CS21004 Formal Languages and Automata Theory (FLAT)

Abhijit Das

Sudeshna Kolay

3-1-0

3 hours of recording

1 hour of tutorial + doubt clearance

F4

Friday (11:00 am – 12:00 noon)
++

3 – 4 short tests

3 – 4 long tests

1 – 2 programming assignments

Dexter Kozen, Automata and Computability
— Good Exercise Sets

Why this course?

- To understand ourselves as Computer Scientists
- what is computation?
- what is computable? what is not?
- Is everything computable? (NO)

Historically

David Hilbert, 1900, a set of open mathematical problems

- Hilbert's tenth problem

Entscheidungsproblem
"Decision" problem
Decidability problem

whether there exists an "effective computation" procedure

→ what is understood by this?

Several paradigms

- λ-calculus
- μ-recursion
- Post problems
- Unrestricted Grammars
- "Turing machines" ⟵
- ...
- C programs / Assembly-language problem

effective computability

All encompass the <u>same</u> notion of computability.

<u>Church-Turing Thesis</u>: Define whatever is computable.

Formal Treatment of Computability

$U \rightarrow$ universal set

(Every element of U has a finite representation)

$A \subseteq U$

$\hookrightarrow$ some subset

$x \in U$

To decide: whether $x \in A$.

Important: A must also be finitely specified.

- Not exactly a loss of generality

$f : A \rightarrow B$

B: $\{0, 1\}$, $\{F, T\}$, $\{N, Y\}$

- stood the test of time

How to specify A finitely?

- English-language description
- Mathematical description
- Using a set of rules (usually recursive)
- Machines

If A is finite, A can be exhaustively enumerated.

If A is infinite, we need some formalism to fully characterize the members of A.

$U \rightarrow$ universal set

$A \subseteq U$ A is called a <u>language</u>

English Language

— Grammar (a set of rules)

+ a vocabulary.

<u>Languages of numbers</u> (integers)

$$\mathbb{N} = \{1, 2, 3, \ldots\}$$

$$\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$$

$$\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$$

A language of numbers is a set of integers (+ve/non-ve)

$\mathbb{N}, \mathbb{N}_0, \mathbb{Z} \longrightarrow$ countable

The set of all languages of numbers

is $\mathcal{P}(\mathbb{N}) \longrightarrow$ not countable

by the power-set theorem

Not all languages can have a finite representation.

Some do have

└── interested in these

Some will introduce the notion

of uncomputability.

Examples:

$U = \mathbb{N}, \mathbb{N}_0, \mathbb{Z}$

Each integer has a finite representation

— decimal, binary, ...

1. $E = \boxed{\text{the set of even natural numbers}}$ → English language description

$= \{2, 4, 6, 8, 10, 12, \ldots\}$ → not a finite representation

$= \boxed{\{2n \mid n \in \mathbb{N}\}}$ mathematical description

$x \in \mathbb{N}$ decide whether $x \in E$

binary rep → x ends with 0

decimal rep → x ends with 0, 2, 4, 6, 8.

2. $\mathbb{P} = \{2, 3, 5, 7, 11, 13, \ldots\}$ ← not a finite description

$= \boxed{\text{the set of all primes}}$

To check $x \in \mathbb{P}$

→ primality-testing problem

3. $PS = \{ n^2 \mid n \in \mathbb{N} \}$

$x \overset{?}{\in} PS$ Compute $a = \lfloor \sqrt{x} \rfloor$. Check whether $x = a^2$.

4. $PT = \{ 2^n \mid n \in \mathbb{N}_0 \}$

5. Fibonacci numbers

$$F = \{0, 1, 2, 3, 5, 8, 13, \ldots\}$$

$$= \{F_n \mid n \geq 0\}$$

Rules to generate F_n

$$\boxed{\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \quad \text{for } n \geq 2 \end{aligned}}$$

finite description to generate/define all Fibonacci numbers

$x \in \mathbb{N}$, to check whether $x \in F$, that is, $x = F_n$ for some n.

$F_0, F_1, F_2, \ldots, F_n$ so long as $F_n \leq x$. Check whether $F_n = x$.

6. Happy numbers

$21 \rightarrow 2^2 + 1^2 = 5 \rightarrow 5^2 = 25 \rightarrow 2^2 + 5^2 = 29$

$\rightarrow 2^2 + 9^2 = 4 + 81 = 85 \rightarrow 8^2 + 5^2 = 89 \rightarrow$

$8^2 + 9^2 = 145 \rightarrow 1^2 + 4^2 + 5^2 = 42 \rightarrow 16 + 4 = 20$

$\rightarrow 4 \rightarrow 16 \rightarrow 37 \rightarrow 3^2 + 7^2 = 58 \rightarrow 5^2 + 8^2 = 89$

$4 \rightarrow 16 \rightarrow 37 \rightarrow 58 \rightarrow 89 \rightarrow 145 \rightarrow 42 \rightarrow 20$

unhappy number

sad

$23 \rightarrow 2^2 + 3^2 = 13 \rightarrow 1^2 + 3^2 = 10 \rightarrow 1^2 + 0^2 = 1$

$\rightarrow$ happy number.

A procedure describes/specifies the happy numbers.

Recursive procedure

Algorithm

Termination

$\text{ishappy}(n)$

$\Leftrightarrow (n = 1) \text{ or } (\text{ishappy}(\text{sos}(n)))$

2020, 2021 → not happy

looping ???

happy ← 2003, 2008, 2019, 2026, 2030, ...

— Keep a list of numbers generated so far

Exercise: Every unhappy number falls into the loop involving 4. (Prove it.)

Termination: $(n = 1)$ or $(n = 4)$ → unhappy

happy

7. Sphenic numbers

$$SPH = \{ pqr \mid p, q, r \text{ are distinct primes} \}$$

$2 \times 3 \times 5 = 30$, $\quad 1001 = 7 \times 11 \times 13$,

$2022 = 2 \times 3 \times 337$

are sphenic

$2^2 \times 3 \times 5 = 60$, $\quad 210 = 2 \times 3 \times 5 \times 7$ are not sphenic.

$\underline{x \overset{?}{\in} SPH}$ $\qquad$ Factor x $\longrightarrow$ doable in finite time

8. Iterated logarithm function

$$\log \equiv \log_2$$

$n \quad \log(n) \quad \log(\log(n)) \quad \ldots$ until we get a value ≤ 1

$\log 2 = 1$

$\log 3 = 1.\ldots \qquad \log(\log 3) < 1$

$\log 4 = 2, \ \log(2) = 1$

$\log^* 2 = 1$

$\log^* 3 = 2$

$\log^* 4 = 2$

$\log^* 5 = 3$

$\log^* 8 = 3$

Largest ~~Smallest~~ n

$\log^* n = 4 \qquad 16 \to 8 \to 4 \to 2 \to 1$

$\log^* n = 5 \qquad 2^{16} \to 16 \to 8 \to 4 \to 2 \to 1$

$\log^* n = 6 \qquad 2^{2^{16}} \to 2^{16} \to \ldots \to 1$

2^{65536} > the no. of electrons in the known universe

$$IL = \{ x \mid \log^* x = n \in \mathbb{N} \text{ and } x \text{ is } \cancel{\text{smallest}} \text{ largest among those } y \text{ s.t. } \log^* y = n \}$$

$$IL_1 = 1$$

$$IL_2 = 2$$

$$IL_3 = 8$$

$$IL_4 = 16$$

$$IL_5 = 2^{16} = 65536$$

$$IL_6 = 2^{65536}$$

$$IL_7 = 2^{IL_6} = 2^{2^{65536}}$$

$$IL_n = 2^{IL_{n-1}}$$

<u>recursively</u> defined

fast-growing function of n.

9. Busy-beaver numbers

$BB(n) \longrightarrow$ can be defind using Turing machines

$BB(n)$ grow no rapidly with n that computers cannot keep track of them

uncomputable numbers:

If $f(n)$ is <u>any</u> <u>computable</u> functions, then

$$BB(n) > f(n).$$

Languages of

- numbers
- sets (finite)
- polynomials
- graphs
- sequences
- ...

"Generic model" of a <u>computational</u> <u>problem</u>

Languages of <u>things</u>

$U \to$ set of <u>things</u>

Each thing must have a finite representation

$A \subseteq U$

$\hookrightarrow$ finite specification

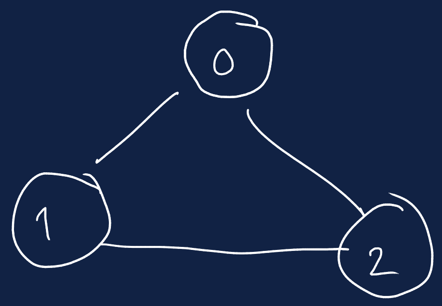
$x \in U$, decide whether $x \in A$.

Almost every thing that has a finite representation has a <u>finite encoding</u> as a string

$\mathbb{N}$ binary representation (sequence of 0's and 1's)

$G = (V, E)$ $|V| = n$

$\underbrace{11 \cdots 1}_{n \text{ times}} 0 \boxed{n^2 \text{ bits}} \to$ Adjacency matrix



$\boxed{1110011101110}$ finite encoding

* Use <u>strings</u> to define languages.