CSE211 - Formal Languages and Automata Theory

U1L2: Basics of Formal Language

Dr. P. Saravanan
School of Computing
SASTRA Deemed University

Agenda

- Recap of previous class
- What is Formal Language?
- Relation between the formal language and Automata
- Alphabet
- String
- Language

Importance of the Course

- The course forms the basis for:
  - Writing efficient algorithms that run in computing devices
  - Programming language research and their development
  - Efficient compiler design and construction
- Focusing on
  - Automata Theory
  - Computability Theory
  - Complexity Theory

Unit 1

- Introduction: Alphabet, languages and grammars, productions and derivation, Chomsky hierarchy of languages
- Regular languages and finite automata: Deterministic Finite Automata (DFA) - Nondeterministic Finite Automata (NFA) – Finite Automata with Epsilon Transitions – Regular Expressions - Finite Automata and Regular Expressions - Kleene’s theorem- Regular grammars and Equivalence with Finite Automata - Properties of Regular Languages: Proving Languages Not to Be Regular–Closure Properties of Regular Languages - Myhill-Nerode theorem -Minimization of Finite Automata

Unit II

- Context-free languages and pushdown automata: Context-free grammars (CFG) – Parse Trees - Ambiguity in Grammars and Languages - nondeterministic pushdown automata (PDA) - Equivalence of PDAs and CFGs - Deterministic Pushdown Automata - Properties of Context Free Languages: Normal Forms for Context Free Grammars - Chomsky and Greibach normal forms - Pumping lemma for context-free languages - closure properties of CFLs

Unit III

- Context-sensitive languages: Context-sensitive grammars (CSG) and languages - linear bounded automata and equivalence with CSG
- Introduction to Turing machines: The Turing Machine (TM) - Church-Turing thesis Programming Techniques for Turing Machines – extensions to the Basic Turing Machine – Restricted Turing Machine - Turing recognizable (recursively enumerable) and Turing-decidable (recursive) languages and their closure properties, variants of Turing machines, nondeterministic TMs and equivalence with deterministic TMs, unrestricted grammars and equivalence with Turing machines, TMs as enumerators

Unit IV

- Undecidability: A Language That Is Not Recursively Enumerable (RE)- Diagonalization languages - An Undecidable Problem that Is RE - Universal Turing machine - undecidable problems about Turing Machines - Reductions between languages and Rice s theorem
- Basic Introduction to Complexity: Intractable Problems - Introductory ideas on Time complexity of deterministic and nondeterministic Turing machines – Classed P and NP, An NP- complete Problem - Cook's Theorem - Additional NP -Complete problems

Dr.PS

Text books

- John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Pearson, 3rd Edition, 2011.
- Peter Linz, An Introduction to Formal Languages and Automata, Jones and Bartle Learning International, United Kingdom, 6th Edition, 2016.

LETS START...

Automata theory & Formal Languages

- How do we formalize the question

Can device A solve problem B?

- First, we need a formal way of describing the problems that we are interested in solving
- Called Formal language.
- For formal description we need
  - Alphabets
  - Strings
  - Language

Dr.PS

Example problems

- Examples of problems we will consider
  - Given a word s, does it contain the subword "SASTRA"?
  - Given a number n, is it divisible by 5?
  - Given a pair of words a and b, are they the same?
  - Given an expression with brackets, e.g. (( ) ( )) , does every left bracket match with a subsequent right bracket?
- All of these have "yes/no" answers.
- There are other types of problems, that ask "Find this" or "How many of that".

Alphabets and strings

- An alphabet is a finite, nonempty set of symbols
- A common way to talk about words, number, pairs of words, etc. is by representing them as strings
- To define strings, we start with an alphabet.

An alphabet is a finite set of symbols.

- Examples

$\Sigma_1 = \{a, b, c, d, \ldots, z\}$: the set of letters in English

$\Sigma_2 = \{0, 1, \ldots, 9\}$: the set of (base 10) digits

$\Sigma_3 = \{a, b, \ldots, z, \#\}$: the set of letters plus the special symbol $\#$

$\Sigma_4 = \{\, (, \,)\, \}$: the set of open and closed brackets

Dr.PS

Strings

A string over alphabet Σ is a finite sequence of symbols in Σ.

university is a string over $\Sigma_1 = \{a, b, c, d, \ldots, z\}$

#include is a string over $\Sigma_3 = \{a, b, \ldots, z, \#\}$

main() is a string over $\Sigma_4 = \{a, b, c, d, \ldots, z, (,)\}$

- The empty string is the string with zero occurrences of symbols, denoted by ε
- Length of a String is the number of symbols in the string. The standard notation for the length of a string w is |w| For example |SASTRA|=6

Dr.PS

Power of an Alphabet

- If Σ is an alphabet, the set of all strings of a certain length from that alphabet by using an exponential notation.
- We define Σ^k to be the set of strings of length k each of whose symbols is in Σ
- Note that $\Sigma^0 = \varepsilon$ regardless of what alphabet Σ is
- If $\Sigma=\{a,b\}$ then
  - $\Sigma^1=\{a, b\}$,
  - $\Sigma^2=\{aa,bb,ab,ba\}$
  - $\Sigma^3=\{aaa,bbb,aab,abb,bab,bba,aba,baa\}$ and so on…

Dr.PS

Power of an Alphabet…

- The set of all strings over an alphabet Σ is conventionally denoted Σ^*
- For instance,
  - $\Sigma^* = \{a, b\}^* = \{\varepsilon, a, b, aa, bb, ab, ba, aaa, bbb, aab, abb, bab....\}$
  - $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 ....$
  - $\Sigma^+ = \{a, b\}^+ = \{a, b, aa, bb, ab, ba, aaa, bbb, aab, abb, bab....\}$
  - $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 ....$
- Σ^* is a language for any alphabet
- ϕ the empty language is a language over any alphabet

Dr.PS

Concatenation of Strings

- Let x and y be strings. Then xy denotes the concatenation of x and y
  - Let x =001100 and y=111, then
  - xy= 001100111 and yx=111001100
- For any string w
  - $\varepsilon w = w\varepsilon = w$. That is ε is the identity for concatenation
- L={ε}, the language consisting of only the empty string is also a language over any alphabet
- $\phi \neq \{\varepsilon\}$, the former has no strings and the latter has one string

Dr.PS

Languages

A language is a set of strings over an alphabet.

- Languages can be used to describe problems with "yes/no" answers
- For example:

$L_1 =$ The set of all strings over Σ_1 that contain the substring "SASTRA"

$L_2 =$ The set of all strings over Σ_2 that are divisible 5
$= \{5,10,15,20, \ldots\}$

$L_3 =$ The set of all strings of the form $s\#s$ where s is any string over $\{a, b, \ldots, z\}$

Dr.PS

Structural Representations

- There are two important notations that are not automaton but used languages
- They play an important role in the study of automata and their applications
  - Grammars: Useful models when designing software that processes data with a recursive structure
    - Ex: Parser uses grammars for parsing the string
  - Regular Expressions: Denote the structure of data especially text strings
    - [A-Z][a-z]*

Dr.PS

Formal Language

- Formal language – is specified by well-defined set of rules of syntax
- We describe the sentences of a formal language using a grammar
- Formal languages provide models for both natural languages and programming languages

Summary

- Automata and Formal Languages
- Alphabets
- String
- Language

References

- John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Introduction to Automata Theory*, Languages, and Computation, Pearson, 3rd Edition, 2011.
- Peter Linz, An Introduction to Formal Languages and Automata, Jones and Bartle Learning International, United Kingdom, 6th Edition, 2016.

Dr.PS

Next Class:

Grammars and Derivation

THANK YOU.