CS3063 Theory of Computing

Semester 4 (20 Intake), Feb – Jun 2023

Lecture 13

Decidability (Solvability) – 1

Sanath Jayasena

Announcements

- This is the final week of the semester
  - Lecture 13
  - Lecture 14

- Please fill Student Feedback on Moodle

- Final Exam
  - 5^{th} July, 9.00 11.00am

Outline:

Lecture 13

Decidability - 1

- Decidability
  - Decidable problem?
  - Hilbert's 10^{th} Problem
  - Solving a Polynomial
- Countable / Uncountable Sets
  - Diagonalization Method
- Classes of Languages

PART 1

Outline:

Lecture 13

Decidability - 1

- Decidability
  - Decidable problem?
  - Hilbert's 10th Problem
  - Solving a Polynomial
- Countable / Uncountable Sets
  - Diagonalization Method
- Classes of Languages

What is a Decidable Problem?

- A problem is **decidable** if it can be answered with either a **yes** or **no** after an algorithmic process (a finite # of steps)
- Otherwise it is **undecidable**

- What is an algorithm? Can we define it?
  – Although algorithms have a long history, the notion not defined precisely until 20th century

Hilbert’s 10^{th} Problem

- In 1900 mathematician David Hilbert posed as challenges 23 problems
  - 10^{th} one was in general about algorithms, specifically on integral roots of a polynomial
  - Hilbert’s 10^{th} problem: devise an “algorithm” that tests if a polynomial has an integral root
    - Exact term “algorithm” was not used
  - His assumption: an algorithm exists
  - But this problem is algorithmically unsolvable
  - Could not conclude at that time

Notion of Algorithm

- By *Alan Turing* and *Alonzo Church* in 1936
  - *Church* used **λ-calculus** and Turing used **machines** to define algorithms
  - These 2 definitions were shown equivalent
  - This is the idea in the Church-Turing Thesis

- In 1970, Hilbert's 10^{th} problem was shown to have no algorithm

Example 1: Solving a Polynomial

- Let us consider the Hilbert's 10^{th} problem

  Let D = {p | p is a polynomial with an integral root}

- The problem: is the set D decidable?
  - Answer: No (but it is Turing-acceptable)

- Let's consider a simpler problem; let

  D1={p | p is a polynomial over x with an integral root}

Solving a Polynomial

- Here is a TM T1 that accepts D1

> T1 = “The input is a polynomial p over the variable x.
>
> 1. Evaluate p with x set successively to the values 0, 1, -1, 2, -2, 3, -3, …. If at any point the polynomial evaluates to 0, accept.”

High-level description of a Turing machine

  - If p has an integral root, T1 will eventually find it and accept
  - If not, T1 will run for ever

Solving a Polynomial

- For the general (multivariable) case, we can have a similar TM T that accepts D
  - Here T goes through all possible settings of its variables to integral values
- Both T1 and T are acceptors, not deciders
  - T1 can be converted to a decider for D1 (can find bounds for search; then reject if fails)
  - But it is impossible to find such bounds for multivariable case (for T and D)

Example 2: Connected Graphs

- Let A be the language consisting of all strings representing undirected graphs that are *connected* (a graph is connected if every node can be reached from every other node by traveling along the edges)
- We write

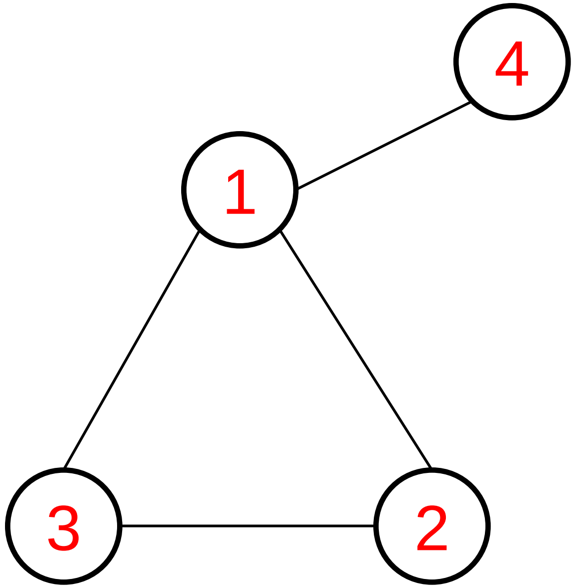
  A = {<G> | G is a connected undirected graph}

  (<G> is the encoding of G into a string)
- Give a Turing machine T that decides A

Example 2: Connected Graphs

- T = "On <G>, the encoding of graph G:
  1. Select the first node of G and mark it
  2. Repeat the following step until no new nodes are marked
  3. For each node in G, mark it if it attached by an edge to a node that is already marked
  4. Scan all the nodes of G to determine if they all are marked. If yes, accept; else reject."

Example Graph and Its Encoding



(a) Graph G

(1, 2, 3, 4)
((1,2), (2,3), (3,1), (1,4))

(b) Encoding <G>

PART 2

Outline:

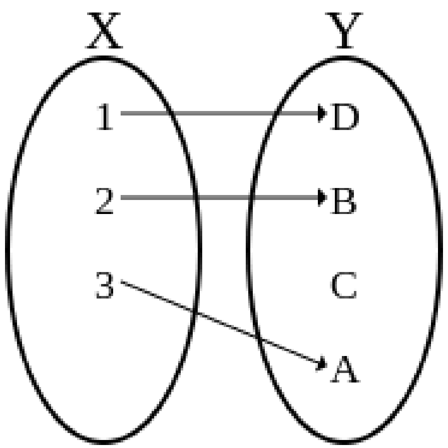
Lecture 13

Decidability - 1

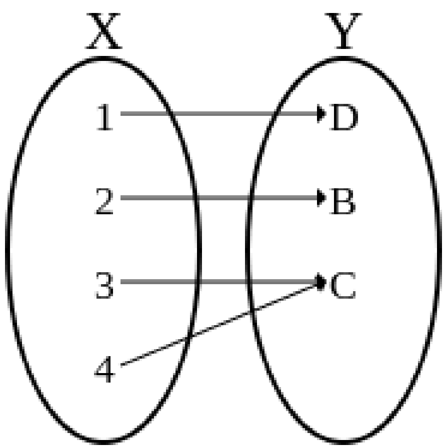
- Decidability
  - Decidable problem?
  - Hilbert's 10^{th} Problem
  - Solving a Polynomial
- Countable / Uncountable Sets
  - Diagonalization Method
- Classes of Languages

Countable Sets etc.

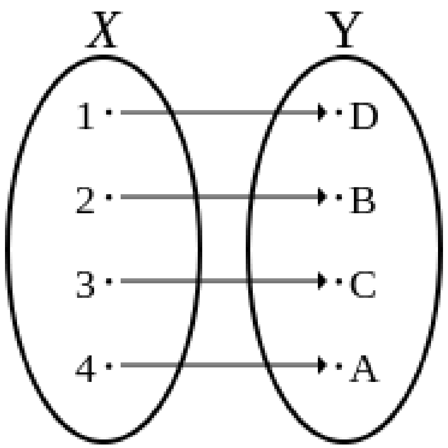
- A **countable set** (due to Georg Cantor)
  - A set that is either finite or has the same size as the set of *natural numbers* **N**={1,2,3,…}
  - A set **S** for which there exists an *injective function* ***f*** from **S** to **N**={1,2,3,…}
  - E.g., set of evens/odds, +ve rational numbers
- An **uncountable set**
  - A set that is not countable
  - E.g., set of real numbers **R**



Injective function or **injection** or **one-to-one** function: preserves distinctness, no 2 elements in X map to same y in Y



Surjective function or a **surjection** or **onto** function: every element y in Y has a corresponding element x in X



Bijective function or a **bijection** or a **one-to-one correspondence**: exact pairing of elements of two sets (both injective and surjective)

Comparing Infinite Sets

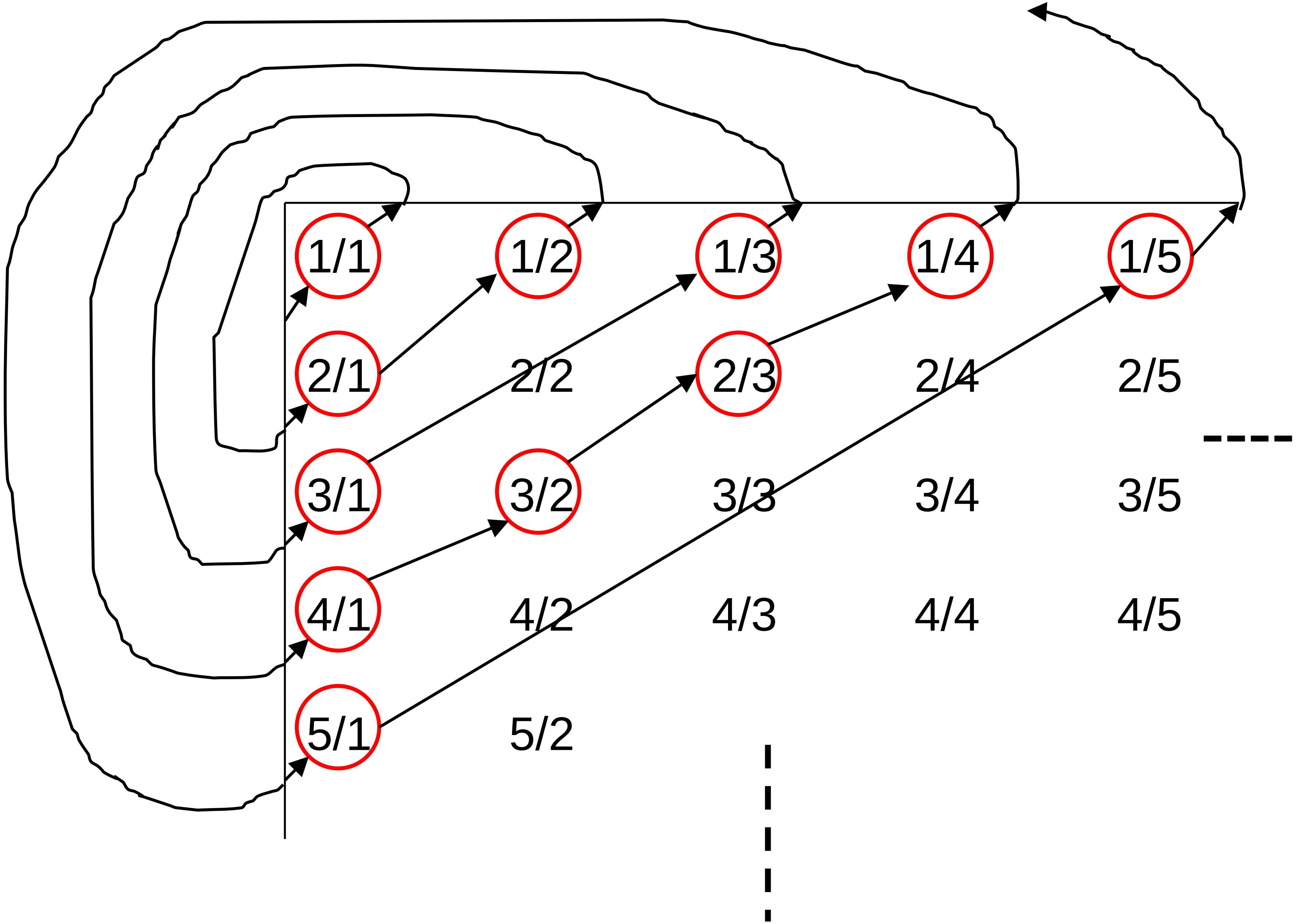
- If we have two infinite sets, how can we say if one is larger, or they are same size?
  - Can do easily for finite sets by counting
  - Counting method will not work for infinite sets
  - E.g., {even integers} vs {all strings in {0,1}*}
- Georg Cantor proposed *diagonalization method*
  - Based on pairing of elements between 2 sets
  - Compares sizes without counting

Example 1

- Show that the set of natural numbers **N**={1,2,3,…} and the set of even natural numbers **E**={2,4,6,…} have the same size
  - Using Cantor's idea: size of **N** = size of **E**
  - Correspondence f to map **N** to **E**, f(n) = 2n
- Is this correct?
  - Intuitively **E** seems smaller than **N**
- But each element of **N** can be paired with a unique element in **E**

Example 2

- Let **Q**={m/n | m, n are in **N**} be the set of positive rational numbers
  - **Q** seems much larger than **N**
  - Yet they are of same size (as per definition)
  - Can give correspondence with N to show Q is countable
  - But, how exactly?
  - List all elements of **Q**, then start pairing with **N**
  - List the elements in a matrix, go diagonally


1/1 1/2 1/3 1/4 1/5
2/1 2/2 2/3 2/4 2/5
3/1 3/2 3/3 3/4 3/5
4/1 4/2 4/3 4/4 4/5
5/1 5/2

Uncountable Sets

- For some infinite sets, no correspondence with **N** exists
- Such sets are “too big”, and “uncountable”
- E.g., the set of real numbers **R**
- Cantor proved **R** is uncountable
  - Introduced the diagonal method in doing so
- How to show **R** is uncountable?
  - Think about it, and read… (homework)

Uncountable Sets

- Implications to theory of computation
  - Some languages are not decidable or even Turing-acceptable
  - Because, there are uncountably many languages
  - But only countably many Turing machines
  - (Each TM can accept one language and there are more languages than TMs)

Languages not accepted by a TM?

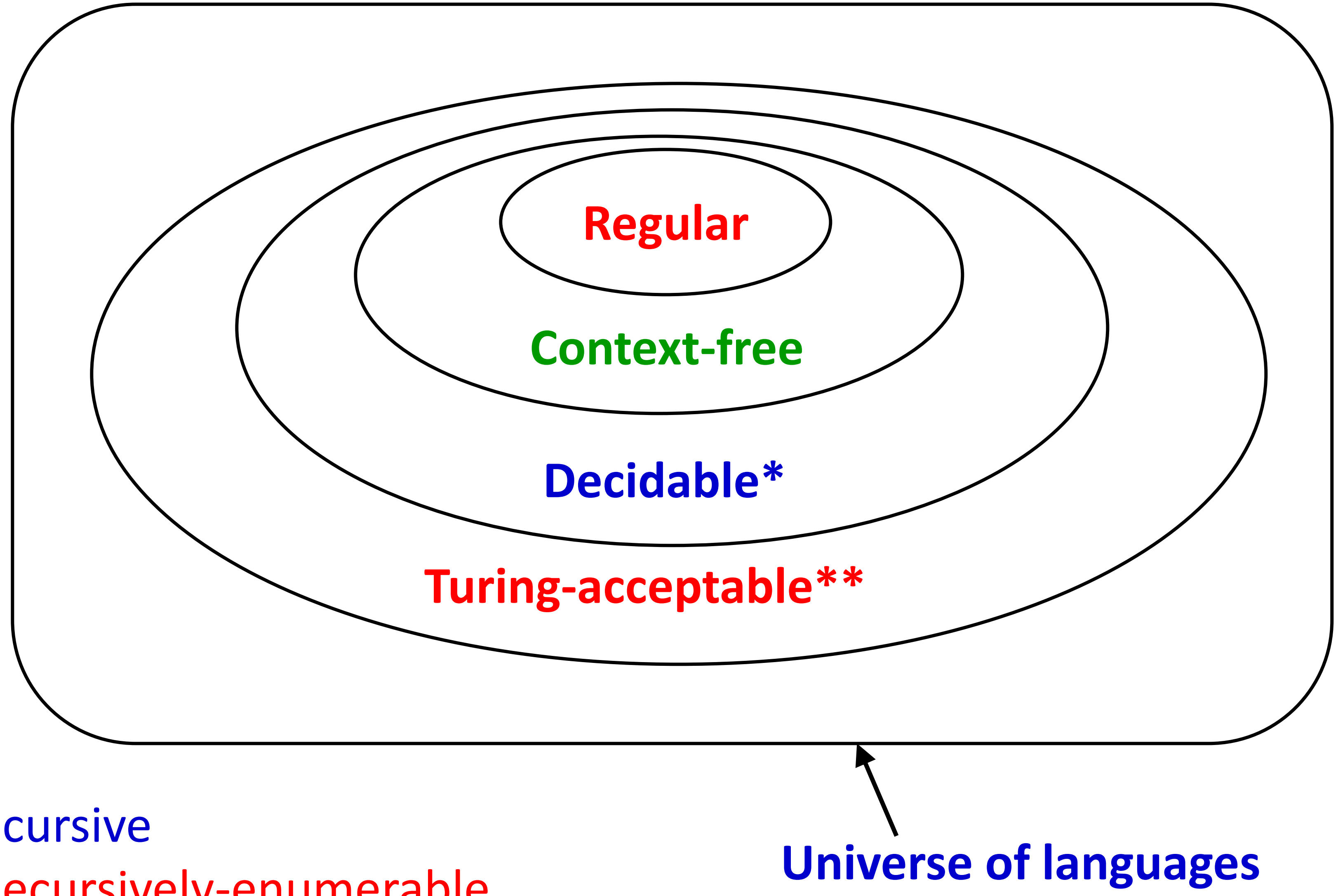
- Not all languages recursively enumerable (RE)
- Set of languages (includes that are not RE) is bigger than the set of languages that are RE
- Proof based on counting set elements, countable and uncountable sets
  - Main idea: the set of languages bigger than the set of TM's (a TM can accept 1 language)
  - Both are infinite sets but the 1st set is bigger !!

Languages not accepted by a TM?

- Some results
  - Languages are sets
    - If Σ is a finite set, the set Σ^* of strings is countable
  - The set of RE languages is countable
  - The set of languages is uncountable
- There are languages that are not RE
  - These cannot be accepted by TM

- Ref: Section 10.5 and Chapter 11

Classes of Languages



* Recursive

** Recursively-enumerable

L13: Conclusion

- We discussed
  - Decidability
    - Hilbert's 10^{th} problem
    - Notion of an Algorithm
    - Solving a Polynomial
  - Countable / Uncountable Sets
    - Diagonalization Method
  - Classes of Languages