EECS 2001N : Introduction to the Theory of Computation

Suprakash Datta
Office: LAS 3043

Course page: `http://www.eecs.yorku.ca/course/2001N`
Also on Moodle

Terminology for Languages

- Alphabet: a finite, non-empty set of characters, e.g., $B = \{0, 1\}$. Alphabets are generally denoted by the symbols Σ, Γ and characters are usually denoted by lowercase characters
- Strings: a concatenation of some (possibly zero) characters, e.g., 0111 is a string "over" (using the characters of) B
- Words: synonym for strings of alphabetical characters
- Languages: a set of words

Strings/Words

- Defined over an alphabet Σ
- Is a finite sequence of symbols from Σ
- Length of string w ($|w|$): length of sequence
- ϵ: the empty string is the unique string with zero length
- Concatenation of w_1 and w_2 (written $w_1 w_2$) – copy of w_1 followed by copy of w_2
- Notation: $x^k = xxx \dots x$ (k times)
- w^R: reverse of string w; e.g. if $w = abcd$ then $w^R = dcba$
- Lexicographic ordering: definition

Languages

- A language over Σ is a set of strings over Σ
- Typical examples ($\Sigma = \{0, 1\}$):
  - $L_1 =$ the set of finite bit strings
  - $L_2 = \{x | x \text{ is a bit string with two zeros } \}$
  - $L_3 = \{a^n b^n | n \in \mathbb{N}\}$
  - $L_4 = \{1^n | \ n \text{ is prime}\}$

A Special Language

- Definition: Σ^* is the set of all strings over Σ
- Any language L over Σ is a subset of Σ^* ($L \subseteq \Sigma^*$)
- Recursive definition of Σ^*:
  - $\epsilon \in \Sigma^*$
  - $\forall a \in \Sigma, \forall x \in \Sigma^*, xa \in \Sigma^*$
  - No other strings are in Σ^*

Exercises

- Suppose $\Sigma = \{a, b\}$. Define L recursively as
  - $a \in L$
  - $\forall x \in L,\ ax \in L$
  - $\forall x, y \in L,\ bxy \in L,\ xby \in L$ and $xyb \in L$
  - No other strings are in L
- Prove that L is the language of strings with more a's than b's.
- Does L change if you remove the first bullet point?
- Does L change if you add a rule $\epsilon \in$ Ł?

I/O vs Decision Problems

- Input/Output problems: Given appropriate inputs, compute output(s)
- Decision Problems: a problem whose output is YES/NO (or 1/0)

Examples

- Input/output problem: find the mean of n integers
  Decision Problem: Is the mean of the n integers equal to k ?
- Input/output problem: compute the cost of the shortest path between nodes u, v in a graph G
  Decision Problem: Is the cost of the shortest path between nodes u, v in a graph G equal to k ?

Note: You can solve the decision problem if and only if you can solve the input/output problem (Why?)

Languages and Decision Problems

- Decision Problem: output is YES/NO (or 1/0)
- Language: set of all inputs where output is yes
- So solving the decision problem is equivalent to checking if an input belongs in the language
- E.g.: Suppose we are given a method IsEven() that checks if a positive integer is even
  This solves the DECISION problem of checking if a given positive integer is an even number
  Define the language $L_{even} = \{2, 4, 6, \ldots\}$. A number is even if and only of it is in L_{even}

More Examples

- Even string length
  - I/O Problem:
    Input: String w
    Output: length of string w if w has even length, -1 otherwise
  - Decision Problem: Does w have even length?
    Input: String w
    Output: Yes, if $|w|$ is even, no otherwise
  - Language: Set of all strings of even length
- Code Reachability
  - I/O Problem:
    Input: Java computer code
    Output: Lines of unreachable code.
  - Decision Problem: Input: Java computer code and line number
    Output: Yes, if the line is reachable for some input, no otherwise
  - Language: Set of strings denoting Java code and reachable lines.

Relationship to Functions

- Use the set of k-tuples view of functions from before
- A function is a set of k-tuples (words) and therefore a language
- E.g.: All-pairs shortest paths in graphs – the set of triples of 2 nodes and the cost of the shortest paths between them is a set of paths (words) and therefore a language

Solving Problems: Multiple Views

- Normal (I/O) view: Given some input, compute the output
- Decision view: Given some input and possible output, return YES if the output given is correct and NO otherwise
- Language view: Given some word (tuple containing input and output), determine if it is part of the language corresponding to the problem
- Machine view: Given some word (tuple containing input and output), the machine indicates if it is part of the language corresponding to the problem or not

Terminology: Machine "decides" the language, by "accepting" and "rejecting" inputs