COL 352 Lecture 2 Jan 2, 2019

	List A x_i	w_i List B
1	10	101
2	011	11
3	101	011

$i_1 = 1$: no other option

10 101

$i_2 = 1$ 1010 $\boxed{0}$ not possible to get a soln 101 $\boxed{1}$ 01

$i_2 = 3$ $\boxed{10 \cdot 101}$ missing 10101 $\boxed{1}$

This pattern will repeat, i.e. no soln

Post correspondence Problem (PCP)

Given K pairs of strings, (x_1, w_1) (x_2, w_2) (x_k, w_k) is there a sequence $i_1\, i_2 \cdots i_m$ $i_j \in \{1, \cdots k\}$ s.t. $x_{i_1} \cdot x_{i_2} \cdot x_{i_3} \cdots x_{i_m} = w_{i_1} w_{i_2} \cdots w_{i_m}$

An intuitive algorithm

Generate all sequences of strings over $\{1 \ldots k\}$ in increasing lengths, say $i_1 i_2 \ldots i_\alpha$ and check if $x_{i_1} x_{i_2} \ldots x_{i_\alpha} = w_{i_1} \cdot w_{i_2} \ldots w_{i_\alpha}$

<u>Observation</u>. If answer is Yes, it will terminate otherwise will continue forever

<u>Thm</u>: There is no algorithm that solves all instances of PCP

(similar to halting problem)

Decision problem : correspond to functions that have binary output

$f : \mathbb{Z} \rightarrow \{YES/NO\}$

More general functions $f : \mathbb{Z} \rightarrow \mathbb{Z}$ say $f(x) = y$, then we define a two input function $g(x,y) : \mathbb{Z} \times \mathbb{Z} \rightarrow \{Y, N\}$

such that $g(x, y) = \begin{cases} Y & \text{if } y = f(x) \\ N & \text{otherwise} \end{cases}$

So we will deal only with decision problems

Strings : sequence of symbols over some finite alphabet Σ

Σ : all strings of length 1

$$\Sigma \cdot \Sigma = \{ a \cdot b \mid a, b \in \Sigma \}$$

$\Sigma^k = \Sigma \cdot \Sigma^{k-1}$ strings of length k

$\Sigma^0 : \varepsilon$ a special symbol 'empty string'

$$\varepsilon \cdot w = w \cdot \varepsilon = w$$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$$

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \cdots$ (no length 0 string)

<u>Important</u> : any string is a finite sequence of symbols even though Σ^* is infinite

(Every integer is finite although $\mathbb{Z}$ is infinite)

Strings and Integers

Σ^* can be mapped to integers,
for eg. enumerate all strings starting
from ε, followed by strings of length 1,
length 2, ... etc.

Within the same length strings use any
ordering, say lexicographic

Lexicographic ordering doesn't define a mapping
when it is used on Σ^* directly

So we are going to use strings and integers
interchangeably

Language $L \subset \Sigma^*$ (and therefore $\subset \mathbb{Z}$)

Membership problem in Languages:

Given a language L and a string $x \overset{?}{\in} L$

YES NO

Eg. L_E : set of even integers

L_O : set of odd integers

L_P : set of primes

L_{sq} : set of perfect squares

Characteristic function of a language L

$$\chi_L(y) = \{1 \text{ if } y \in L \text{ and } 0 \text{ otherwise}\}$$

Also called language recognition

Certain languages are harder to recognise

for example L_{PCP} : strings that are YES instances

is not even possible

Hierarchy of classes of languages

From easier to "harder"

- in terms of circuit/data structure
- in terms of time/space

Chomsky hierarchy Natural language ??

Some historical perspective

David Hilbert felt that all of mathematics can be mechanized or proved algorithmically

Kurt Göedel disproved this with his famous Incompleteness of logic

Not all true statements in any axiomatic system (like number-theory) can be proved

Many more true statements than proofs

Different attempts for defining computation

Church/Rosser : λ calculus

Alan Turing : Turing machine

But all attempts converged to a single "equivalence class" which is now considered as a universal notion of computation

Church – Turing thesis

A cardinality based argument

If all languages are subsets of Z

$\Rightarrow$ # languages 2^{Z} (power set)

All programs can be thought of as strings

$\Rightarrow$ # programs $\leq Z$

There is a fundamental mismatch of the set and powerset (Russel's paradox)

So too few programs and too many languages