Automata Theory - Lecture 5

Kleene's Theorem, Symbolic Logic and Formal Systems

Lecturer: Martha Gichuki

Lecture learning outcomes

At the end of the lecture you will be able to:

(i) Define Kleene's Theorem.

(ii) Describe Formal Systems.

(iii) Differentiate symbolic logic and formal systems

5.1 Kleene's Theorem

From lecture one, we indicated that Automata Theory is closely related to **formal language theory** because automatons are often classified by the class of formal languages they can recognize.

Recall that an automaton is anchored on the basic concepts of **symbols** (characters)**, words** (symbols) **alphabets** (set of symbols and letters) and **strings.** A **Language** is a set of words formed by symbols in each alphabet and according to Kleene ***Closure***, a language may be thought of as a sub-set of possible words. The set of all possible words may in turn be thought of as the set of all possible concatenations of strings denoted as ***** and the super script $^{(*)}$ is called the **Kleene Star**.

An automaton has a mechanism to **read inputs** which is a **string over a given alphabet**

Kleene's theorem states that Finite Automatons accept regular languages and regular expressions are used to describe regular languages. A set of regular expressions is defined over an alphabet Σ recursively and all the elements of that set is from **regular expression**s.

5.2 Symbolic Logic

Symbolic logic provides a way to express logical expressions by using symbols and variables instead of using natural languages such as English, in order to remove vagueness.

What are logical expressions?

Recall from Lecture 3 that logical statements are statements with a truth value: either true or false however, there is no need of knowing whether a logical expression is true or false, all we need to know is that it has a truth value.

Consider a question like 'What are you eating?' This has no truth value.

Consider a statement like "It is raining" This expression is either true or false.

Propositions:

The smallest logical expression possible, which, if broken down further loses meaning is called a **proposition**.

Example of a proposition:

'James and John live together' cannot be broken down without a loss in meaning. 'James lives together' doesn't even make sense.

Example:

'Joyce and Joy go to work' can be broken into 'Joyce goes to work' and 'Joy goes to work,' There is no evidence showing that Joyce and Joy go to work at the same time.

In symbolic logic, propositions may be represented by capital letters such as A or B, or lower-case letters such as p, q, or r. Example 'My house is big' may represented by A, and 'My puppy is white' may be represented by B.

Propositions are written in the affirmative, for example we use the not symbol (~) to make a **negation** (a not statement) instead of writing 'my puppy is not white', we write using symbols, as follows: ∼ B. Therefore, logically, negation changes an expression's truth value e.g. if 'my puppy is white', then B would be true, and ∼ B would be false, or if my house is small, then A would be false, and ∼ A would be true.

Truth Tables

From Lecture 3 recall that a **truth table** lists whether something is true or false and as stated above, the act of negation changes all the true values into false and the false values to true as shown below.

P	~ P
True	False
False	True

To construct a truth table, a column for every proposition in the expression must be created, and for every possible true and false value, a row is created showing all the possible combination that a given set of propositions can take.

Example: For two propositions, a truth table for all the possible truth value combinations will have four rows for T/F possibilities and two columns for the propositions. In general, there will be **2^n rows for *n* different propositions**.

A	B
True	True
True	False
False	True
False	False

Logical operators

These are needed to link together propositions and build more complex logical expressions and they operate in the same way as +, -, ×, and ÷ operators used to link mathematical expressions. The basic logical operators, along with **negation**, are **conjunction**, **disjunction**, **conditional**, and **biconditional**.

5.3 Formal Systems

Formal systems are also known as logistic systems in abstract terms and they have a formal language comprising of symbols acted on by certain rules allowed in the system. The rules are developed from axioms therefore formal systems have inbuilt formulas that use finite combinations of primitive symbols.

Model structure are used to interpret the symbols of a formal system and often used in conjunction with formal systems.

What constitutes formal systems?

A formal system has the following components: -

(i) A **finite alphabet** of symbols, finite because we need precise models.
(ii) A **syntax** that defines which strings of symbol are in the language of our formal system.
(iii) A **decidable** set of **axioms** and a finite set of **rules** from which the set of **theorems** of the system is generated and the rules must take a finite number of steps to apply.

Review Questions

1. For any alphabet ∑, the set of all strings over ∑ is denoted by ∑*, suppose ∑ is in the two-symbol alphabet {a, b}, what are the possible subsets of ∑*

Possible subsets= {Ø, {a}, {b}, {ab}, {ba}, {aa}, {bb}, {aba}, {bab}, {abba}, {baab}, {bbaa}, {aabb}, ...}

2. Describe the following sets by regular expressions:

   a) {01, 10}

   Set of alternate zeros and ones in pairs

   b) {101}

   Set of triple palindromes of zeros and ones

   c) {λ, 1,1,1,1,1, 1…}

   Set of single digits of one with λ included

   d) {1,1,1,1,1,1, 1…}

   Set of single digits of one

3. If C is a set with *c* elements, how many elements are in the *power set of C*? Explain your answer.

   Power set of C has **$2^{|C|}$** ***where*** **|C| is the cardinality (length) of C**

4. If A has *a* elements and B has *b* elements, how many elements are in *A X B*? Explain your answer.

5. Design a machine, **M_2** which accepts the following language:

   {W∈{a,b}*: W has an odd number of a's and an even number of b's}

References

1. Rowan G. & John T., (2009), *Discrete Mathematics: Proofs, Structures and Applications*, CRC Press, ISBN: 9781439812808.
2. W. D. Wallis (2003), *A Beginners Guide to Discrete Mathematics*, Springer Science & Business Media, ISBN: 978-0817642693.
3. Introduction to the theory of computation (3rd ed.), Michael, S. Boston, Cengage Learning. ISBN-13: 978-1133187790, (2012).
4. Introduction to languages and the theory of computation (3rd ed.), Martin, J., New York: McGraw-Hill. ISBN-13: 978-0072322002, (2002)