Theory of Computing :

11. Decidability

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

Outline :

- Algorithms and Church's Thesis
- Decidable Languages & Examples
- Diagonalization Method & Infinities
- Proving by Reducibility
- Undecidability of the Halting Problem
- Post Correspondence Problem

There is a solution to every problem ?

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 **U** :
  - *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
  $L = \{\langle M, w \rangle \mid M \text{ is a TM and } w \in L(M)\}$
  - Given an **encoded representation** of the turing machine M with an input string w
  - The word w is accepted by the language represented by the turing machine M

Universal Turing Machine

- We can construct a universal TM that accepts the language

  L = {<

The notation < M, w > : The encoded representation of :

Turing Machine accepting the word w

  - C … e M with

    a
  - The word w is accepted by the language represented by the turing machine M

Algorithms and Church's 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*)

Algorithms and Church's 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

Algorithms and Church's Thesis

- Church-Turing thesis. There is an "effective procedure" for a problem if and only if there is a TM for the problem.
    - For a given problem, if we can write the algorithm or construct the turing machine, it is computable or algorithmically solvable

Algorithms and Church's Thesis

- Church-Turing thesis. There is an "effective procedure" for a problem if and only if there is a TM for the problem.
  - For a given problem, if we can write the algorithm or construct the turing machine, it is computable or algorithmically solvable
  - We are speaking always about the general cases, as there are unsolvable/uncomputable problems, but there are **instances** of that problem that we **can solve**.

Algorithms and Church's Thesis

- What's an Algorithm :
  - An algorithm is an effective/systematic/mechanical method for achieving the desired result for a given problem.
    - The desired result can be :
      - Decisional : Yes or No Answer
      - Computational : Conduct some computational and outputs the result.

Algorithms and Church's Thesis

- What's an Algorithm :
  - An algorithm is an effective/systematic/mechanical method for achieving the desired result for a given problem.
    - It has a **finite** number of instructions.
    - If carried out without error, it produces the desired result in a finite number of steps.
    - It can be carried out by a human with only paper and pen ?

Algorithms and Church's Thesis

- What's an Algorithm :
  - An algorithm is an effective/systematic/mechanical method for achi

    It should work for all input instances from a given domain

    -
    - t in a finite number of steps.
    - It can be carried out by a human with only paper and pen ?
    - It should work for all input instances from a given domain

Algorithms and Church's Thesis

- What's an Algorithm :
  - An algorithm is an effective/systematic/mechanical method for ach

    ■

    ■ ult in a finite number of steps.

    ■ It can be carried out by a human with only paper and pen ?

Hypothesis : Any algorithm can be represented and executed on a Turing Machine

Decidable Languages & Examples

- The collection of strings that M **accept**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

Decidable Languages & Examples

- **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.

Decidable Languages & Examples

- **Problem** : Given a DFA called B, and given another word *w* . is the Acceptance problem decidable ?
    - Algorithmically Computable
    - Halts with either : Accept or Reject.

- *Remember : General case = ALL instances of problems*

Decidable Languages & Examples

- **Problem** : Given a DFA called B, and given another word *w*, is the Acceptance problem decidable ?
    - Computable
    - Halts with either : Accept or Reject.
- The problem is expressed as a Language:

$$A_{DFA} = \{ < B, w > \mid B \text{ is a DFA that accepts input string } w \}$$

Decidable Languages & Examples

- **Problem** : Given a DFA called B, and given another word *w*, is the Accep
  - Co
  - Ha
- The pr

Showing that the language is decidable is the same as showing that the computational problem is decidable

$A_{DFA} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w \}$

Decidable Languages & Examples

- **Problem** : Given a DFA called B, and given another word *w*, is the Acceptance problem decidable ?
- We simply need to present a Turing Machine M that decides A_{DFA} .
  - M =
    - *On input < B, w > , where B is a DFA and w is a string:*
    - *Simulate B on input w.*
    - *If the simulation ends in an accept state, "**accept**" . If it ends in a non-accepting state, "**reject**"*

Decidable Languages & Examples

- **Problem** : Given a DFA called B, and given another word *w*, is the Acceptance problem decidable ?
- Steps for the Turing Machine M
    1. *Verification of Input Syntax for the encoded string < B, w > such that B can be represented as a list containing 5 elements Q, Σ, δ, q0 , and F*
    2. *Store the initial state on the memory tape at some position that you can return easily*
    3. *Start reading the symbols of the word w and for each symbol you read, find the relevant transitions.*
    4. *Update the current state*
    5. *When you complete reading all symbols of the word w, examine if the current state is accepting → Accept, otherwise, Reject.*

Decidable Languages & Examples

- **Problem** : Given a DFA called B, and given another word *w*, is the Acceptance problem decidable ?
- Steps f

The better encoding you propose for the problem, the easier it will be for constructing the TM

1. *Ver ... e rep*
2. *Sto ... rn easily*
3. *Start reading the symbols of the word w and for each symbol you read, find the relevant transitions.*
4. *Update the current state*
5. *When you complete reading all symbols of the word w, examine if the current state is accepting → Accept, otherwise, Reject.*

Decidable Languages & Examples

- **Problem** : Given a DFA called B, and given another word *w*, is the Acceptance problem decidable ?
  - A_{DFA} is a decidable language.

Decidable Languages & Examples

- **Problem** : Given a NFA called B, and given another word *w*, is the Acceptance problem decidable ?

Decidable Languages & Examples

- **Problem** : Given a NFA called B, and given another word *w*, is the Acceptance problem decidable ?
    - We convert the NFA to DFA
    - We follow the same procedure described for the DFA to arrive that the acceptance problem for the NFA is computable

Decidable Languages & Examples

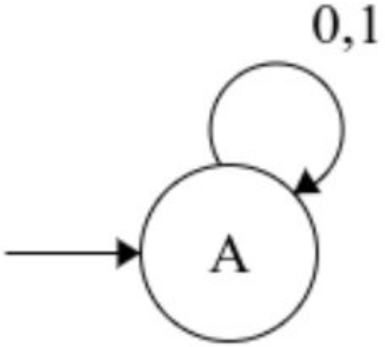
- **Problem** : Given a NFA called B, and given another word *w*, is the Acceptance problem decidable ?
  - We convert the NFA to DFA
  - We follow the same procedure described for the DFA to arrive that the acceptance problem for the NFA is computable
- **Same Problem** : Can we generate such a string w using a given regular expression : **Generation Problem ?**

Decidable Languages & Examples

- **Problem** : Given a DFA for the empty set, is it computable for any given string ? it is decidable ? "***Emptiness testing***"

Decidable Languages & Examples

- **Problem** : Given a DFA for the empty set, is it computable for any given string ? it is decidable ?

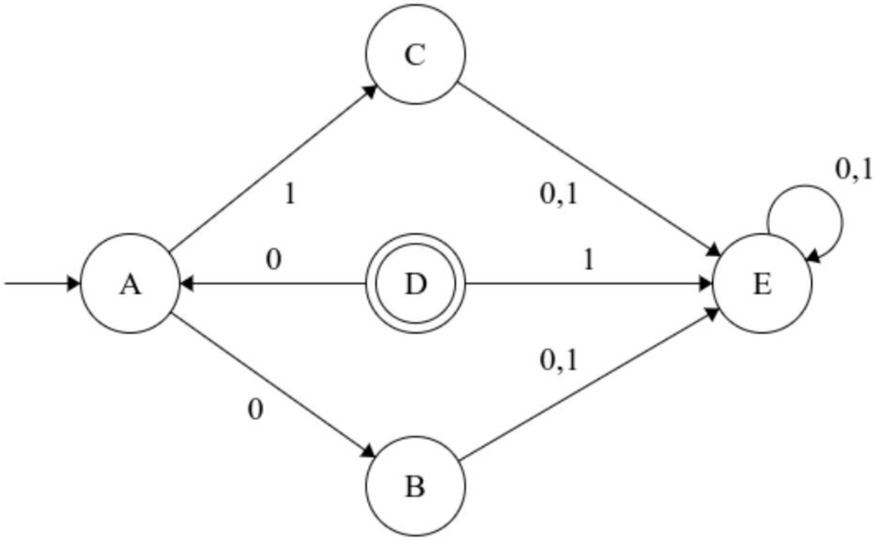


An SINGLE Instance of the problem is the shown DFA along with the word 101

Decidable Languages & Examples

- **Problem** : Given a DFA for the empty set, is it computable for any given string ? it is decidable ?

Does this instance of the DFA represents the language of the empty set



Decidable Languages & Examples

- **Problem** : Given a DFA for the empty set, is it computable for any given string ? it is decidable ?
- The problem is represented as a Language $\mathbf{E_{DFA}}$

E_{DFA} = {< A > | A is a DFA and L(A) = ø }.

Decidable Languages & Examples

- **Problem** : Given a DFA for the empty set, is it computable for any given string ? it is decidable ?
- The problem is represented as a Language $\mathbf{E_{DFA}}$

  E_{DFA} = {< A > | A is a DFA and L(A) = ∅ }.
- *A DFA accepts some string iff reaching an accept state from the start state by traveling along the arrows of the DFA is possible*

Decidable Languages & Examples

- **Problem** : Given a DFA for the empty set, is it computable for any given string ? it is decidable ?
- We simply need to present a Turing Machine T that decides $\mathbf{E_{DFA}}$.
  - T = On input <A>, where A is a DFA
    1. *Mark the start state of A.*
    2. *Repeat until no new states get marked:*
       - *Mark any state that has a transition coming into it from any state that is already marked*
    3. *If no accept state is marked, accept ; otherwise, reject ."*

Decidable Languages Examples

Same Algorithm as the Problem of Connected Graphs

- **Problem** : Given a DFA for the empty set, is it computable for any given string ? it is decidable ?
- We simply need to present a Turing Machine T that decides $\mathbf{E_{DFA}}$.
    - T = On input <A>, where A is a DFA
        1. *Mark the start state of A.*
        2. *Repeat until no new states get marked:*
            - *Mark any state that has a transition coming into it from any state that is already marked*
        3. *If no accept state is marked, accept ; otherwise, reject ."*

Decidable Languages & Examples

- **Problem** : Given two DFAs, can we algorithmically compute or decide if they are equivalent ?

Decidable Languages & Examples

- **Problem** : Given two DFAs, can we algorithmically compute or decide if they are equivalent ?
- The problem whether two DFAs recognize the same language is decidable, is represented as a language EQ_{DFA} such that

$$EQ_{DFA} = \{< A, B > \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}.$$

Decidable Languages & Examples

- **Problem** : Given two DFAs, can we algorithmically compute or decide if they are equivalent ?
- We construct a new DFA C from A and B, where C accepts only those strings that are accepted by **either** A **or** B but **not by both**
  - **If A equals B , it implies that C is nothing.**

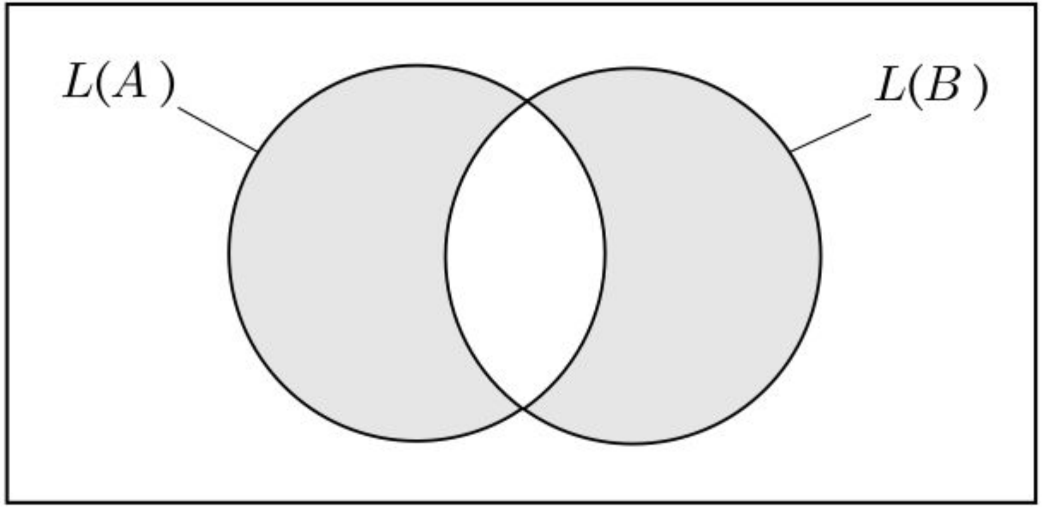
Decidable Languages & Examples

- **Problem** : Given two DFAs, can we algorithmically compute or decide if they are equivalent ?
- We construct a new DFA C from A and B, where C accepts only those strings that are accepted by **either** A **or** B but **not by both**
  - **If A equals B , it implies that C represents the empty set.**
- Language for C can be represented as:

$$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$$

Decidable Languages & Examples

- **Problem** : pute or decide if they are
- We constr pts only those strings tha **th**
  - **If A ec pty set.**



- Language for C can be represented as:

$$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$$

Decidable Languages & Examples

- **Problem** : Given two DFAs, can we algorithmically compute or decide if they are equivalent ?
- Language for C can be represented as:

$$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$$

- The construction of C using Union, Complement , Intersection are algorithms that can be carried out by Turing machines
- *Provided that we prove that such constructions are closed under the class of regular languages…*

Decidable Languages & Examples

- **Problem** : Given two DFAs, can we algorithmically compute or decide if they are equivalent ?
- Language fo

$$L(C \quad \cap L(B)\Big)$$

To prove that L(C) is the empty set, use the case discussed earlier

- The construction of C using Union, Complement , Intersection are algorithms that can be carried out by Turing machines

Decidable Languages & Examples

- Other problems for CFG:
  - Is generation of words using Context-Free Grammar decidable ?
  - Is the equivalence of two CFG computable ?
  - Is the emptiness testing for a language represented by CFG computable ?

Diagonalization Method & Infinities

- For Turing Machines : is the acceptance problem decidable for During Machine ?
    - $A_{TM} = \{ < M, w > \mid M \text{ is a TM and } M \text{ accepts } w \}$.

Diagonalization Method & Infinities

- For Turing Machines : is the acceptance problem decidable for Turing Machine ?
  - A_{TM} = { < M, w > | M is a TM and M accepts w } .

- A_{TM} is **undecidable**

Diagonalization Method & Infinities

- For ... for During
  Ma...
  - ...
- AT...

Another different Question:
Can we represent any problem or any language using Turing Machine ?

Diagonalization Method & Infinities

- Comparing two sets:
  - If two sets of finite size,
    - For instance : {a,b,c,d } and { 1, 2, 3, 4} :
  - Question :
    - How to determine that they have the same size

(**Assume we don't know how to count**)

Diagonalization Method & Infinities

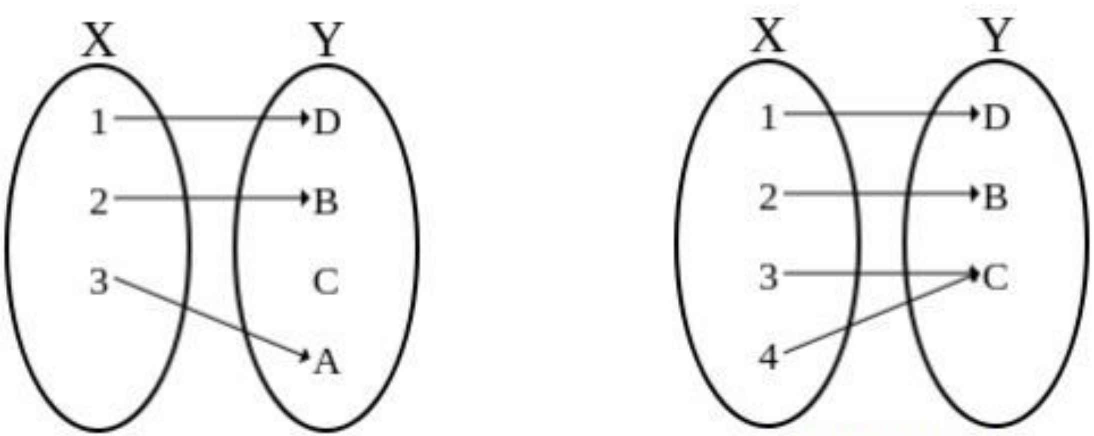
- Comparing two sets:
  - If two sets of finite size,
    - For instance : A={a,b,c,d } and B={ 1, 2, 3, 4} :
  - Answer :
    - For each element *x* in A, find a mapping or correspondence y in B such that y is mapped only by x

Diagonalization Method & Infinities

- Definitions:
  - **One-to-One function** : A function ***f*** which maps two sets: A to B.
    - If it never maps two different elements to the same place.
    - That is, if $f(a) \neq f(b)$ whenever $a \neq b$.
  - Other terminology : **injective**

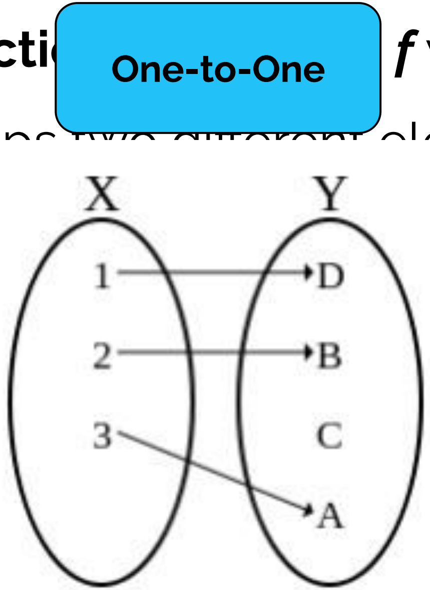
Diagonalization Method & Infinities

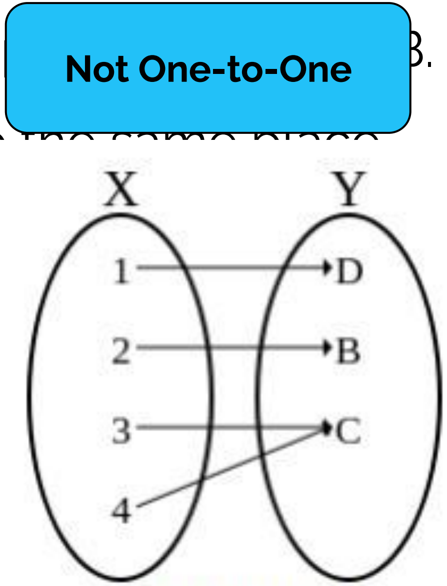
- Definitions:
    - **One-to-One function** : A function ***f*** which maps two sets: A to B.
        - If it never maps two different elements to the same place
        - That is, if f
    - Other termino



Diagonalization Method & Infinities

- Definitions:
  - **One-to-One functi** *f* which ma 3.
    - If it never maps two different elements to the same place
    - That is, if f
  - Other termino





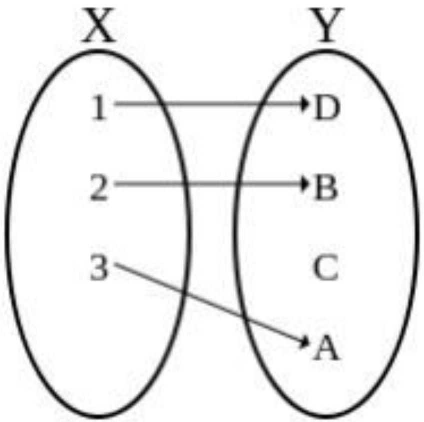
Diagonalization Method & Infinities

- Definitions:
  - **Onto Function :** A function ***f*** which maps two sets: A to B
    - If it hits every element of B
    - That is, if for every $b \in B$, there is an $a \in A$ such that $f(a) = b$.
  - Other terminology : **surjective**

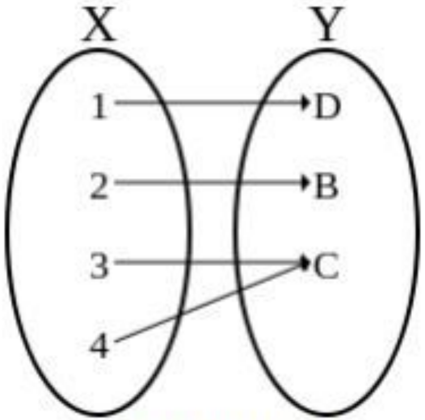
Diagonalization Method & Infinities

- Definitions:
  - **Onto Function :** A f … h maps two set
    - If it hits every element of B
    - That is, if f
  - Other termino

Not Onto



Onto



Diagonalization Method & Infinities

- Definitions:
  - **Correspondence Function** is both **one-to-one** and **onto.**
  - Two sets have the same size if there is a correspondence function f : A→B.
    - Every element of A maps to a unique element of B and each element of B has a unique element of A mapping to it.
  - A correspondence is simply a way of pairing the elements of A with the elements of B.
  - Other terminology : **bijective**

Diagonalization Method & Infinities

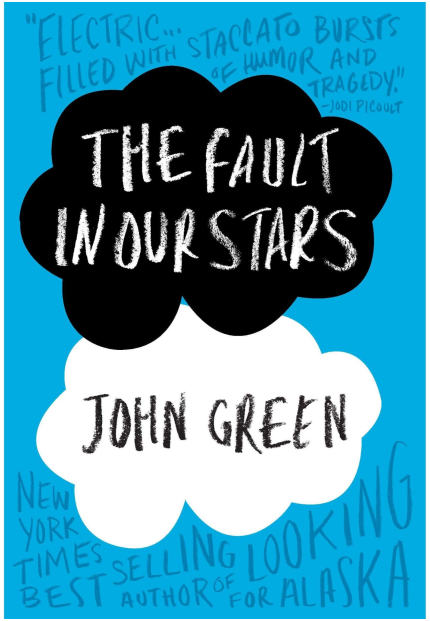
- Given now an infinite set :
  - Example : Natural numbers {0,1,2,3,..}
- Question:
  - Can we count the elements in such set ?

Diagonalization Method & Infinities

- Given two infinite sets :
  - Example :
    - Natural numbers {1,2,3,..}
    - Odd numbers {1,3,5..}
- Question:
  - Can an infinite set be bigger than another infinite set ?

Diagonalization Method & Infinities

Reading the **Novel** written by John Green : *"There are infinite numbers between 0 and 1. There's .1 and .12 and .112 and an infinite collection of others. Of course, there is a bigger infinite set of numbers between 0 and 2, or between 0 and a million. Some infinities are bigger than other infinities... I cannot tell you how grateful I am for our little infinity. You gave me forever within the numbered days, and I'm grateful. "*



Diagonalization Method & Infinities

- **Cantor Definition** :
  - A set A is countable if either it is finite or it has the same size as N (N is the set of natural numbers)

Diagonalization Method & Infinities

- **Cantor Definition** :
  - A set A is countable if either it is finite or it has the same size as N (N is the set of natural numbers)
    - There is a **correspondence function** between the set of natural numbers and the set A.
  - *Georg Cantor (1848-1918) is a german mathematician. He was interested in the problem of measuring the sizes of infinite sets*

Diagonalization Method & Infinities

- **Cantor Definition** :
    - A set A is countable if either it is finite or it has the same size as N (

If we can enumerate the elements in the set , it is countable

-
    - *Georg Cantor (1848-1918) is a german mathematician. He was interested in the problem of measuring the sizes of infinite sets*

Diagonalization Method & Infinities

- **Question** : is the set of even numbers **countable ?** or does it have the same size as N

Diagonalization Method & Infinities

- **Question** : is the set of even numbers **countable ?** or does it have the same size as N
- **Answer** :
  - Let :
    - N be the set of natural numbers {1, 2, 3, . . .}
    - E be the set of even natural numbers {2, 4, 6, . . .}.
  - Using Cantor's definition of size, there is a correspondence function **f(n)=2n**

Diagonalization Method & Infinities

- **Question** : is the set of even numbers **countable ?** or does it have the same size as N

This is a simple visualization for the correspondence function

-
    - al numbers {1, 2
        - E be the set of even natural number
    - Using Cantor's definition of size, there is
      function **f(n)=2n**

n	$f(n)$
1	2
2	4
3	6
$\vdots$	$\vdots$

Diagonalization Method & Infinities

- **Question** : is the set of even numbers **countable ?** or does it have the same size as N
-

It may seem strange that even E is proper subset of N, they have the same size, but as long we can pair any member from N to E, they have the same size

	(n)
	2
	4
	6
$\vdots$	$\vdots$

  - Using Cantor's definition of size, there is
    function **f(n)=2n**

Diagonalization Method & Infinities

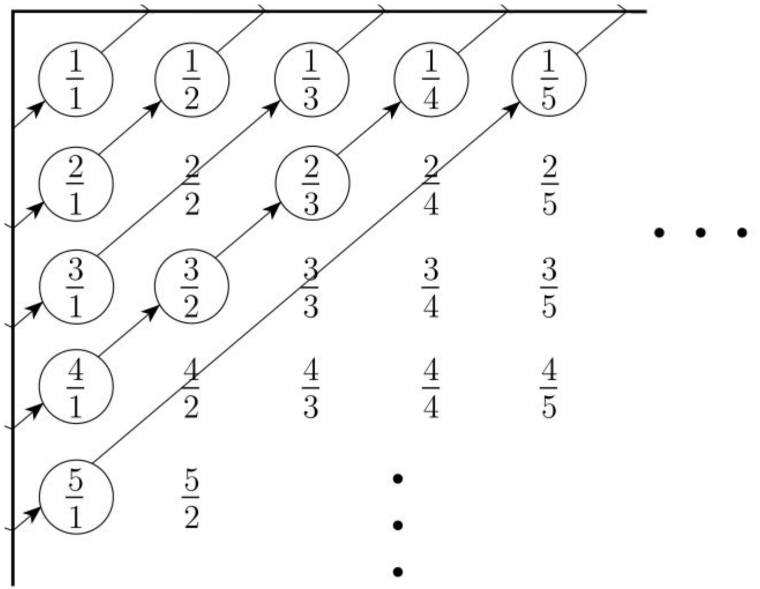
- **Question** : is the set of rational numbers **countable ?** or does it have the same size as N

$$\mathcal{Q} = \{\tfrac{m}{n} \mid m, n \in \mathcal{N}\}$$

  - Can we find a correspondence function between them ?

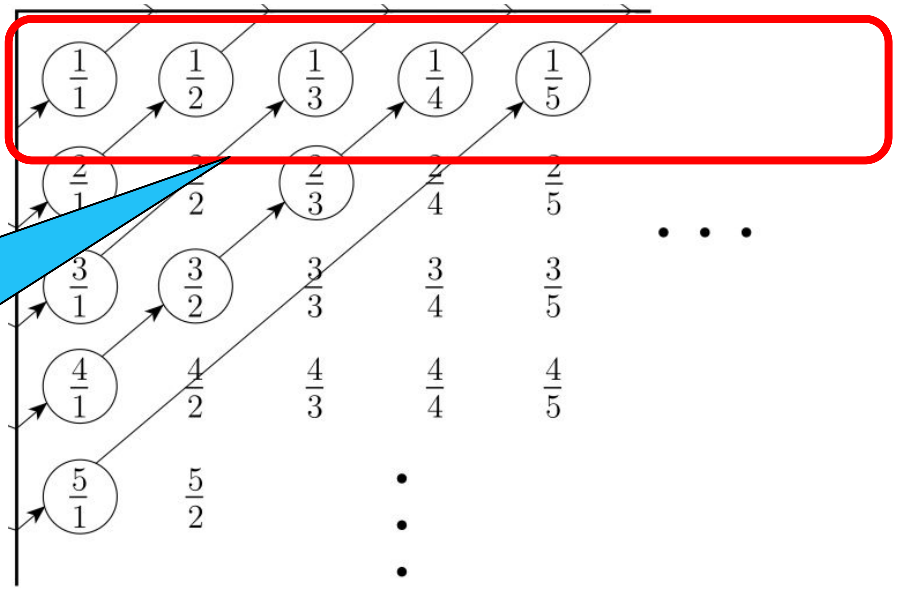
Diagonalization Method & Infinities

- **Question** : is the set of rational numbers **countable ?** or does it have the same size as N



Diagonalization Method & Infinities

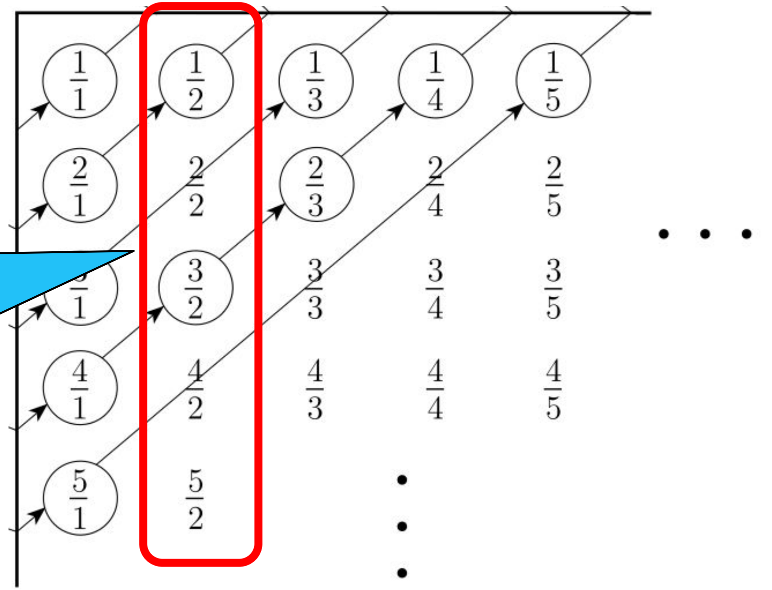
- **Question** : is the set of rational numbers **countable ?** or does it have the same size as N



Each row corresponds to a given number placed as a numerator

Diagonalization Method & Infinities

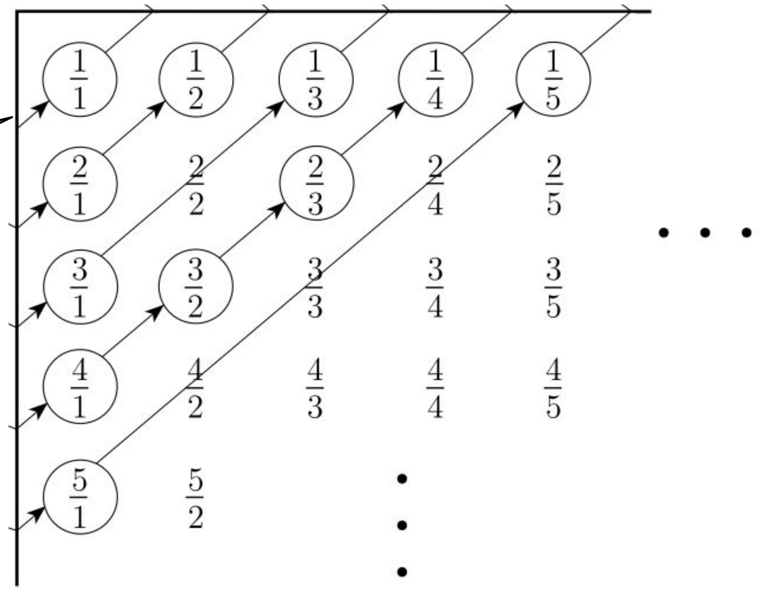
- **Question** : is the set of rational numbers **countable ?** or does it have the same size as N



Each column corresponds to a given number placed as a denominator

Diagonalization Method & Infinities

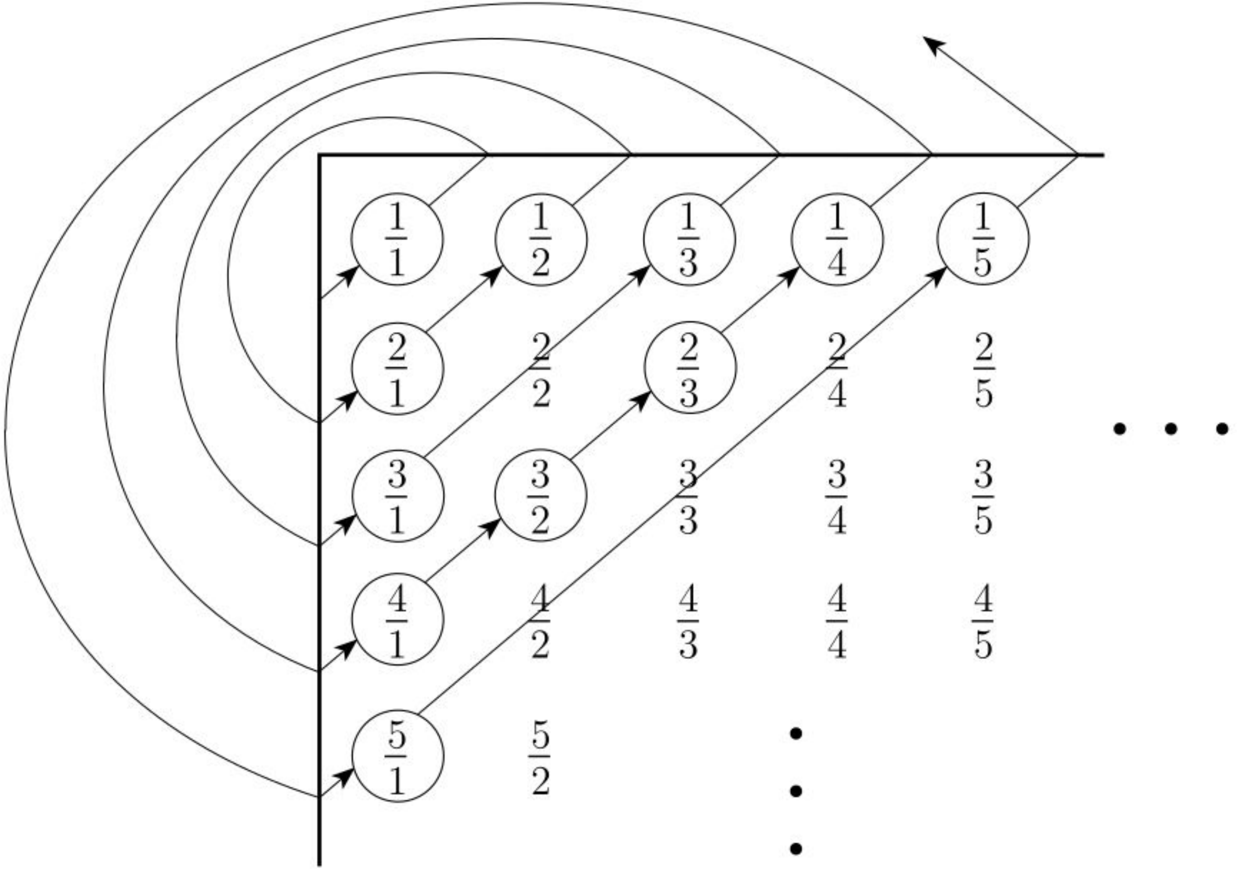
- **Question** : is the set of rational numbers **countable ?** or does it have the same size as N



This way, ALL rational numbers are represented. Can we enumerate them ?

Diagonalization Method & Infinities

n	f(n)
1	1/1=1
2	2/1=2
3	1/2
4	3/1=3
5	1/3
..	..



Diagonalization Method & Infinities

- **Question** : is the set R of real numbers **countable ?** or does it have the same size as N
    - Find a correspondence function
    - Same as Rational Number ?

Diagonalization Method & Infinities

- **Diagonalization** :
  - Given the following list of words,
    - Give a word not from the list.

?	?	?	?	?

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Diagonalization** :
  - Given the following list of words,
    - Give a word not from the list.
      - Hello ?

H	E	L	L	O

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Diagonalization** :
  - Given the following list of words,
    - Give a word not from the list.
    - What's the **Algorithm ?**

?	?	?	?	?

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Diagonalization** :
  - Algorithm
    - Sequentially:
      - For each letter from the diagonal, choose a different letter, Possibly the next next letter
        - Example :

          at step 1: we choose R

R	**?**	**?**	**?**	**?**

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Diagonalization** :
  - Algorithm

onal,

sibly

Will our new word (R????) be the same as the first word : QUIET ?
Never,

Because they have different first letter

R	?	?	?	?

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

at step 1: we choose R

Diagonalization Method & Infinities

- **Diagonalization** :
  - Algorithm
    - Sequentially:
      - For each letter from the diagonal, choose a different letter, Possibly the next next letter
        - Example :

          at step 2: we choose U

R	**U**	**?**	**?**	**?**

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Diagonalization** :
  - Algorithm

Will our new word (R**U**???) be the same as the second word : S**T**ONE ?
Never,

Because they have different second letter

onal,

sibly

at step 2: we choose U

RUGBY

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Diagonalization** :
  - Algorithm

At step n, Will our new word (RTAN…) be the same as the n^{th} word
Never,

Because they have different n^{th} letter

onal,

sibly

at step 2: we choose U

R	**U**	**?**	**?**	**?**

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Diagonalization** :
  - This table for only 5 words ? does this method of “diagonalization” work for an infinite set ?

R	U	G	B	Y

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Diagonalization** :
  - This table for only 5 words ? does this method of “diagonalization” work for an infinite set ?
    - Each **ROW** corresponds to an existing word, we ensure to change the symbol for the constructed new word ⇒ therefore, it would never be in the list

R	U	G	B	Y

1	Q	U	I	E	T
2	S	T	O	N	E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Diagonalization** :
  - This table for only 5 words ? does this

The new word cannot be on the list: it is different from first word in first letter, different from second word in second letter, etc.

  - It works : Each column corresponds to an existing word, we ensure to change the symbol for the constructed new word ⇒ therefore, it would never be in the list

	R	U	G	B	Y
					T
					E
3	O	F	F	E	R
4	C	L	E	A	R
5	P	H	L	O	X

Diagonalization Method & Infinities

- **Question** : is the set R of real numbers **countable ?** or does it have the same size as N
  - Let's **assume that** we can enumerate all real numbers

1	.	1	0	0	0	...
2	.	1	1	0	0	..
3	.	2	1	2	3	...
4	.	3	1	4	3	...
5	.	..	..	..	...	..
..	.	..	..	..	..	...

Diagonalization Method & Infinities

- **Question** : is the set R of real numbers **countable ?** or does it have the same size as N
  - Let's **assume that** we can **enumerate all** real numbers
  - Using the diagonalization method:
    - We can always generate new **real** numbers that are not inside the table

1	.	1	0	0	0	...
2	.	1	1	0	0	..
3	.	2	1	2	3	...
4	.	3	1	4	3	...
5	.	..	..	..	...	..
..	.	..	..	..	..	...

Diagonalization Method & Infinities

- **Question** : is the set R of real numbers **countable ?** or does it have the same size as N
  - Let's **assume that** we can **enumerate all** real numbers
  - Using the diagonalization method:

.4357 ...

real

e table

1	.	**1**	0	0	0	...
2	.	1	**1**	0	0	..
3	.	2	1	**2**	3	...
4	.	3	1	4	**3**	...
5	.	..	..	..	...	..
..	.	..	..	..	..	...

Diagonalization Method & Infinities

- **Question** : is the set R of real numbers **countable ?** or does it have the same size as N

The set of real numbers R is uncountable

  - We can always generate new **real** numbers that are not inside the table

				0	0	…
2	.	1	1	0	0	..
3	.	2	1	2	3	…
4	.	3	1	4	3	…
5	.	..	..	..	…	..

Diagonalization Method & Infinities

- **Important Questions**
  - Question 1 : is the language of Turing Machine :
    - Finite or Infinite ?
    - Countable or Uncountable ?
  - Question 2 :
    - Can we solve all problems ? Are all problems computable ?
      Can all problems represented by Turing Machines ?

Diagonalization Method & Infinities

- **Important Questions**
  - Question 1 : is the language of Turing Machine :
    - It is infinite (You just keep adding states….)
    - But :
      - Each TM has :
        - Finite number of states
        - Finite Transitions
        - ⇒ Each TM can be encoded as a finite string which itself can be converted to a natural number.

Diagonalization Method & Infinities

- **Important Questions**
  - Question 1 : is the language of Turing Machine :
    - It is infinite
    - Another proof for TM as a countable and infinite set:
      - TMs can be represented/encoded as a string, therefore, it is a subset of the language $\Sigma*$
      - We can enumerate strings of length 1 { 0, 1}, then strings of length 2, then strings of length 3 and so on…

Diagonalization Method & Infinities

- **Important Questions**
  - Question 1 : is the language of Turing Machine :

The number of Turing Machines is infinite and countable

- Finite number of states
- Finite Transitions
- ⇒ Each TM can be encoded as a finite string which itself can be converted to a natural number.

Diagonalization Method & Infinities

- **Important Questions**
  - Question 2 :
    - Can we solve all problems ? Are all problems computable ?

      Can all problems represented by Turing Machines ?
    - All problems = All languages
      - The set of all languages can be represented as $P(\Sigma*)$
      - ϕ , {1}, {0}, {1,11,111,...}, {0,00,000...},{10,100,1000,...} ....

Diagonalization Method & Infinities

- **Important Questions :** Can we solve all problems ? Are all problems computable ?
  - Let *B* be the set of all **infinite binary sequences**.
    - 0000110000 ….
    - 0101010000 ….
    - 111111111111 …
    - 101111110011 …
    - …

Diagonalization Method & Infinities

- **Important Questions :** Can we solve all problems ? Are all problems computable ?
  - Let ***B*** be the set of all **infinite binary sequences**. We can show that B is uncountable by using a **proof by diagonalization** using the same way used for real numbers.
  - Let ***L*** be the set of all languages over alphabet Σ
  - We need to prove that B (which is uncountable) has the same size as L in order to conclude that L is uncountable

Diagonalization Method & Infinities

L is the set of all languages

- **Important Questions :** Can we solve all problems ? Are all problems computable ?
  - **We need to find a correspondence function between B and L**
  - Let $\Sigma* = \{s_1, s_2, s_3, \ldots\}$ such that s_1, s_2, s_3 are just words from the language.
  - A is a language **A,** $\in$ L ,
    - A a unique sequence in B.
    - The i^{th} bit of sequence is a 1 if $s_i \in$ A and is a 0 if s_i not in A,

Diagonalization Method & Infinities

- **Important Questions :** Can we solve all problems ? Are all problems computable ?

$$
\begin{array}{lllllllllllll}
\Sigma^* & = \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \cdots & \} ; \\
A & = \{ & & 0, & & 00, & 01, & & & 000, & 001, & \cdots & \} ; \\
\chi_A & = & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots & .
\end{array}
$$

  - ε not in A ? → 0
  - 1 in A → 1
  - 00 not in A → 0
    - Until we create X_A which is the **characteristic sequence** of A

Diagonalization Method & Infinities

- **Impo**

  co

Σ^*
A
χ_A

We have created a correspondence function between L (Infinite set of all languages) and B (The infinite and uncountable set of infinite binary sequences)

  -
  -
  - 00 not in A → 0
    - Until we create X_A which is the **characteristic sequence** of A

Diagonalization Method & Infinities

- **Important Questions :** Can we solve all problems ? Are all problems computable ?

The set of all languages is infinite and uncountable
Vs.
The number of Turing Machines is infinite and countable

Diagonalization Method & Infinities

- **Important Questions :** Can we solve all problems ? Are all problems computable ?
  -

There are languages that are not recognizable by turing machines
i.e.
There are problems that they are not algorithmically computable

Diagonalization Method & Infinities

- **Last Question :**
  - We have seen that :
    - Acceptance in regular is solvable/computable
    - Acceptance for Context-Free Grammar is also solvable. (Convert to Chomsky Normal form …)
    - What about the Acceptance Problem for Turing Machine ?

Diagonalization Method & Infinities

- **Acceptance Problem for Turing Machine :**
  - A_{TM} = {< M, w > | M is a TM and M accepts w}.
  - We assume that A_{TM} is decidable and obtain a contradiction. Suppose that H is a decider for A_{TM} such that.

$$H(\langle M, w\rangle) = \begin{cases} accept & \text{if } M \text{ accepts } w \\ reject & \text{if } M \text{ does not accept } w. \end{cases}$$

Diagonalization Method & Infinities

- **Acceptance Problem for Turing Machine :**
  - We construct a new Turing machine D with H as a subroutine. But, D does the opposite of H
  - D = “On input <M>, where M is a TM:
    - *Run H on input <M, <M>>.*
    - *Output the opposite of what H outputs.*

$$D(\langle M\rangle) = \begin{cases} accept & \text{if } M \text{ does not accept } \langle M\rangle \\ reject & \text{if } M \text{ accepts } \langle M\rangle . \end{cases}$$

Diagonalization Method & Infinities

- **Acceptance Problem for Turing Machine :**
  - In English :
    - H accepts <M, w> exactly when M accepts w.
    - D rejects <M> exactly when M accepts **<M>**.
    - D **rejects** <D> exactly when D **accepts** <D> ?

$$D(\langle M\rangle) = \begin{cases} accept & \text{if } M \text{ does not accept } \langle M\rangle \\ reject & \text{if } M \text{ accepts } \langle M\rangle. \end{cases}$$

Diagonalization Method & Infinities

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	*accept*	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	*reject*	*reject*		*reject*	
M_4	*accept*	*accept*	*re*[illegible]	[illegible]		[illegible]	
$\vdots$		$\vdots$					
D	*reject*	*reject*	*acc*[illegible]	[illegible]		[illegible]	
$\vdots$		$\vdots$					$\ddots$

M2 accepts <M2>,
Therefore
D rejects <M2>

pt $\langle M \rangle$

Diagonalization Method & Infinities

-

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	<u>*accept*</u>	*reject*	*accept*	*reject*		*accept*	
M_2	*accept*	<u>*accept*</u>	*accept*	*accept*	$\cdots$	*accept*	$\cdots$
M_3	*reject*	*reject*	<u>*reject*</u>	*reject*		*reject*	
M_4	*accept*	*accept*	*reject*	<u>*reject*</u>		*accept*	
$\vdots$		$\vdots$			$\ddots$		
D	*reject*	*reject*	*accept*	*accept*		<u>*?*</u>	
$\vdots$		$\vdots$					$\ddots$

pt $\langle M \rangle$

Diagonalization Method & Infinities

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	$\cdots$
M_1	*accept*	*reject*	*accept*	*reject*		*accept*	

Which is a contradiction, therefore, The acceptance problem for Turing machine is not decidable

Proving by Reducibility

- **Problem** : Compute the area of a given square
    - The problem can be **reduced to** finding the width of the square

- A reduction is a way of converting one problem to another problem in such a way that a solution to the second problem can be used to solve the first problem
- **Reducibility** always involves two problems, which we call A and B. If A reduces to B, we can use a solution to B to solve A

Undecidability of the Halting Problem

- **Problem** : Can we decide whether turing machine halts on a given input either by accepting or rejecting without looping forever ?
- In the same way : Can we prove that a C/Pascal/Python program terminates on a given input ?
  - Question from the first lecture ?
- This is known as the **Halting Problem**

```
input n;
assume n>1;
while (n !=1) {
   if (n is even)
    n := n/2;
  else
    n := 3*n+1;
}
```

Undecidability of the Halting Problem

- **Problem** : Halting Problem
- Language Representation :
  - $HALT_{TM}$ = {< M, w > | M is a TM and M halts on input w}
- Proof :
  - We use the undecidability of A_{TM} to prove the undecidability of the halting problem by reducing A_{TM} to $HALT_{TM}$

Undecidability of the Halting Problem

- **Problem** : Halting Problem
- Let's assume :
  - S is the Turing Machine for the acceptance problem.
  - R is the Turing Machine for the Halting Problem

Undecidability of the Halting Problem

- **Problem** : Halting Problem
- Let's assume :
  - S is the Turing Machine for the acceptance problem.
  - R is the Turing Machine for the Halting Problem
- S = "*On input <M, w>, an encoding of a Turing machine M and a string w:*
  - *Run TM R on input < M, w>*
    - *If R rejects, reject .*
    - *If R accepts, simulate M on w until it halts.*
  - *If M has accepted, accept ; if M has rejected, reject .*

Undecidability of the Halting Problem

- **Problem** : Halting Problem
- Let's assume :
  - S is the Turing Machine for the acceptance problem.
  - R is the Turing Machine for the Halting Problem
- S = "*On input <M, w>, an encoding of a Turing machine M and a string w:*
  - *Run TM R on input < M, w>*
    - *If R rejects, reject .*
    - *If R accepts, simulate M on w until it halts.*
  - *If M has accepted, accept ; if M has rejected, reject .*

This shows clearly that the Acceptance Problem is Decidable, which is a contradiction,

Therefore,

The halting problem is undecidable

More Undecidable Questions

- E_{TM} is undecidable
- EQ_{TM} is undecidable.
- …

Post Correspondence Problem

- Post Correspondence Problem (PCP) :Given a collection of dominos in the form of two string: $$\left\{ \left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right] \right\}.$$
- The task is to make a list of these dominos (repetitions permitted) so that the string we get by reading off the symbols on the top is the same as the string of symbols on the bottom

Post Correspondence Problem

- Post Correspondence Problem (PCP) :Given a collection of dominos in the form of two string:

$$\left\{ \left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right] \right\}.$$

We have to start with this domino because the first letter of the numerator is the same as first letter of the denominator

se dominos (repetitions permitted) so

g off the symbols on the top is the

on the bottom

Post Correspondence Problem

- Post Correspondence Problem (PCP) :Given a collection of dominos in the form of two string: $$\left\{ \left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right] \right\}.$$
- The task is to make a list of these dominos (repetitions permitted) so that the string we get by reading off the symbols on the top is the same as the string of symbols on the bottom

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]$$

Post Correspondence Problem

- Post Correspondence Problem (PCP) :Given a collection of dominos in the form of two string: $\left\{ \left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right] \right\}.$
- The task is to make a list of these dominos (repetitions permitted) so that the string we get by reading off the symbols on the top is the same as the string of symbols on

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]$$

We need a domino with a leading b on the numerator

Post Correspondence Problem

- Post Correspondence Problem (PCP) :Given a collection of dominos in the form of two string:

$$\left\{ \left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right] \right\}.$$

- The task is to make a list of these dominos (repetitions permitted) so that the string we get by reading off the symbols on the top is the same as the string of symbols on the bottom

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]\left[\frac{\mathtt{b}}{\mathtt{ca}}\right]\left[\frac{\mathtt{ca}}{\mathtt{a}}\right]\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]\left[\frac{\mathtt{abc}}{\mathtt{c}}\right]$$

Post Correspondence Problem

- Post Correspondence Problem (PCP) :Given a collection of dominos in the form of tv

  $\left[\frac{abc}{c}\right]\Big\}.$

- The task is to permitted) so that the string we get by reading off the symbols on the top is the same as the string of symbols on the bottom

This instance of the problem is SOLVABLE. COMPUTABLE, DECIDABLE

BUT

$$\left[\frac{a}{ab}\right]\left[\frac{b}{ca}\right]\left[\frac{ca}{a}\right]\left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right]$$

Post Correspondence Problem

- Post Correspondence Problem (PCP) :Given a collection of dominos in the form of tv … $\left[\frac{abc}{c}\right]\Big\}$.

Is the problem in general (for all instances) solvable or computable ?

- The task is to … ermitted) so that the string we get by reading off the symbols on the top is the same as the string of symbols on the bottom

$$\left[\frac{a}{ab}\right]\left[\frac{b}{ca}\right]\left[\frac{ca}{a}\right]\left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right]$$