Introduction to Theory of Computation

Prof. (Dr.) K.R. Chowdhary
Email: kr.chowdhary@iitj.ac.in

Former Professor & Head, Department of Computer Sc. & Engineering
MBM Engineering College, Jodhpur

Wednesday 30th November, 2016

What is TOC?

The TOC is that knowledge of computer science, which does not change with time & technology. The course is divided into:

1. Theory of automata: Every process (including computing) can be divided into discrete sequence of states, where there is always *start* state, and there is a *final* state, with in between states. In this category, we may consider the computation, and growth of life, and even the planatary motions. All these systems are automata. The concept of automata was given by Von Neumann, in 1940s, in the form of cellular automata.
2. Theory of languages: Under this comes all types of grammars - computers' languages grammars, spoken languages grammars, and yet unknown grammars, and all corresponding languages. The concept of generative grammar was given first time by Prof. Noam Chomsky, in 1956.
3. Theory of Computation: The contributors were Kurt Gödel, Alonzo Church, Alan Turing, Kleene. However, the maximum contribution is due to Alan Turing, who gave the concept of Turing machine in his 1936 paper (much before the computers came into existence).

What is TOC?

- TOC attempts to establish a theory to all subjects of CS, just like classical dynamics to motion of objects
- Solvability of a problem $\Rightarrow$ existence of algorithm.
- TOC says that some problems are unsolvable, e.g., Russell's Paradox, given by $\{x|x \notin x\}$
- TOC deals with Physical limits of computing, Methodologies of Algorithm design, etc.
- CS is : Science of algorithm processing, representation, storage, transmission of information
- Nature of problems we deal with are classified based on their level of difficulty for computer,i.e., complexity
- Application of computer theory: writing programs to construct compilers, assemblers, OS, ..., and languages

Meta-Theory and Proofs

- Meta Theory: Theory about the theory (mathematics)
- Defines what is computable: A thing is computable if it can be representable by an algorithm.
- Defines the Limits of computation
- Proofs: Establishing that a statement is valid. One approach is: a statement is valid if there is no counter example. e.g., All even integers are divisible by 2 is valid. Proof by contradiction: find at least a single case, where this is not true. Other approaches are:
  1. Proofs by Deduction: Taking one value, then iterating is proof by induction.
  2. Proof by Induction: Execution of a sequence of program statements is deductive proof

Functions

- $f : D \to R$
- for $a, b \in D, f(a) = f(b) \Rightarrow a = b; f$ is injection or one-to-one mapping.
- for each $b \in R$, there is always $a \in D$ such that $f(a) = b; f$ is surjection (onto)
- A relation which is both injection and surjection is called bijection.

  Defining Natural Numbers

  $0 = \{\} = \phi$

  $0^+ = 1 = \{\text{elements of 0, set 0 as element}\} = \{\phi\}$

  $1^+ = 2 = \{\phi, \{\phi\}\}$
- $2^+ = 3 = \ldots$
- $\ldots$

Infinite sets

- Hence,

  $i)0 \in \mathbb{N}$

  ii)if $n \in \mathbb{N}, \therefore n^+ \in \mathbb{N}$
- If there is a bijection between any set A and set $n \in \mathbb{N}$,then A is finite, alternatively,
- If there is a bijection between any set A and set $\mathbb{N}$, then A is infinite,
- You can add any thing into an infinite, it results infinite only. Example: Hilbert's Hotel: To accommodate more guests in this hotel, (say 1), push occupant of room 1 into 2, of 2 into room 3, and so on. The room 1 is vacated, and ready for new guest.
- You can subtract any thing from infinity, it remains infinity. For example, you can map 40 onwards to infinity. For example, $40 \leftrightarrow 1, 41 \leftrightarrow 2, 43 \leftrightarrow 3, \ldots$

Diagonalization Theorem

Theorem

There is no bijection between infinite sets $\mathbb{N}$ and $\mathbb{R}$.

Proof.

- Assume that $f : \mathbb{N} \to \mathbb{R}$ is bijection, $\therefore$,

$\mathbb{N}$	$\mathbb{R}$
0	$0.b_{00}b_{01}b_{02}\ldots$
1	$0.b_{10}b_{11}b_{12}\ldots$
2	$0.b_{20}b_{21}b_{22}\ldots$
$\ldots$	$\ldots$
i	$0.b_{i0}b_{i1}b_{i2}\ldots$

Therefore this list of $\mathbb{R}$ is exhaustive. $b_{ij} \in \{0,1\}$

Now consider the number, $x = 0.\neg b_{00}\neg b_{11}\neg b_{22}\ldots$ Since x is new, hence the list of $\mathbb{R}$ is not exhaustive, this contradicts the assumption. Thus, the mapping is not bijection. $\mathbb{R}$ is uncountably infinite.

Cantor's Theorem

Theorem

For $f : A \to \wp(A), f$ is not surjection.

Proof.

Assume that f is surjection. Since $\wp(A)$ are all subsets of A, consider $B \subseteq A$, as an image of an element of A, which does not include the element itself. $\therefore$, $B = \{a \in A | a \notin f(a)\}$

- Thus, for every $a \in A$, we have $a \in B$ *iff* $a \notin f(a)$. $\therefore B \neq f(a)$, for all $a \in A$.
- Thus, B is not in the image of f. This contracts our assumption that we considered in the beginning of this proof. Hence, this proves that f is not a surjection, and hence not a bijection also.

Cantor's Theorem for infinite sets

Theorem

For $f : \mathbb{N} \to \wp(\mathbb{N}), f$ is not surjection.

Proof.

Assume that f is surjection, to ultimately contradict it. Consider now:

$$\begin{array}{lcl} \mathbb{N} & \Leftrightarrow & \wp(\mathbb{N}) \\ 0 & \Leftrightarrow & \{5\} \\ 1 & \Leftrightarrow & \{2,3\} \\ 2 & \Leftrightarrow & \{1,2,5\} \\ 3 & \Leftrightarrow & \{2,4,8\} \\ \ldots & \Leftrightarrow & \ldots \end{array}$$

- We call the mapping between 2 and $\{1,2,5\}$ as *selfish*, and between 1 and $\{2,3\}$ as *nonselfish.*
- Next we build a special set D of *nonselfish* numbers, to arrive at contradiction. Naturally, $D \in \wp(\mathbb{N})$. And, $D = \{d | d \in \mathbb{N}, d \notin f(d)\}$. $\therefore D \neq f(d)$ (i.e., D is not any image!).This is contradiction. This shows that f is not surjection. (Note: Compare this with previous).

Functions

- $f : D \rightarrow R$

  $+ : int \times int \rightarrow int$

  $\subseteq : S_1 \times S_2 \rightarrow \{T, F\}$
- **Total Function:** defined over the entire domain

  $f : x \rightarrow 2x, x \in \mathbb{N}, \mathbb{N} = \{0, 1, 2, \ldots\}$
- **Partial Function:** defined over some domain points only.

  $f : x \rightarrow 2/x, x \in \mathbb{N}$

  A partial function can be used to model an iteration or infinite loops.

  $f(x)$: *if (x==0) then 1 else f(x)*

Strings

- $a, b, c, d, \ldots$ for individual alphabets
- $x, y, z, w, \ldots$ for variable names for strings
- We use symbol Σ for alphabet set for languages, e.g. $\Sigma = \{a, b\}$.
- Set of all strings on the alphabet set Σ are represented by Σ^* (Kleene Star).
- Strings have only one operation:Concatenation. E.g., $x \circ y$, $\varepsilon \circ x = x \circ \varepsilon = x$
- **Semigraoups** are algebraic systems which are closed on some binary operation, and have the property of associativity. For example, $\langle S, * \rangle$, such that for $a, b \in S$, there is $a * b \in S$.
- The semigroup is an algebra. It can be helpful in mapping all the string operations in computer to this algebra so that they can be formally treated, without technology boundaries of computers.
- **Monoid** is semigroup with a *neutral* element. E.g., $\langle M, \otimes, 1 \rangle$ is a Monoid with multiplication operation.

Languages

- *language:* set of words (or strings) defined over some alphabet. These words are simpler form of sentences. For example, $\{aab, bba, abbbb, \varepsilon, baaa\}, \{\varepsilon, a, bbbb, aaa\}, \phi$ are the languages over the alphabet set $\Sigma = \{a, b\}$.
- $\{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$ is an **infinite** language over $\Sigma = \{0, 1\}$, which contains all the binary words.
- $L_1 = \{w \in \Sigma^* | w \text{ has property } P\}$
- $L_2 = \{\}$; language with no sentences
- $L_\varepsilon = \{\varepsilon\}$; language with single null sentence
- Let $\Sigma = \{a, b\}$, then $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$
- $L_3 = \{a, ab, ba, aaa\}$ is **finite** language
- $L_4 = \{a^p | \text{ p is prime}\}$, $L_5 = \{a^i b^i | i \in \mathbb{N}\}$
- $L_3, L_4, L_5 \subseteq \Sigma^*$.
- $L_6 = \{a^i b^i c^i | i \in \mathbb{N}\}$

Representation of Languages

- Strings oveer Σ^* are countably infinite. Say $\mathbb{N}$
- Languages over Σ^* are $2^{|\Sigma^*|}$, which is uncountably infinite, say **N**.
- Think of the mapping, $f : \mathbb{N} \to \mathbf{N}$. It is not bijection, as $\mathbf{N} > \mathbb{N}$. $\therefore$, some languages cannot be represented ! (what ever tools, and methods may be used. Also, it cannot be possible to design machines to recognize some languages, what ever may be advancement of technology or science!)
- **Algorithmic problems:**

  $A : \Sigma_1^* \to \Sigma_2^*$; A is an algorithm which maps input strings to output, in some complex manner.

  if $\forall x\ A(x) = B(x), \therefore A \equiv B$, i.e., if two algorithms A and B produce same output for same input, then they are equal.

Operations on Languages

- Let L_1, L_2 be languages over the same alphabet Σ, then there are following operations on L_1, L_2:
- $L_1 \cup L_2 = \{w | w \in L_1 \text{ or } w \in L_2\}$
- $L_1 \circ L_2 = \{w = xy | x \in L_1, y \in L_2\}$
- $\overline{L_1} = \{w \in \Sigma^* | w \notin L_1\}$
- $L_1^k = \underbrace{(L_1 \circ L_1) \dots L_1}$, k-times, with $L_1^0 = \{\varepsilon\}$
- $L_1^* = \{w \in \Sigma^* | \exists k \geq 0, w \in L_1^k\}$. This is called Kleene Star of language L_1.
- $L_1^+ = \{w \in \Sigma^* | \exists k \geq 1, w \in L_1^k\}$.

Conclusion

- By study of theory of computation, we reach to important conclusions in computation, like from thermodynamic we conclude that perpetual machines cannot be made
- The course gives treatment to the subject, independent of technology.
- What problems we can solve by computers?
  1. Those, that produce a definite output: e.g., sorting the numbers
  2. Those that produce Y/N : Called Membership Problem, e.g., Is $x \in A$?
  3. Another, e.g., given a C program, to determine whether it will halt on so and so given input? (Halting Problem) - unsolvable.

Bibliography

- John C. Martin, "Introduction to Languages and Theory of Computation", McGraw-Hill.
- Mishra and Chandrashekharan, "Theory of Computer Science (Automata, Languages, and Computation) ",PHI, India
- Sipster, "Introduction to Theory of Computation", Thompson Press.
- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, "Introduction to Automata Theory, Languages, and Computation", Pearson Press, India.
- Christos H. Papadimitriou and Harry Lewis, "Elements of the Theory of Computation (2nd Edition)", Pearson Press, India.