Automata Theory (SCS 2112)

Mathematics, rightly viewed, possesses not only truth but supreme beauty …

Bertrand Russell

SCS 2112

Recommended Reading:

- Introduction to Languages and The Theory of Computation, John C Martin, Mc Graw Hill

Evaluation Criteria

- Assignments : 30%
- Final Examination : 70%

Automata Theory

- The main emphasis of Automata theory is the study of definitions and properties of mathematical (computational) models (abstract models) that can be used for computations.

Sets/Sequences/Tuples/Pairs

- Set : A collection of objects.
  - A set may contain objects of different types.
  - Order of objects in a set is not important.
- Sequence : A collection of objects in a particular order.
  - Objects can be of any type.
  - Order is important.
- Tuple : A sequence of finite number of elements.

Sets/Sequences/Tuples/Pairs

- k-tuple: A sequence of k elements.
- Pair : 2-tuple.
- $A \times B$ – Cartesian product of A and B

$A \times B = \{(a,b) \mid a \in A \text{ and } b \in B\}$

Functions

- A function is a mapping between two sets satisfying certain constraints.
- $f : D \rightarrow R$
  - Domain of f is D
  - Range of f is R
- Different types of functions
  - Onto : A function that uses all values of the range
  - Into : A function that does not use all values of the range
  - 1-1

Alphabets

- Alphabet (Σ) : A finite nonempty set of symbols.
  - ASCII and EBCDIC, Unicode are examples of computer alphabets.
  - The members of the alphabet are called the **symbols** of the alphabet.

Strings

- String : Finite sequence of symbols from the alphabet.
  - A string is usually written by appending each symbol in the sequence next to one another.

Example

let $\Sigma = \{a, b\}$

then aa,abab are strings on Σ

  - Two strings are considered the same if all their letters are the same and in the same order.

String Operations

- Concatenation
- Reverse of a string
- Length of a string w,
- Sub-string : Any string of consecutive characters in some siting w
- Empty string, denoted by $\in(\lambda)$
- Prefix/suffix
- Σ^* : **Kleene closure**
- Σ^+
- Σ^*, Σ^+ are infinite

- Informally a language L over an alphabet Σ is a subset of Σ^*.
  - finite language on Σ
  - infinite language on Σ
  - Since a language is a set, all set operations can be applied on languages.

- Since languages are sets, the union, intersection, and difference of two languages are automatically defined.
- The complement of a language is defined with respect to Σ^*.

  $L' = \Sigma^* - L$

- Concatenation of two languages L_1 and L_2 contains every string in L_1 concatenated with every string in L_2.

  $$L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

- L^n is defined as the concatenation of L with itself n times

  $L^0 = \{\in\}$

  $L^1 = L$

- The star-closure of a language is defined as

  $$L^* = L^0 \cup L^1 \cup L^2 \ldots..$$

- The positive closure of a language is defined as

  $$L^+ = L^1 \cup L^2 \ldots..$$

- A string in a language L is called a **sentence** of L.

- Languages
  - Natural Languages
    - Difficult to define
    - Dictionary Definition : System suitable for expressing ideas, facts or concepts and rules for their manipulation.
  - Formal Languages
    - Defined precisely so that mathematical analysis is possible.

- Two basic problems in programming language design are
  - How to define a programming language precisely.
  - How to use such definitions to write an efficient and reliable translation programs.
- Theory of formal languages are extensively used in the
  - Definition of programming languages.
  - Construction of interpreters and compilers.

- How specific languages can be defined?
  - Listing out all possible words in the language, if the language is finite.
    - Example a Dictionary
  - Giving a set of rules, which defines all the acceptable words of the language.
  - A language L over an alphabet Σ is a subset of Σ^*. Thus set notations can be used to define languages. However set notation is inadequate to define complex languages.
  - Grammars : Powerful mechanism for defining formal languages.

- Formal languages
  - Alphabet (Σ) : A finite nonempty set of symbols.
  - Syntax : linguistic form of sentences in the language
    - Only concerned with the form rather than meaning
  - Semantics : Linguistic meaning of syntactically correct sentences.
    - A syntactically correct program need not make any sense semantically.

- Formally, a grammar is a four-tuple(N,Σ,P,S)
  - N : the set of non-terminal symbols or variables, denoted by capital letters
  - Σ : the set of terminal symbols (or, simply, terminals).
  - P : set of production rules.

    All production rules are of the form

    $v \to w$ where

    $v \in (N \cup \Sigma)^+$

    $w \in (N \cup \Sigma)^*$

    Production rules specify how the grammar transforms one string to another.

– S : A designated initial non-terminal from which all strings in the language are derived

Note :

– $\sum \cap N = \emptyset$

– $S \in N$

- **Derivations**

  Let w be a string of the form uxv

  i.e w = uxv

  and $x \rightarrow y$ is a production of the grammar.

  Then we say the production $x \rightarrow y$ is **applicable** to the string w, and may replace the occurrence of x in w by y.

  This is written as

  $$uxv \Rightarrow uyv$$

$uxv \Rightarrow uyv$

- We say uyv **derives** uyv or
- uyv is **derived from** uxv

We may derive new string from a given string by applying productions successively in arbitrary order.

$w_1 \Rightarrow w_2 \Rightarrow \ldots\ldots\ldots \Rightarrow w_n$

This can be given as $w_1 \Rightarrow^* w_n$

this means $\mathbf{w_1}$ **derives** $\mathbf{w_n}$

$w_{1,}\ w_2, \ldots\ldots\ W_n$ are called **sentential forms** of the derivation.

- Let G be a grammar. Then the language generated by G is denoted by L(G).
- Two grammars are said to be **equivalent** if they generate the same language.
  - Important in the development of parsers.
  - It is hard/impossible to develop parsers for some grammars.
  - They may be transformed into equivalent grammars that can be parsed.

Example :

The set of all legal identifiers in Pascal is a language.

Informal Definition : Set of strings with a letter followed by an arbitrary number of letters or digits.

Formal Definition : (Grammar)

<id> → <letter><rest>

<rest> → <letter><rest> | <digit><rest> | ∈

<letter> → a | b| c|…..|z

<digit> → 0|1|…..|9

Classes of Grammars

Chomsky's scheme of classification

Based on the format of the productions

assume productions are of the form $\alpha_i \rightarrow \beta_i$

– Type 0 : Phrase structure grammars

no restrictions on form of productions $\alpha_i \rightarrow \beta_i$ for all i

- All formal grammars.
- Generates all languages recognizable by a Turing machine

Chomsky's scheme of classification

- Type 1 : Context-sensitive grammars
  - $|\alpha_i| \leq |\beta_i|$ for all i , where | | denotes the length

    Note : null string would not be allowed as a right hand side of any production.

– Type 2 : Context free grammars (BNF Grammars)

$\forall \alpha_I$ restricted to a single non-terminal symbol, for all i

– Can be recognized by *pushdown automata*

– Context free grammar is a common notation for specifying the syntax of programming languages.

Example :

In C if-else statement

Stmt $\rightarrow$ **if** (expr) stmt **else** stmt

– Type 3 : regular grammars

- all production of the form $A \rightarrow xB$ or $A \rightarrow x$ where A and B are non-terminals and x is in $\sum^*$ - **right liner grammar.**
- all production of the form $A \rightarrow Bx$ or $A \rightarrow x$ where A and B are non-terminals and x is in $\sum^*$ - **left liner grammar.**
- Can be recognized by finite automata .

Note : type t grammars are also type t-1 for all $t > 0$

- A language L(G) is said to be of type k if it can be generated by type k grammar.

Church–Turing thesis(hypothesis)

- Church–Turing thesis states that if some method (algorithm) exists to carry out a calculation, then the same calculation can also be carried out by a Turing machine (as well as by a recursively-definable function, and by a λ-function).
- Though this hypothesis cannot be formally proven, it has a near-universal acceptance.

The trouble with the world is that the stupid are cocksure and the intelligent are full of doubt.

Bertrand Russell