# Automata Theory

## SCS 2212

Dushani Perera

19/10/2020

# Introduction

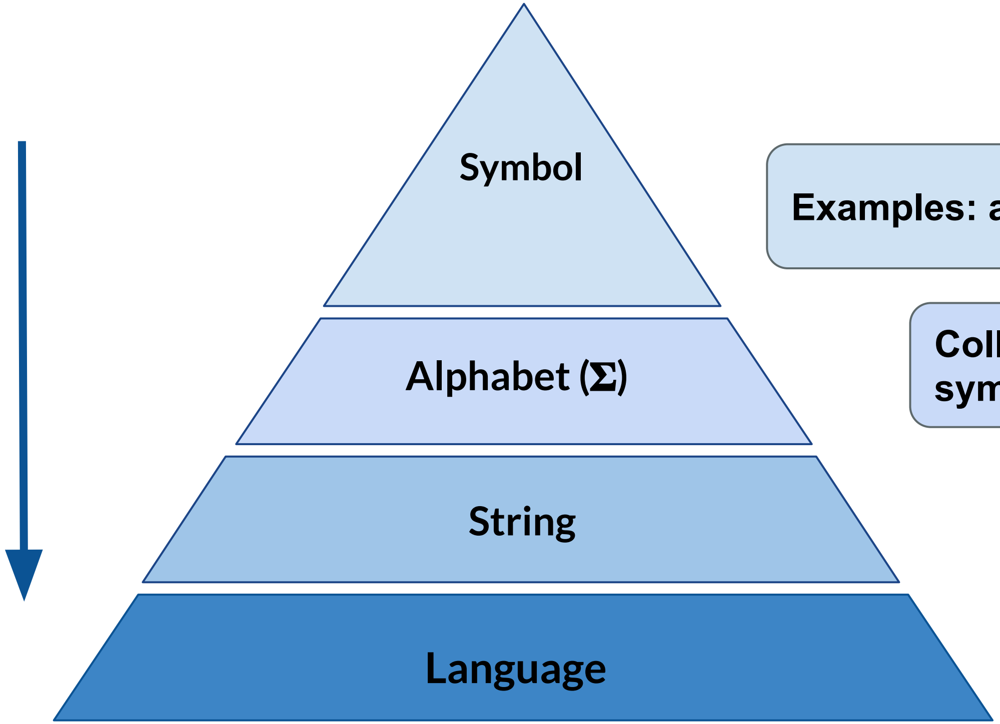
- Recommended reading:
  - Introduction to Languages and The Theory of Computation, John C Martin, Mc Graw Hill.
  - An Introduction to Formal Languages and Automata, Peter Linz, Narosha.
  - Introduction to Automata Theory, Languages and Computation, John E. Hopcroft, Rajeev Motwani, Jeffrey D Ullman, Pearson education.

- 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.

# Basics Of Automata Theory




Examples: a, b, c, 0, 1, etc.

Collection/set of finite nonempty symbols. Examples: {a, b}

Finite sequence of symbols on an alphabet. Example: a, b, aa

Collection of strings.
Example: language $L_1$ = set of all strings that start with ‘a’

# Mathematical Preliminaries

- Set :- Collection of symbols. If x is an element of set S then $x \in S$
- Sequence :- Ordered collection of symbols.
- Tuple :- Sequence of finite number of objects (finite ordered list).
    - k-tuple :- sequence of k elements
        - 2 - tuple :- sequence of 2 elements (this is called a 'Pair')
- Cartesian Product :- A x B
    - If A = {a, b} and B= {c, d} then;
        - A x B = {(a,c), (a,d), (b,c) ,(b,d)}
        - A x B = {(a, b) | $a \in A$ and $b \in B$}

# Mathematical Preliminaries Cont.

- Functions :- 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

# String Operations

- Concatenation :- link strings together
    - $\Sigma_1 = \{a, b\}$ $\Sigma_2 = \{c, d\}$ ac, ad, bc, bd
- Reversal
    - $w = a_1a_2a_3 \dots ,a_n \Rightarrow w^R = a_na_{n-1}a_{n-2} \dots ,a_1$
- Length of a string $|w|$
- Substring
    - w= {a, b} substrings = {ε, a, b, ab}
- Empty string (ε)
    - Length = 0
    - $\Sigma^0 = \{\varepsilon\}$

# Lengths Of Possible Strings

- Let's say the Alphabet $\Sigma$ = {a, b} $|\Sigma| = 2$
  - How many strings of length 2 are possible on $\Sigma$ ? 4 strings - aa, ab, ba, bb ($2^2$)
  - How many strings of length n are possible on $\Sigma$ ? $2^n$

- Number of strings of length n possible on an alphabet $\Sigma$ is $|\Sigma|^n$

# Powers Of Σ

- Let's say the alphabet $\Sigma = \{a, b\}$ $|\Sigma| = 2$
  - $\Sigma^1$ :- set of all strings that can be formed on the alphabet which are of length 1.
    - Answer set of strings = {a, b} $(2^1)$
  - $\Sigma^2$ :- set of all strings that can be formed on the alphabet which are of length 2.
    - Concatenation as $\Sigma\Sigma$
    - Answer set of 4 strings = {aa, ab, ba, bb} $(2^2)$
  - $\Sigma^3$ :- set of all strings that can be formed on the alphabet which are of length 3.
    - Concatenation as $\Sigma\Sigma\Sigma$
    - Answer set of 8 strings = {aaa, aab, aba, abb, baa, bab, bba, bbb} $(2^3)$
  - $\Sigma^0$ :- set of all strings that can be formed on the alphabet which are of length 0.
    - $\Sigma^0 = \{\varepsilon\}$
  - $\Sigma^n$ :- set of all strings that can be formed on the alphabet which are of length n.
    - Concatenation of $\Sigma$ with itself n times

# Powers Of Σ Cont.

## $\Sigma^*$

- $\Sigma^*$ = set of all strings of all lengths possible on the alphabet $\Sigma$
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots\ldots$
- $\Sigma^* = \{\varepsilon\} \cup \{a, b\} \cup \{aa, ab, ba, bb\} \ldots\ldots$
- Kleene closure / star closure

## $\Sigma^+$

- $\Sigma^+ = \Sigma^* - \{\varepsilon\}$
- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \ldots\ldots$
- $\Sigma^+ = \{a, b\} \cup \{aa, ab, ba, bb\} \ldots\ldots$
- Positive closure

# Languages

- **If** $L_1, L_2, L_3$ are languages defined on the alphabet $\Sigma = \{a, b\}$ as:
  - $L_1$ = set of all strings of length 2 (finite language)
  - $L_2$ = set of all strings of length 3 (finite language)
  - $L_3$ = set of all strings on $\Sigma$ where all strings start with ‘a’ (infinite language)
    - {a, aa, ab, aaa, aab, aba, abb………}

**And**

- **If** $\Sigma^*$ is the set of all strings of all lengths possible on the alphabet $\Sigma$, **then**:
- $L_1 \subseteq \Sigma^*$, $L_2 \subseteq \Sigma^*$, $L_3 \subseteq \Sigma^*$

# Languages Cont.

- Language is a subset of strings that belong in the set of all strings defined on an alphabet.
- Since languages are sets, the union, intersection, and difference of two languages are defined.
- The complement of a language L is defined with respect to $\Sigma^*$.

$L' = \Sigma^* - L$

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

$L_1 L_2 = \{x\, y \mid x \in L1, y \in L2\}$

# Languages Cont.

- 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.

# Languages Cont.

- Two basic problems in programming language designs are:
  - How to define a programming language precisely ?
  - How to use such definitions to write an efficient and reliable translation program ?

- Theory of formal languages are extensively used in:
  - Definition of programming languages.
  - Construction of interpreters and compilers.

# Powers of a Language L

- $L^n$ is defined as the concatenation of L with itself n times.
- $L^0 = \{\varepsilon\}$
- 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.

# Methods Of Defining Languages

- Definition methods:
  - Listing out all possible words in the language, if the language is finite. Eg:- Dictionary
  - Giving a set of rules, which defines all the acceptable words of the language.
- A language L over an alphabet $\Sigma$ is a subset of $\Sigma^*$. Thus, set notations can be used to define languages. However, set notations are inadequate to define complex languages.
- Therefore, Grammars are used.
  - Grammar :- Powerful mechanism for defining formal languages.

# Formal Language

- A formal language has:
  - Alphabet ($\Sigma$) : 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.

# Definition Of a Grammar

- Grammar G is defined as : G = (V, $\Sigma$, S, P)
  - V : finite set of objects called variables (non-terminals, denoted by capital letters)
  - $\Sigma$ : finite set of objects called terminal symbols
  - S : Initial non-terminal deriving symbol/ start symbol ($S \in V$)
  - P : finite set of productions

# Definition Of a Grammar Cont.

- P : set of production rules.
    - All production rules are of the form
    - $v \longrightarrow w$ where
        - $v \in (V \cup \Sigma)+$
        - $w \in (V \cup \Sigma)^*$
- Production rules specify how the grammar transforms one string to another.
    - Let w be a string of the form uxv ; i.e. w = uxv and $x \longrightarrow y$ is a production of the grammar.
    - Then we say that production is applicable to the string w, and may replace the occurrence of x in w by y.
    - This is written as $uxv \longrightarrow uyv$

# Definition Of a Grammar Cont.

$uxv \longrightarrow uyv$

- 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.

$w_1 \longrightarrow w_2 \text{ ……… } \longrightarrow w_n$

- This can be given as $w_1 \longrightarrow^* w_n$

This means $w_1$ derives $w_n$

- $w_1$, $w_2$,……. $w_n$ are called ‘sentential forms’ of the derivation.

# Sentential Forms

- A sentential form is the start symbol S of a grammar or any string in $(V \cup \Sigma)^*$ that can be derived from S.
- Consider the following linear grammar.
  - $G = (\{S,B\}, \{a,b\}, S, \{S \rightarrow aB, S \rightarrow B, B \rightarrow bB, B \rightarrow \in\})$
  - Derivation of the above grammar :
  - $S \Rightarrow aS \Rightarrow aB \Rightarrow abB \Rightarrow abbB \Rightarrow abb$
  - Each of {S, aS, aB, abB, abbB, abb} is a sentential form. Since this grammar is linear, each sentential form has at most one variable (Non terminal). Hence there is never any choice about which variable to expand next.

# Definition Of a Grammar Cont.

- 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 Definition

- 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

# Noam Chomsky: Classification of Grammars

| Grammar Type | Grammar Accepted | Language Accepted | Automaton |
|---|---|---|---|
| Type 0 | Unrestricted grammar | Recursively enumerable language | Turing Machine |
| Type 1 | Context-sensitive grammar | Context-sensitive language | Linear-bounded automaton |
| Type 2 | Context-free grammar | Context-free language | Pushdown automaton |
| Type 3 | Regular grammar | Regular language | Finite state automaton |

# Chomsky Classification of Grammars

- Type - 0 Grammar
  - The productions have no restrictions.
  - They are any phase structure grammar including all formal grammars.
  - The productions can be in the form of $\alpha \rightarrow \beta$ where $\alpha$ is a string of terminals and nonterminals with at least one non-terminal and $\alpha$ cannot be null. $\beta$ is a string of terminals and non-terminals.
  - Example

$S \rightarrow ACaB$

$Bc \rightarrow acB$

$CB \rightarrow DB$

$aD \rightarrow Db$

# Chomsky Classification of Grammars

- Type - 1 Grammar
  - Type-1 grammars generate context-sensitive languages.
  - The productions must be in the form $\alpha\ A\ \beta \rightarrow \alpha\ \gamma\ \beta$ where $A \in V$ (Non-terminal) and $\alpha, \beta, \gamma \in (\boldsymbol{\Sigma} \cup V)^*$ (Strings of terminals and non-terminals).
  - The strings $\alpha$ and $\beta$ may be empty, but $\gamma$ must be non-empty.
  - The rule $R \rightarrow \varepsilon$ is allowed if R does not appear on the right side of any rule. The languages generated by these grammars are recognized by a linear bounded automaton.
  - Example

$AB \rightarrow AbBc$

$A \rightarrow bcA$

$B \rightarrow b$

# Chomsky Classification of Grammars

- Type - 2 Grammar
  - The productions must be in the form $A \rightarrow \gamma$ where $A \in V$ (Non terminal) and $\gamma \in (\Sigma \cup V)^*$ (String of terminals and non-terminals).
  - These languages generated by these grammars are recognized by a non-deterministic pushdown automaton.
  - Example

$S \rightarrow X a$

$X \rightarrow a$

$X \rightarrow aX$

$X \rightarrow abc$

$X \rightarrow \varepsilon$

# Chomsky Classification of Grammars

- Type - 3 Grammar
  - Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal.
  - The productions must be in the form $X \rightarrow a$ or $X \rightarrow aY$ where $X, Y \in V$ (Non terminal) and $a \in \Sigma$ (Terminal).
  - The rule $R \rightarrow \varepsilon$ is allowed if R does not appear on the right side of any rule.
  - Example

$$X \rightarrow \varepsilon$$

$$X \rightarrow a \mid aY$$

$$Y \rightarrow b$$

# Chomsky Classification of Grammars

- Type - 3 Grammar Cont.
  - All production of the form $A \rightarrow aB$ or $A \rightarrow a$ where A and B are non-terminals and 'a' is in $\Sigma^*$
    - Right linear grammar.
  - all production of the form $A \rightarrow Ba$ or $A \rightarrow a$ where A and B are non-terminals and 'a' is in $\Sigma^*$
    - Left linear grammar.

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