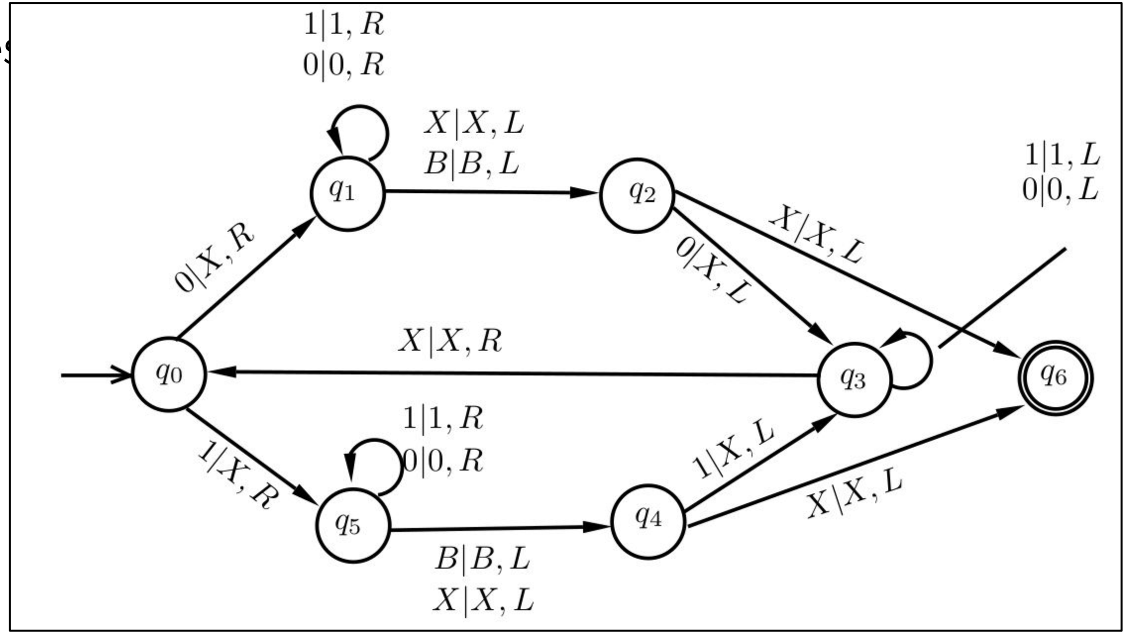
romes

If next transition takes directly to X, Just Accept

1	1	0	1	1					
	1	0	1	1					
	1	0	1	X					
			X	X					

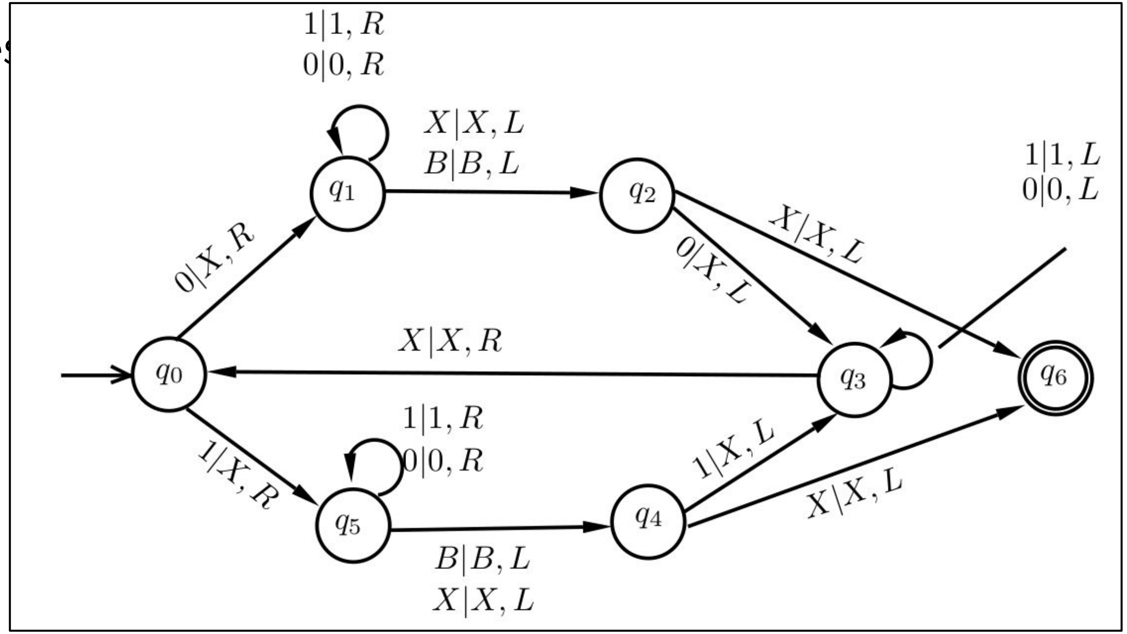
TM as Language Recognizer/Acceptor

- What about the following language :
  - Odd-length palindromes
  - Turing Machine ?



TM as Language Recognizer/Acceptor

- What about the following language :
  - Odd-length palindromes
  - Turing Machine ?



This Turing replaces even blanks at the left with X

TM as Language Recognizer/Acceptor

- Given an input string composed of two words separated by the symbol # design the algorithm to check if they are equal
  - If Equal → Accept
  - Otherwise → Reject

TM as Language Recognizer/Acceptor : w#w

- Given an input string composed of two words separated by the symbol # design the algorithm to check if they are equal
    - If Equal → Accept
    - Otherwise → Reject

TM as Language Recognizer/Acceptor : ww

- ? What's the algorithm ?
  - How to detect whether a word is of an even length ?
  - How to inject the symbol **#** into the middle ? or other approach ?
    - Example :
      - 101001 ⇒ 101#001
      - 101001 ⇒ BABXXZ
      - ..?

Turing Machine as Transducers

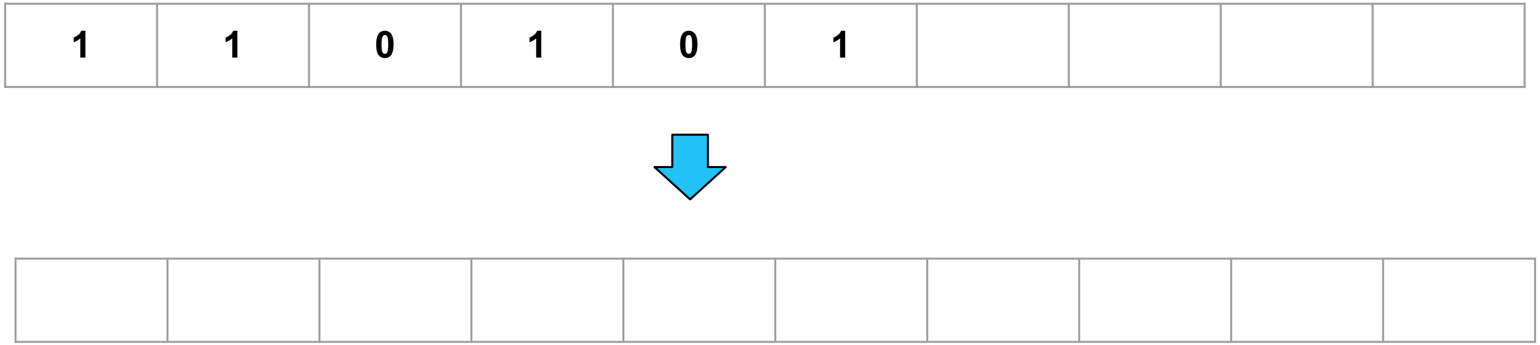
- Transducers are the devices used to convert one form of a signal into a different form.
- Transducer is a type of Turing Machine that is used to convert the given input into a specific output after the machine performs various read-writes.
- It doesn't accept or reject an input but performs series of operations to obtain the output right in the same tape and halts when finished.

Turing Machine as Transducers

- Such type of Turing machine can be used to do:
  - Computation
    - Example :
      - Addition
  - Operation on input strings and text processing
    - Example :
      - Removing special patterns from a string

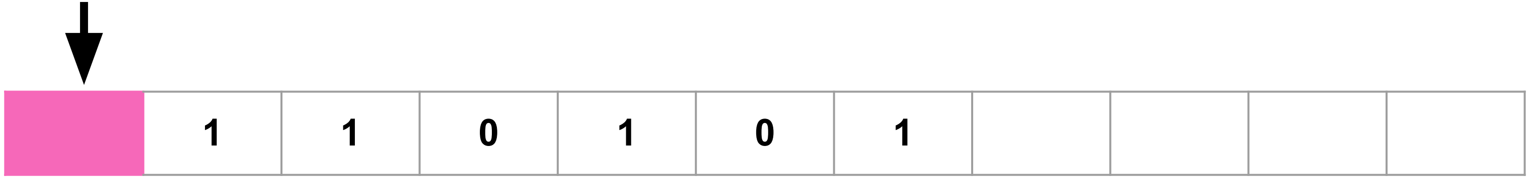
TM as Transducers :
1. Erase input string

- Given an input string, Design the Turing Machine to erase all input symbols



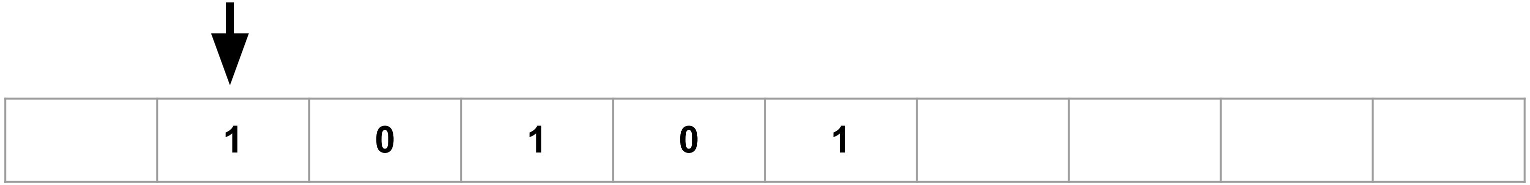
TM as Transducers :
1. Erase input string

- Given an input string, Design TM to erase all input symbols
- Algorithm:



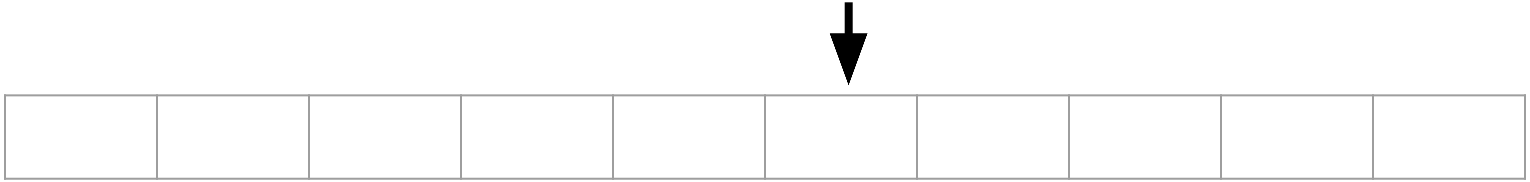
TM as Transducers :
1. Erase input string

- Given an input string, Design Turing Machine to erase all input symbols
- Algorithm:



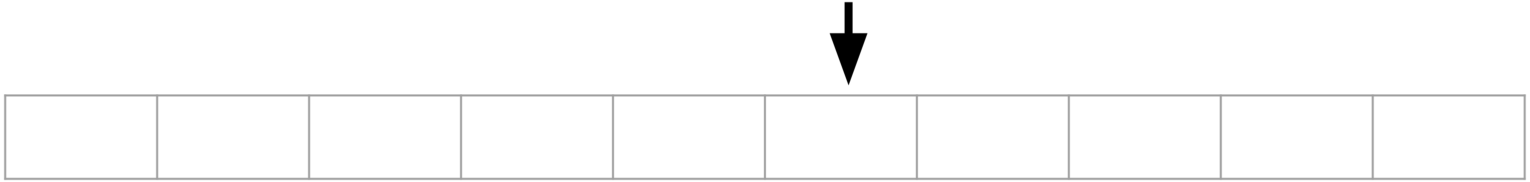
TM as Transducers :
1. Erase input string

- Given an input string, Design Turing Machine to erase all input symbols
- Algorithm:



TM as Transducers :
1. Erase input string

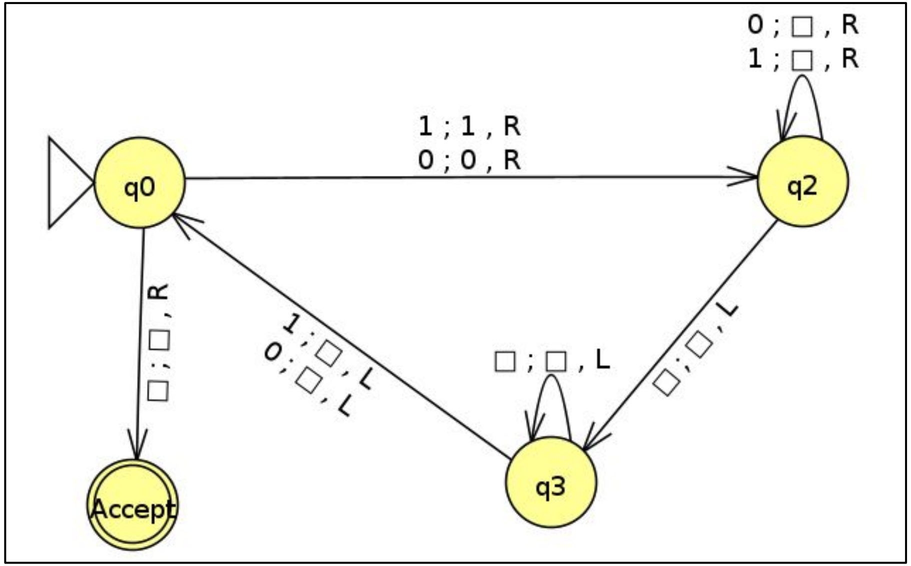
- Given an input string, Design Turing Machine to erase all input symbols
- Algorithm:



How to restore the head to its original position ?
We skip left till we find what ?

TM as Transducers :
1. Erase input string

- Given an input string, Design Turing Machine to erase all input symbols
- Algorithm:



TM as Transducers :
2. Shift Input to Right

- Given an input string, design the algorithm to Shift all all symbols to the right by one cell.

1	1	0	1	0	1				

	1	1	0	1	0	1			

TM as Transducers :
2. Shift Input to Right

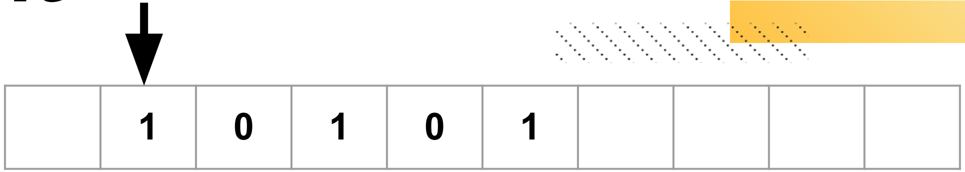
- Given an input string, design the algorithm to Shift all all symbols to the right by one cell.
- Algorithm :

1	1	0	1	0	1				

If first 1 :
DELETE
Move Right
Take Branch A
If first 0 :
DELETE
Move Right
Take Branch B

TM as Transducers :
2. Shift Input to Right

- Given an input string, design the algorithm to Shift all all symbols to the right by one cell.
- Algorithm :



On Branch A :

If 1 :

Keep it

Move Right On Branch A

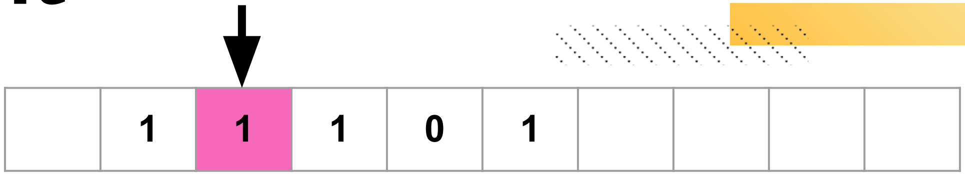
If 0 :

Invert to 1

Move Right on Branch B

TM as Transducers :
2. Shift Input to Right

- Given an input string, design the algorithm to Shift all all symbols to the right by one cell.
- Algorithm :



On Branch A :

If 1 :

Keep it

Move Right On Branch A

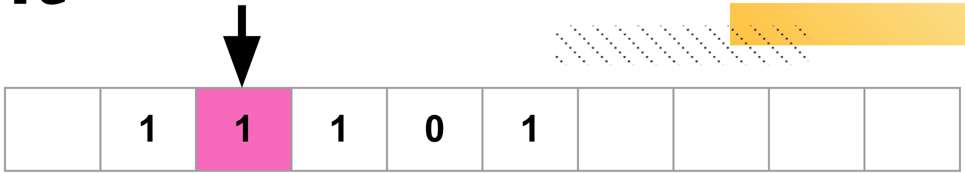
If 0 :

Invert to 1

Move Right on Branch B

TM as Transducers :
2. Shift Input to Right

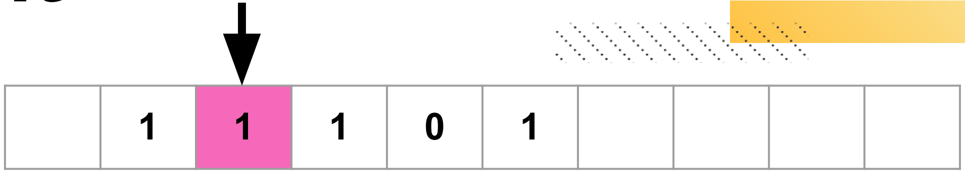
- Given an input string, design the algorithm to Shift all all symbols to the right by one cell.
- Algorithm :



On Branch B :
If 0 :
Keep it
Move Right On Branch B
If 1 :
Invert to 0
Move Right on Branch A

TM as Transducers :
2. Shift Input to Right

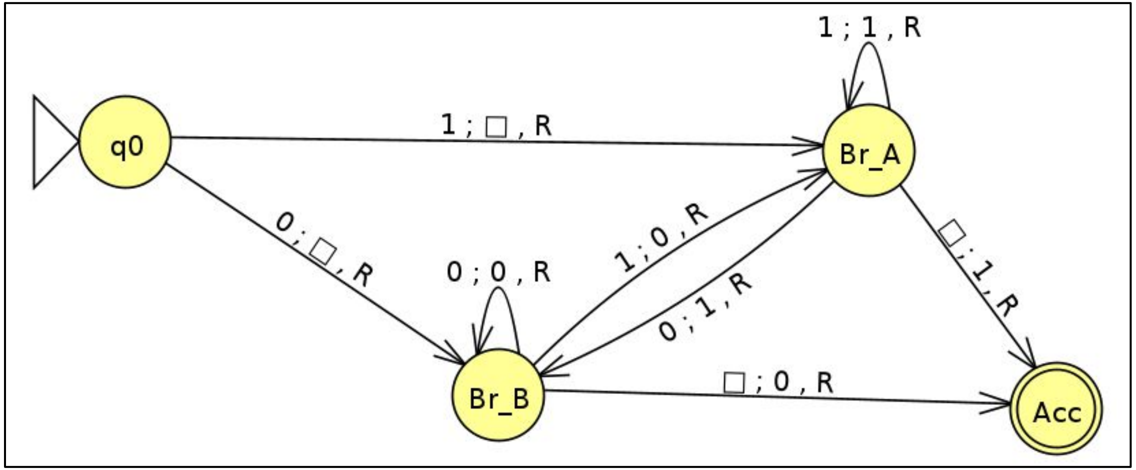
- Given an input string, design the algorithm to Shift all all symbols to the right by one cell.
- Algorithm :



When to Stop ?

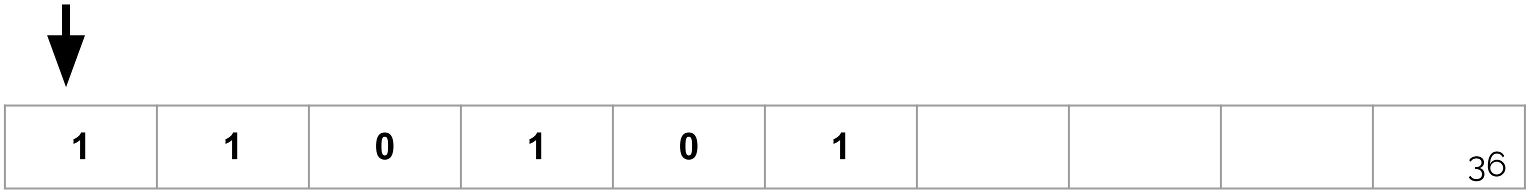
TM as Transducers :
2. Shift Input to Right

- Given an input string, design the algorithm to Shift all symbols to the right by one cell.
- Turing Machine :



TM as Transducers :
2. Shift Input to Right

- Given an input string, design the algorithm to Shift all all symbols to the right by one cell.
- Turing Machine :



TM as Transducers :
3. Copy a String

- Given an input string, design the algorithm to copy the full string separated with a blank to the original string.

1	1	0	1						

1	1	0	1		1	1	0	1	

TM as Transducers :
4. Copy a String

- Given an input string, design the algorithm to copy the full string separated with a blank to the original string.
- Algorithm ?
- Turing Machine ?

TM as Transducers :
4. Copy a String

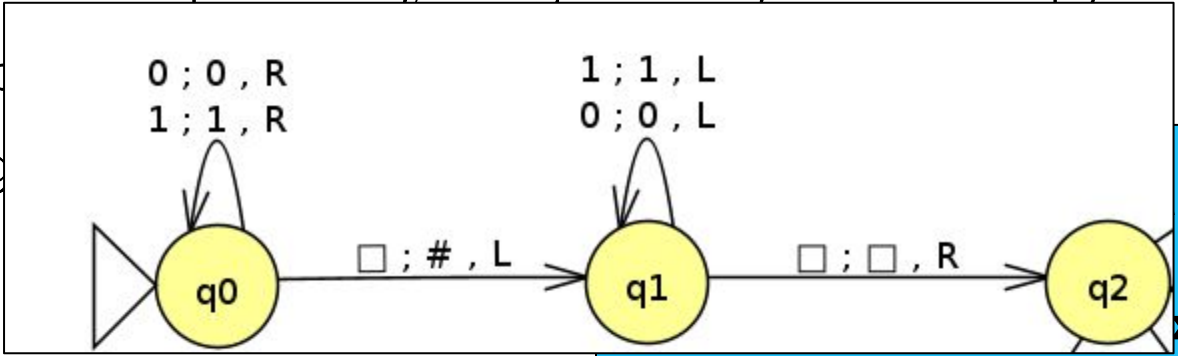
- Given an input string, design the algorithm to copy the full string separated with a blank to the original string.
- Algorithm :

1. Go to the Blank at extreme right
2. Replace it with #
3. Go back to Blank at extreme left, move right

1	1	0	1	#					

TM as Transducers : 4. Copy a String

- Given an input string, design the algorithm to copy the full string sep
- Alg



reme right

xtreme left,

move right

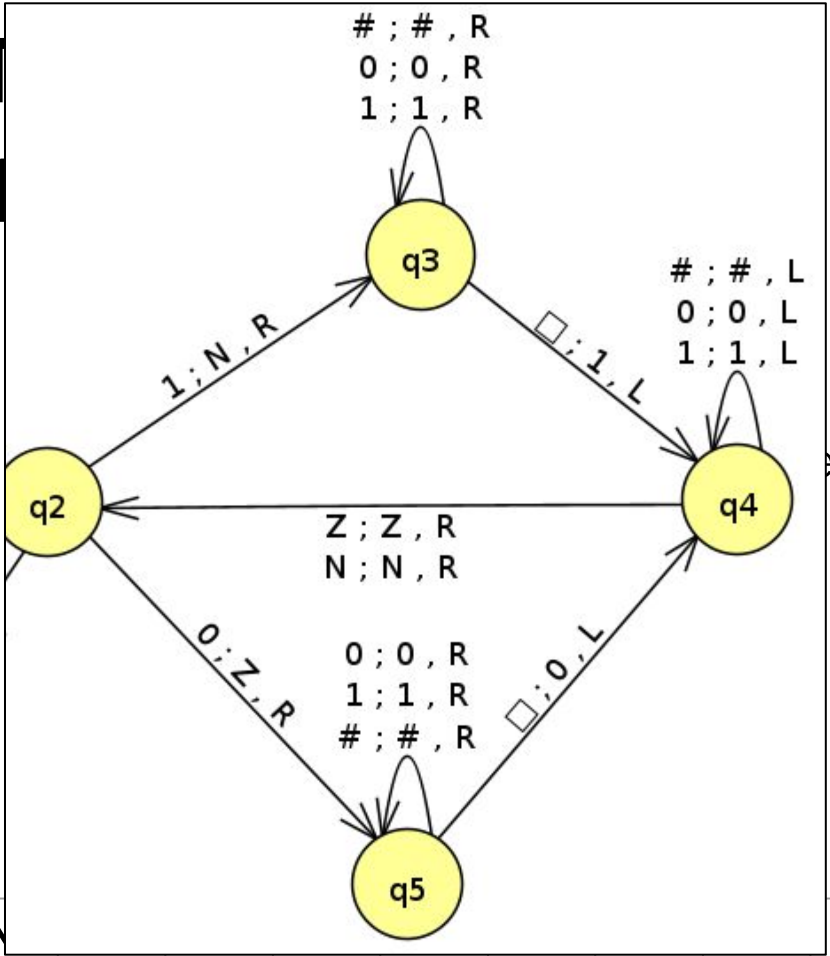
1	**1**	**0**	**1**	**#**					

TM as Transducers :
4. Copy a String

- Given an input string, design the algorithm to copy the full string separated with a blank to the original string.
- Algorithm :

1. **Replace 1 with N , Go to extreme right, write 1**
2. **Replace 0 with Z, Go to extreme right , write 0**
3. **Skip until you find either N or Z , move Right**

N	**1**	**0**	**1**	**#**	**1**				



the algorithm to copy the full string

e original string

1. **Replace 1 with N , Go to extreme right, write 1**
2. **Replace 0 with Z, Go to extreme right , write 0**
3. **Skip until you find either N or Z , move Right**

TM as Transducers :
4. Copy a String

- Given an input string, design the algorithm to copy the full string separated with a blank to the original string.
- Algorithm :

1. No more 1 or 0

2. Invert :

a. N → 1

b. Z → 0

c. # → Blank

N	N	Z	N	#	1	1	0	1	

TM as Transducers :
4. Copy a String

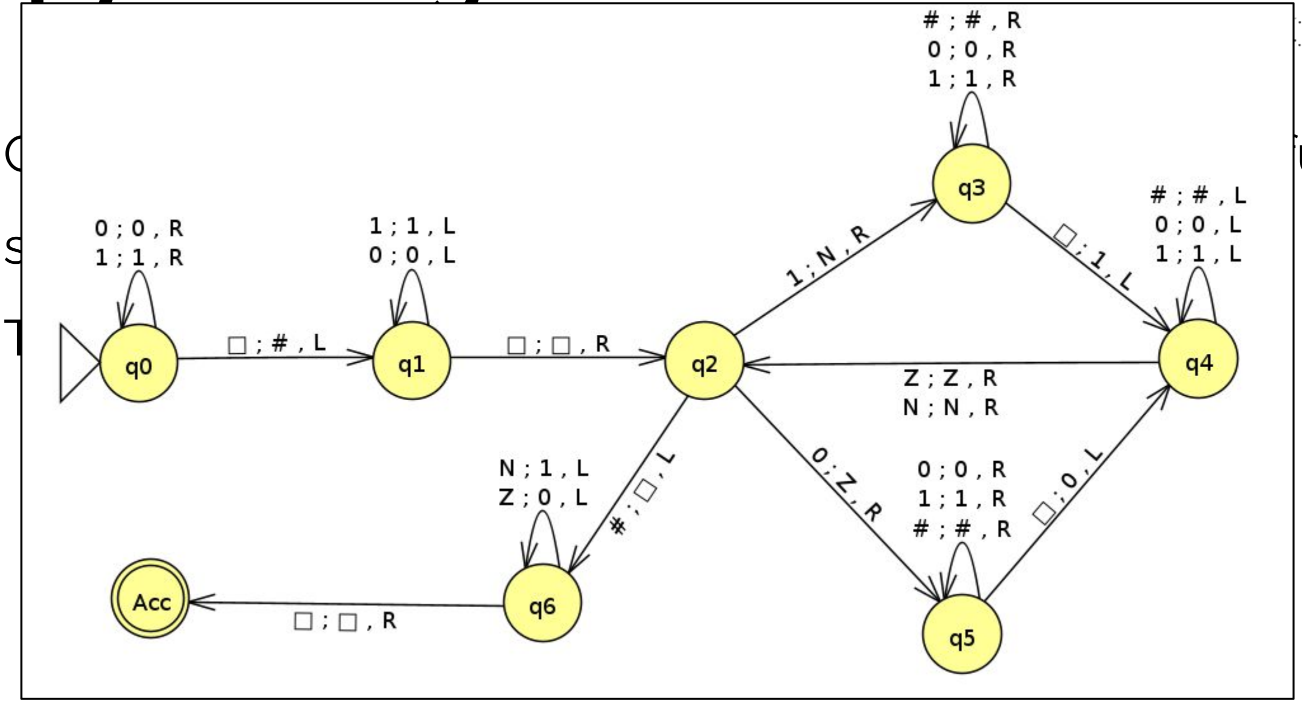
- Given an input string, design the algorithm to copy the full string separated with a blank to the original string.
- Algorithm :

1. No more 1 or 0
2. Invert :
   a. N → 1
   b. Z → 0
   c. # → Blank

1	1	0	1	#	1	1	0	1	

TM as Transducers :
4. Copy a String

- C full string
  s
- T



TM as Transducers :
5. Reversing a String

- Given an input string, reverse it or flip it.

1	1	0	0	1					

1	0	0	1	1					

Combining Turing Machines

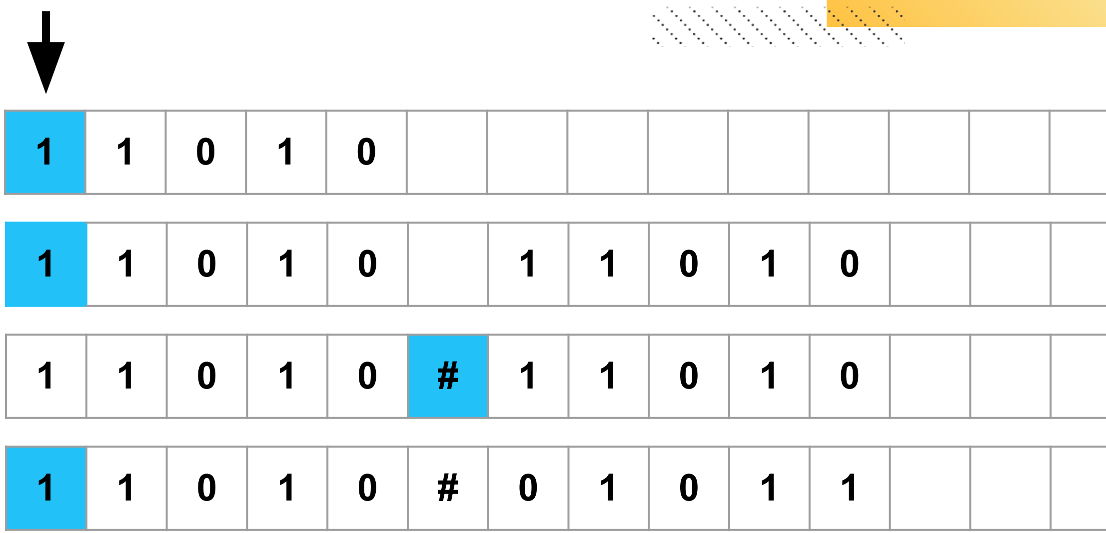
- Without designing a low-level algorithm for Accepting Palindromes (Odd and Even length), use existing Turing Machine Transducers to construct a language acceptor.

Combining Turing Machines

- Without designing a low-level algorithm for Accepting Palindromes, use existing Turing Machine Transducers to construct a language acceptor.
- Turing Machines to utilize :
    - Copy → Find_Blank_Replace_# → Reverse →Equal_Strings

Combining Turing Machines

- Turing Machines to utilize :
  - Copy
  - Find_Blank_Replace_#
  - Reverse
  - Equal_Strings : → Accept or → reject



TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number, Write the algorithm to do the addition

1	1	0	+	1	1				

1	0	0	1						

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number, Write the algorithm to do the addition
- Algorithm :

1	1	0	+	1	1				

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	1	0	+	1	1				

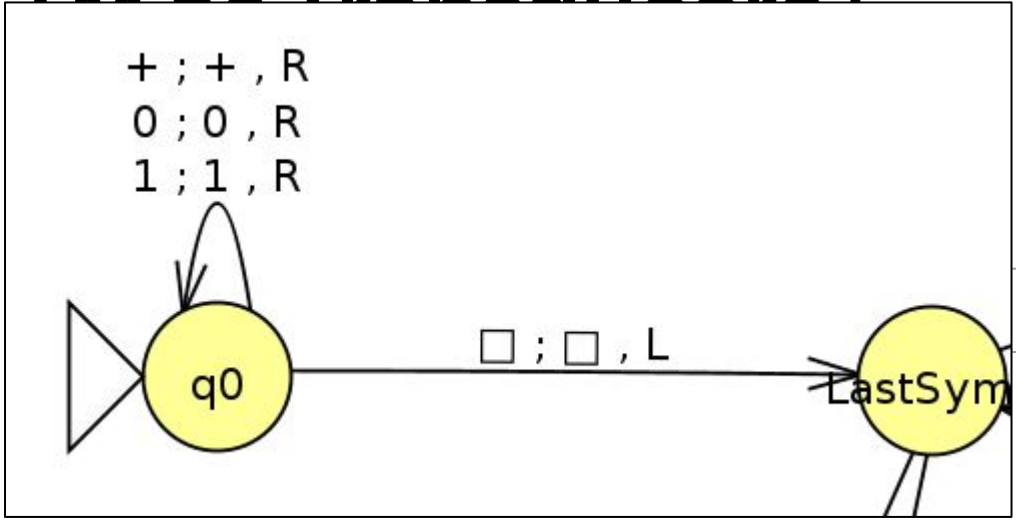
TM as Transducers : 6. Addition of Binaries

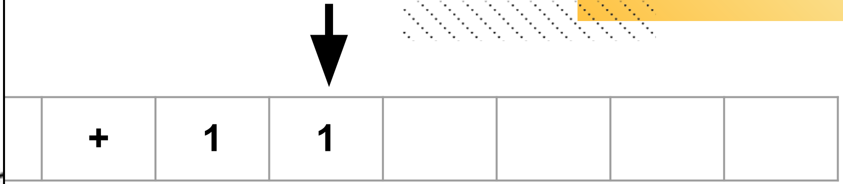
- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	1	0	+	1	1				

1. Go to blank at extreme right , move left one cell.
2. If:
   1 take a branch A
   0 take branch B
3. Erase the Cell

TM as Transducers :





- Write the algorithm to do the addition

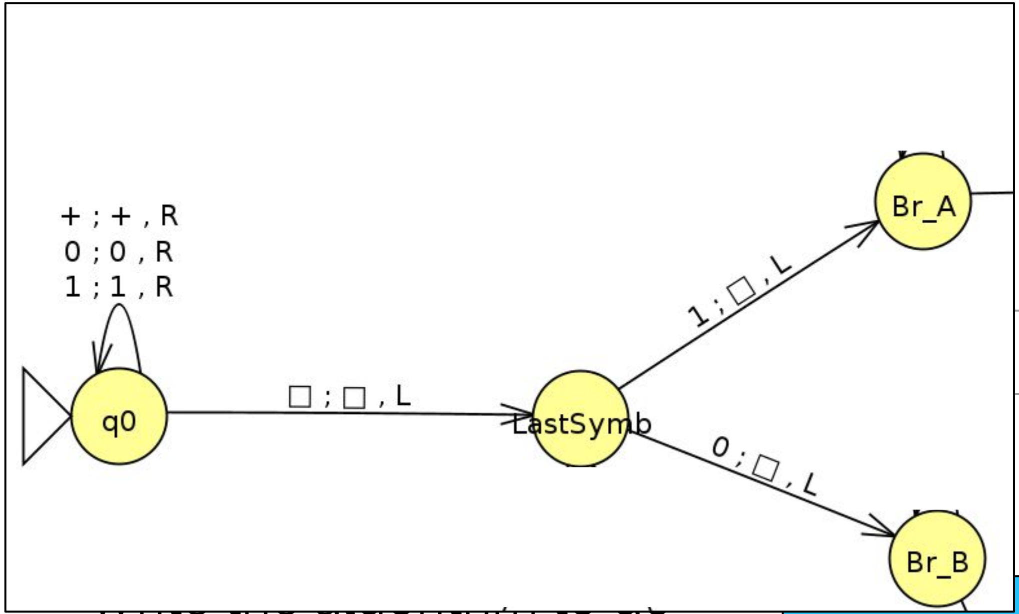
1. Go to blank at extreme right , move left one cell.

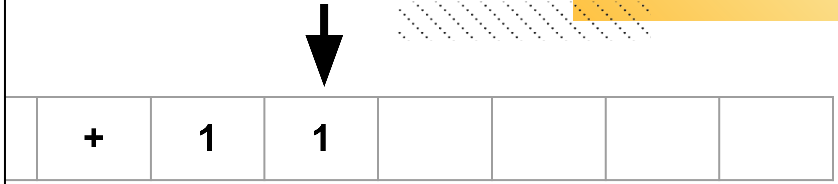
2. If:

    1 take a branch A

    0 take branch B

3. Erase the Cell





the addition

1. Go to blank at extreme right , move left one cell.

2. If:

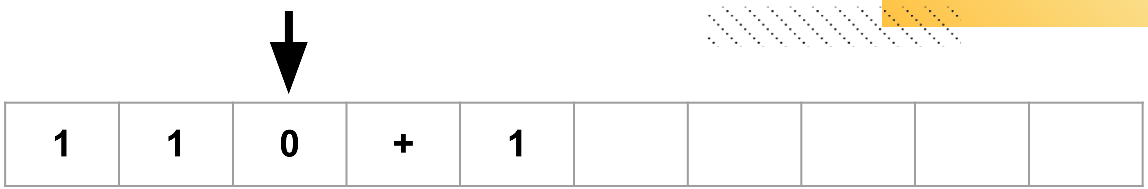
1 take a branch A

0 take branch B

3. Erase the Cell

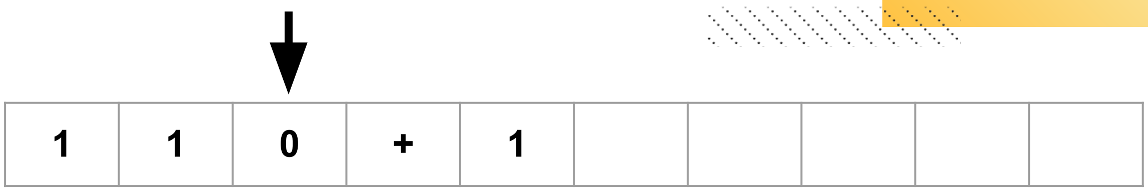
TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number **+** binary number,
- Write the algorithm to do the addition



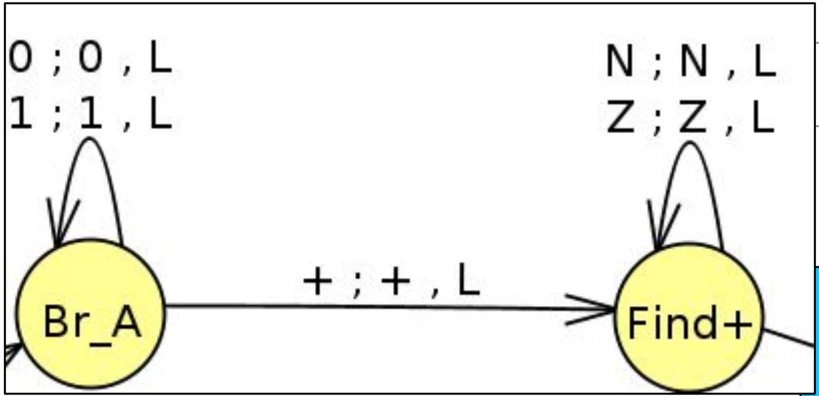
TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

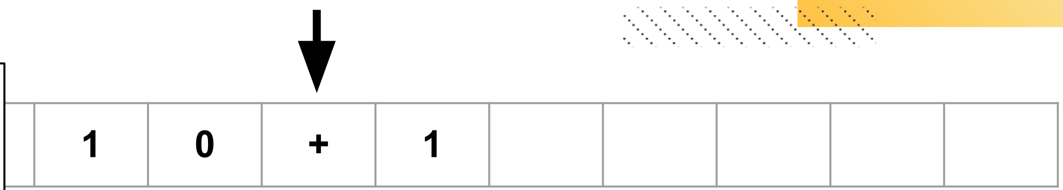


1. Skip Left 1 and 0 till we reach + :

TM as Transducers :
6. Addition of Binaries



the addition



1. Skip Left 1 and 0 till we reach + :

2. We Keep skipping N and Z to the left

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	1	0	+	1					

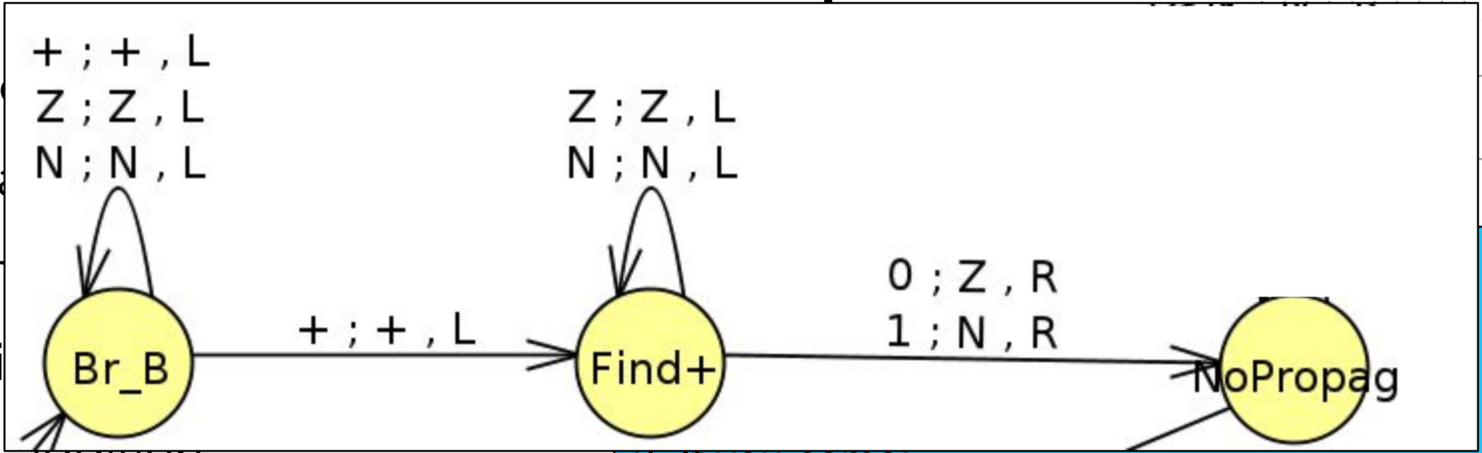
1. Skip Left 1 and 0 till we reach + :

2. We Keep skipping N and Z to the left

3. If you come:
   From Branch B: and you find:
   0 → Invert to Z
   1 → Invert to N

TM as Transducers :
6. Addition of Binaries

- Giv
bina
nur
- Wri
the addition



3. If you come:
   From Branch B: and you find:
   0 → Invert to Z
   1 → Invert to N

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	1	N	+	1					

1. Skip Left 1 and 0 till we reach + :
2. We Keep skipping N and Z to the left
3. If you come:

   From Branch **A** : and you find:

   0 → Invert to N

   1 → Invert to Z ??

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number **+** binary number,
- Write the algorithm to do the addition

1	**1**	**N**	**+**	**1**					

1. Skip Left 1 and 0 till we reach + :
2. We Keep skipping N and Z to the left
3. If you come:

    From Branch **A** : and you find:

    0 → Invert to N

    1 → Invert to Z and

        keep looping left :

            1 → 0

            0 → 1 and break from Loop

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number **+** binary number,
- Write the algorithm to do the addition

1	**Z**	**N**	**+**						

1. Skip Left 1 and 0 till we reach + :
2. We Keep skipping N and Z to the left
3. If you come:

    From Branch **A** : and you find:

    0 → Invert to N

    1 → Invert to Z and

        keep looping left :

            1 → 0

            0 or blank→ 1 and break from Loop

3. Go to Blank at Extreme right and repeat

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	Z	N	+						

1. Skip Left 1 and 0 till we reach + :
2. We Keep skipping N and Z to the left
3. If you come:

    From Branch **A** : and you find:

    0 → Invert to N

    1 → Invert to Z and

        keep looping left :

            1 → 0

            0 or blank→ 1 and break from Loop

3. Go to Blank at Extreme right and repeat

TM as Transducers : 6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

	0	Z	N	+						

1. Skip Left 1 and 0 till we reach + :
2. We Keep skipping N and Z to the left
3. If you come:

    From Branch **A** : and you find:

    0 → Invert to N

    1 → Invert to Z and

        keep looping left :

            1 → 0

            0 or blank → 1 and break from Loop

3. Go to Blank at Extreme right and repeat

TM as Transducers : 6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	0	Z	N	+						

1. Skip Left 1 and 0 till we reach + :
2. We Keep skipping N and Z to the left
3. If you come:

   From Branch **A** : and you find:

   0 → Invert to N

   1 → Invert to Z and

   keep looping left :

   1 → 0

   0 or blank → 1 and break from Loop

3. Go to Blank at Extreme right and repeat

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	0	Z	N	+						

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	0	Z	N	+						

1. If + is on the extreme right.
   1. Delete the + sign
   2. Skip to extreme LEFT and Invert :

      N → 1

      Z → 0

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	0	Z	N							

1. If + is on the extreme right.
   1. Delete the + sign
   2. Skip to extreme LEFT and Invert :

      N → 1

      Z → 0

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	0	Z	1							

1. If + is on the extreme right.
   1. Delete the + sign
   2. Skip to extreme LEFT and Invert :
      N → 1
      Z → 0

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

1	0	0	1							

1. If + is on the extreme right.
   1. Delete the + sign
   2. Skip to extreme LEFT and Invert :
      N → 1
      Z → 0

TM as Transducers :
6. Addition of Binaries

- Given an input string as binary number + binary number,
- Write the algorithm to do the addition

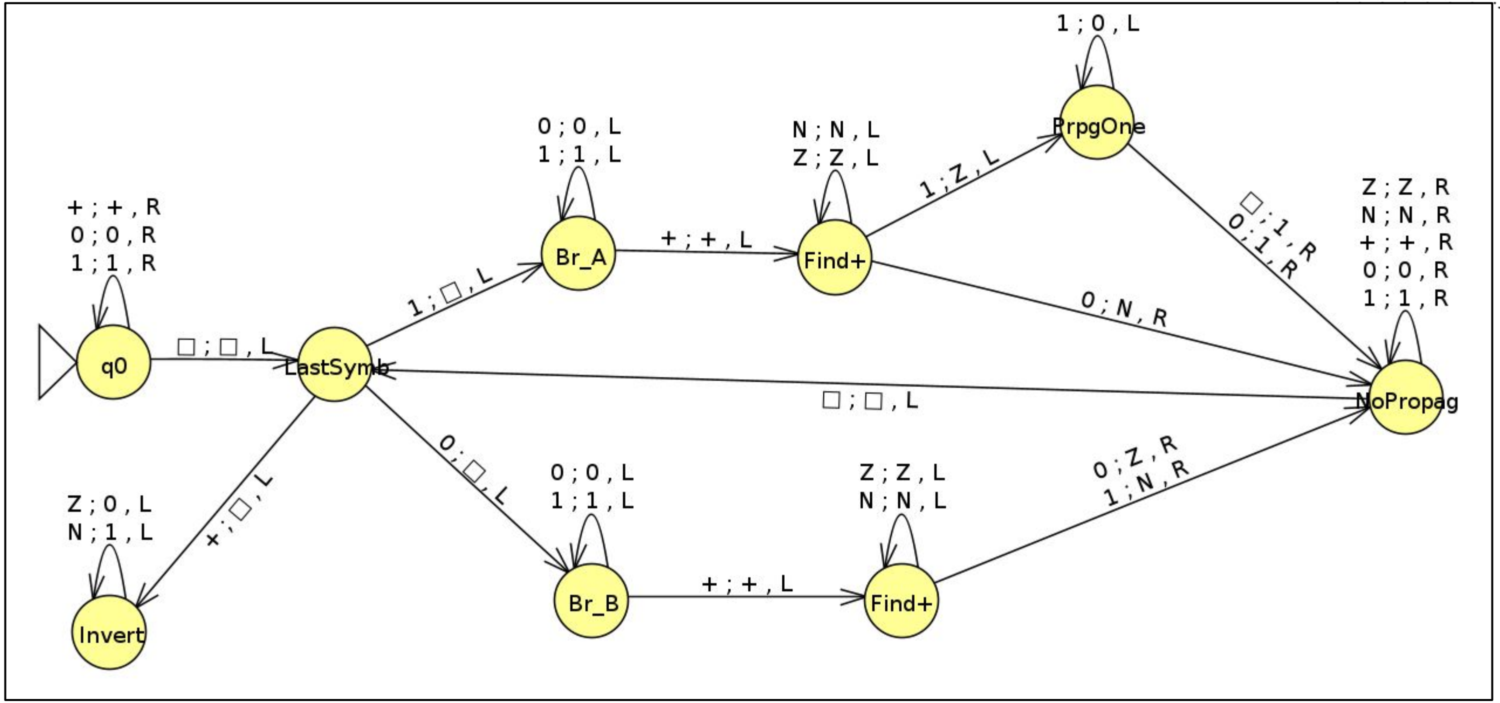
1	0	0	1							

1. If + is on the extreme right.
   1. Delete the + sign
   2. Skip to extreme LEFT and Invert :

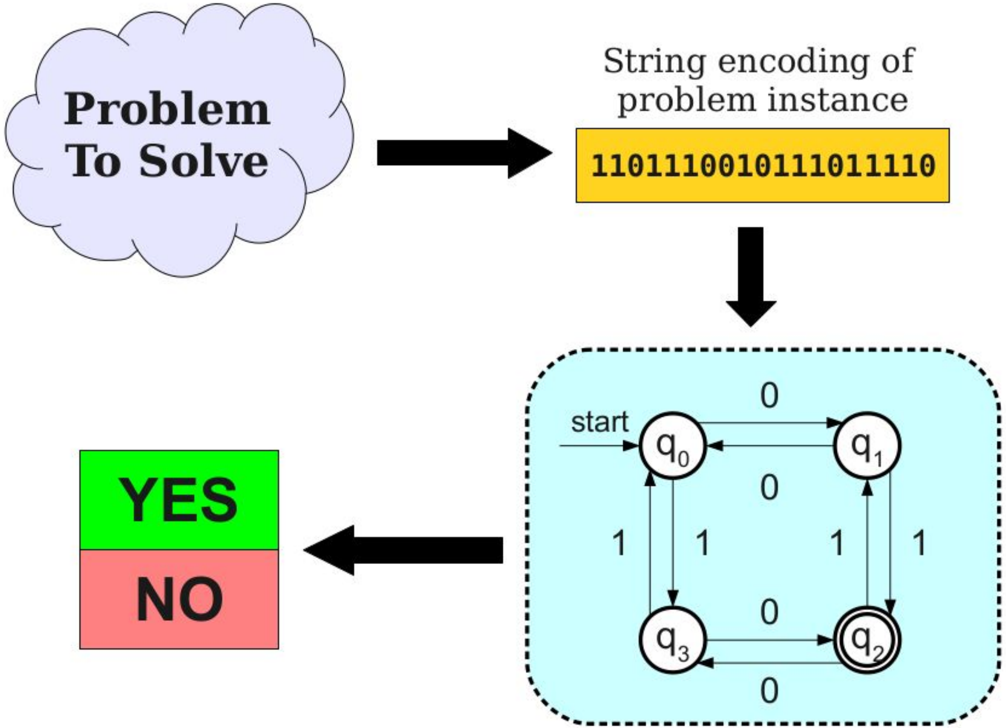
      N → 1

      Z → 0

TM as Transducers : 6. Addition of Binaries

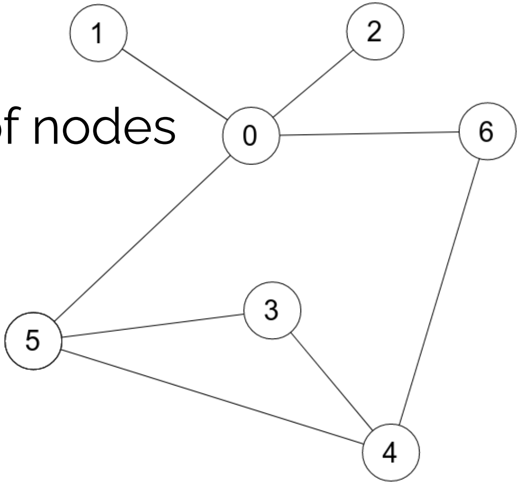


Turing Machine as a Problem Solver



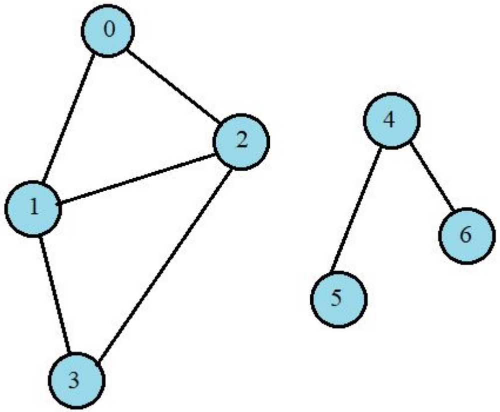
Turing Machine as a Problem Solver

- Given a graph with some nodes and links between the nodes
- The problem : **is the graph connected ?**
  - There is a link or path between any pair of nodes



Turing Machine as a Problem Solver

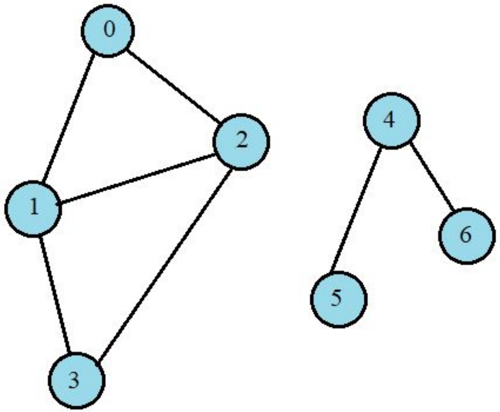
- Given a graph with some nodes and links between the nodes
- The problem : **is the graph connected ?**
  - There is a link or path between any pair of nodes



Turing Machine as a Problem Solver

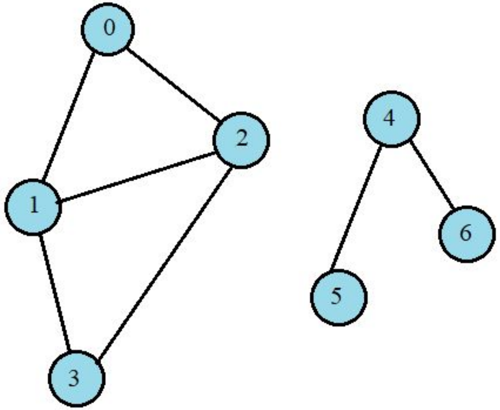
- Given a graph with some nodes and links between the nodes
- The problem : **is the graph connected ?**
- How to construct Turing Machine to answer such

  Question ?



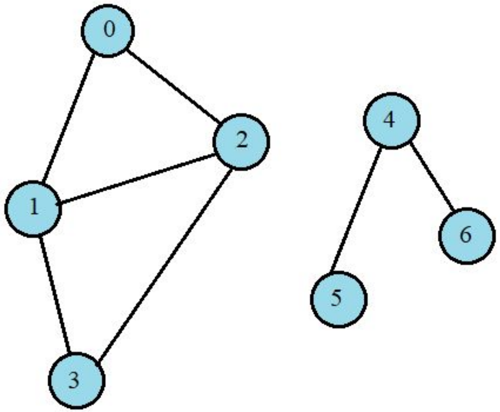
Turing Machine as a Problem Solver

- Given a graph with some nodes and links between the nodes
- The problem : **is the graph connected ?**
- How to construct Turing Machine to answer such Question ?
  - **Important : FIRST how to encode the problem ?**



Turing Machine as a Problem Solver

- Given a graph G
    - Its encoding can denoted as : <G>
    - <G>= ?

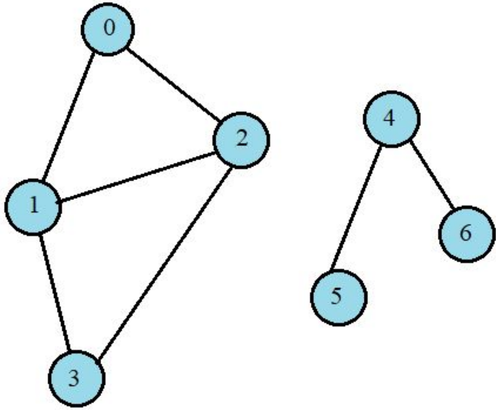


Turing Machine as a Problem Solver

- Given a graph G
  - Its encoding can denoted as : <G>
  - <G>= <(0,1,2,3,4,5,6)((0,1),(0,2),(1,2),(1,3),(2,3),(4,5),(4,6))>

List of nodes

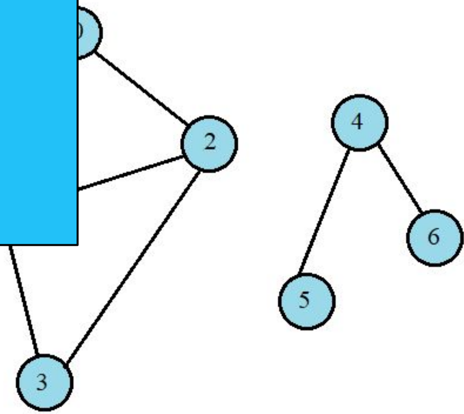
List of edges



Turing Machine as a Problem Solver

- Given a graph G
  - Its enco
  - <G>= <((

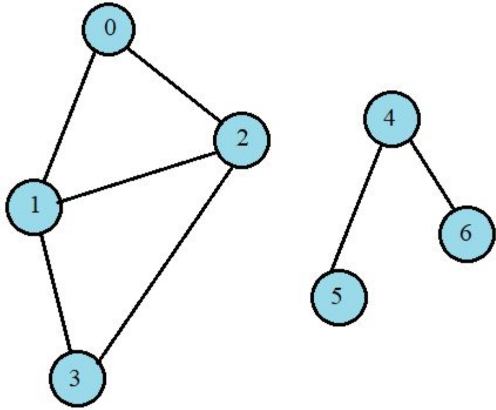
For the representation of numbers, feel free to any use any format:
Unary (1111), binary (0101), decimal (1,2,3), hexadecimal (A) depending on the alphabet.



Turing Machine as a Problem Solver

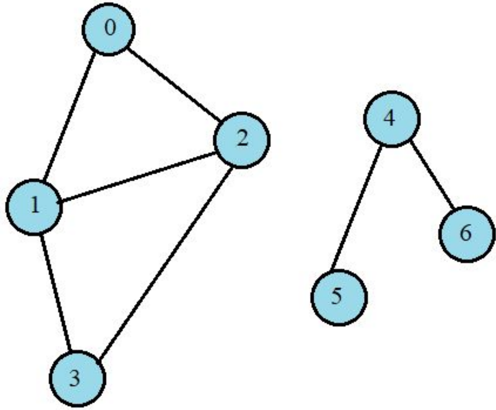
- Given a graph G
    - Its encoding can denoted as : <G>
    - <G>= <(0,1,2,3,4,5,6)((0,1),(0,2),(1,2),(1,3),(2,3),(4,5),(4,6))>
- Language of the connected graphs:

  L = { <G> | G is a connected graph }



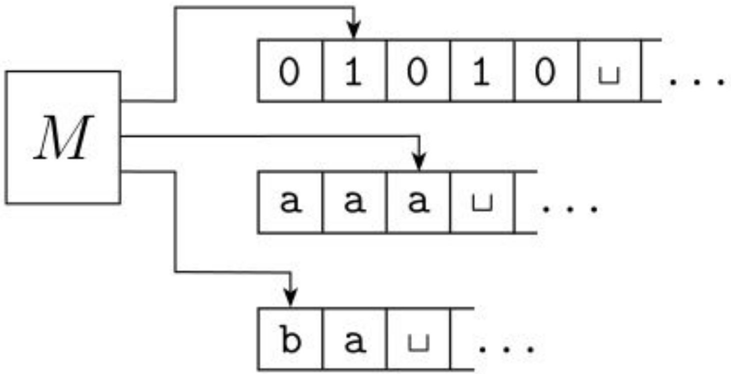
Turing Machine as a Problem Solver

- Construction of the Turing Machine
- Better to use Subroutines:
    - Verify the encoding format
    - Verify that there is no repetition in the first part
    - Check that edges contain only existing nodes
    - Check the links between nodes



Variations of Turing Machine

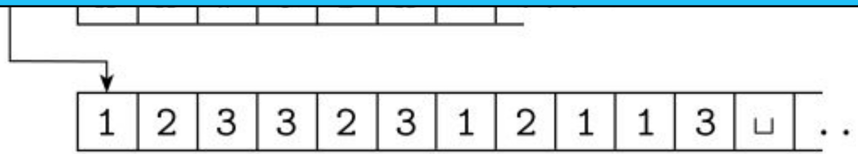
- **Multitape Turing Machine:**
    - The control can access multiple tapes reading, writing and moving the head of each tape at each step.



Variations of Turing Machine

- **Multitape Turing Machine:**
  - T ... ting and
    m ...

It has the same expressive power as a single tape ! Though, it may be easier and efficient in terms of constructing algorithms.
Take the binary addition as an example



Variations of Turing Machine

- **Nondeterministic Turing Machine:**
  - Where at a given state, there can be many possible transitions for the same symbol being read.
  - The machine would keep track of all actions in the same way as an NFA.
  - Every non-deterministic TM, there is an equivalent deterministic Turing machine → Nondeterministic TM has the same expressive power as a normal turing machine.

Universal Turing Machine

- You write a programming code in new Language **X** , it does some computation and return some input.

Universal Turing Machine

- You write a programming code in new Language **X** , it does some computation and return some input.
- You need to compile and execute the program written in X by compiler written in which language ?

Universal Turing Machine

- Given any Turing Machine **M** to conduct some computation, there is the Universal Turing Machine to simulate or execute M for any given input *w*
- The universal Turing Machine :
  - *Halt iff M halts on input w.*
  - *If M is a deciding/semi-deciding machine, then*
    - *If M accepts, accept.*
    - *If M rejects, reject.*
  - *If M computes a function, then U (<M, w>) must equal M (w)*

Universal Turing Machine

- We can construct a universal TM that accepts the language

  $L = \{\langle M, w\rangle \mid M \text{ is a TM and } w \in L(M)\}$

Church-Turing Thesis

- Turing Machines are an abstract model of computation, their purpose is to define in a mathematical way what problems are theoretically computable and which are not.
- In 1900, mathematician David Hilbert identified **23 mathematical problems** and posed them as a challenge for the coming century.
- Hilbert's tenth problem was to devise an **algorithm** that tests whether a **polynomial** has an integral root. (*Integers to be assigned to the polynomial variables to reach a value of zero*)

Church-Turing Thesis

- Hilbert did not use the term algorithm but rather “a process according to which it can be determined by a finite number of operations.”
  - He assumed that such an algorithm must exist—someone need only find it.
  - But : it is algorithmically unsolvable.
- Church–Turing thesis provides the definition of algorithm necessary to resolve Hilbert’s tenth problem

Church-Turing Thesis

- Church-Turing thesis. There is an “effective procedure” for a problem if and only if there is a TM for the problem.

Turing-Complete Systems

- Are there models of computation more powerful than Turing machines?

Turing-Complete Systems

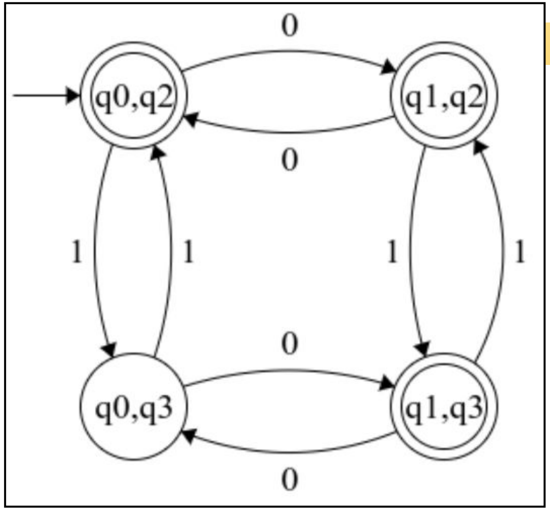
- Are there models of computation more powerful than Turing machines?
  - We do not know if there are more powerful models.
  - However, there are many computational models equivalent in power to TM's. They are called **Turing-complete systems**.
- A system of data-manipulation rules is said to be complete if it can be used to simulate any Turing machine

Turing-Complete Systems

- Modern computers
- Church's lambda calculus.
- Gödel's μ-recursive functions
- Lindenmayer systems or L-systems.
- …

TD8 - Solutions

Convert the following NFA to RegEx



TD8 - Solutions

Produce the context free grammar for the following language :
The number of b + 2 = the number of a. The number of a is more than b but with strictly only two letters.
Example of words in the language : aa , aaba, baaa, aabbaa, baaaab,....

TD8 - Solutions

Let D = {xy|x, y ∈ {0,1}* and |x| = |y| but x ≠ y}. Show that D is a context-free language.

TD8 - Solutions

Use the pumping lemma to show that the following languages are not context free.

$\{0^n 1^n 0^n 1^n \mid n \geq 0\}$

TD8 - Solutions

Use the pumping lemma to show that the following languages are not context free

$\{0^n \#0^{2n} \#0^{3n} \mid n \geq 0\}$

TD8 - Solutions

Let B be the language of all palindromes over {0,1} containing equal numbers of 0s and 1s. Show that B is not context free.

TD8 - Solutions

Prove that $L = \{a^n \mid n \text{ is prime}\}$ is not CFL.

TD8 - Solutions

Over alphabet {0,1}, Produce the RegExs for the languages :

1. L whose words do not contain the substring 101
Easy and non-brainy way -> DFA for the complement -> convert to RegEx
Hard way :
Enumerate possible strings that we need to accept …

11111 → 1*

00000 → 0*

111100000 → 1*0*

0000011111 → 0*1*

001110000111000 → {000*,111*}*

011111000001 → ...

Look for more possible words and later

optimize

TD8 - Solutions

Over alphabet {0,1}, Produce the RegExs for the languages :
L which does not contain the string 101