Propositional Logic for Knowledge Representation and Formalization of Reasoning

Kurdman Abdulrahman Rasol

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
June 2017
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Mustafa Tümer
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Rashad Aliyev
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev
2. Asst. Prof. Dr. Ersin Kuset Bodur
3. Asst. Prof. Dr. Müge Saadetoğlu

ABSTRACT

The purpose of this master thesis is to investigate the basic concepts of propositional logic for knowledge representation and formalization of reasoning in Artificial Intelligence.

The different properties of logical propositions are discussed. The basic and derived logical connectives are used to establish the compound statements, and the truth tables are constructed to investigate the properties of logical connectives. Such propositions as tautology, satisfiability, contradiction, contingency, logical entailment and logical equivalence are analyzed. Three algebraic normal forms - negation normal form, disjunctive normal form and conjunctive normal form are studied. Horn clauses are implemented. Two forms of valid inferences as modus ponens and modus tollens are considered. Some examples are provided to better understand the main properties of propositional logic.

Keywords: Propositional Logic, Logical Connectives, Normal Forms, Horn Clauses, Modus Ponens, Modus Tollens

ÖZ

Bu tezin amacı önermeler mantığının yapay zeka alanında bilgi gösterimi ve akıl yürütme biçimselliştirmesi için temel kavramları incelemektir.

Mantık önermelerinin farklı özellikleri tartışılır. Temel ve türetilmiş mantık bağlaçları kullanarak bileşik önermeler oluşturulur, ve doğruluk tabloları kurarak mantık bağlaçlarının özellikleri araştırılır. Totoloji, tatmin edilebilirlik, çelişki, beklenmedik durum, mantıksal gerektirme ve mantıksal denklik gibi önermeler incelenir. Üç cebirsel normal form - olumsuzluk normal formu, ayırıcı normal formu ve bağlayıcı normal formları irdelenir. Horn cümlecikleri uygulanır. Modus ponens ve modus tollens gibi iki geçerli sonuç çıkarma yöntemleri incelenir. Önermeler mantığının daha iyi anlaşılması için bazı örnekler verilir.

Anahtar Kelimeler: Önermeler mantığı, Mantık bağlaçları, Normal formlar, Horn cümlecikleri, Modus Ponens, Modus Tollens

ACKNOWLEDGMENT

First of all, thanks to Almighty God for peace and wisdom to enhance me to start and complete my master study.

I would like to express my gratitude to my supervisor, Prof. Dr. Rashad Aliyev for his continuous support and guidance. Prof. Dr. Rashad Aliyev is a reference of a professional person who helped me so much during this whole work and no words can express how thankful I am to him, and I am very much honored that I worked under his supervision.

Last but not least, I would like to thank my parents, Mr. Abdulrahaman Rasol and Mrs. Qomri Khaled with whom I have such successful life and proud of both of them. I thank my brothers and sisters, and all my friends. I appreciate all the times they visited, called, chatted and e-mailed with words of faith, encouragement and wisdom that carried me daily through this process. Thanks to my brother Mezgeen Rasol for his encouragements in whole my life.

Finally, I would like to express my appreciation to everyone who made these years in North Cyprus a wonderful experience for me.

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1

INTRODUCTION

Artificial Intelligence (AI) is a branch of computing and engineering that is used in many areas and mimics the human intelligence and behavior, and it is concerned with simulation of intelligent actions in computers. So AI is a study of “intelligent agents” in which devices are used to identify the environment to boost. AI is very useful to let machines think and act like human being to solve different complex problems.

The study of principles of correct analysis is known as logic and it is derived from the Greek word “logos” which means sentence, reason, thought, rule, ratio or what is spoken and contains a systematic study of the form of arguments. When there is a connection between expectations of arguments and conclusion or result then it will be called a valid argument which is very important in logic.

There are different types of logic known as formal, informal, symbolic and mathematical logics.

Representing and reasoning with knowledge is playing a central role in AI. Knowledge representation is committed to representing information about the world in such a way that a computer system can be able to solve complicated tasks. It is very difficult to talk to experts in any system or organization in terms of codes or organizational ways rather than easy and straightforward daily talks.

Knowledge is information about any specific area to solve different kind of problems within this area, and knowledge is used to solve a number of problems and it must be mentioned that knowledge should be represented by a computer. The knowledge is known as representation scheme, and representation is any kind of knowledge that shows the inner and internal workout and structure of data and its output, and when there is a complete representation of knowledge then it is stored by an intelligent agent.

All knowledge representation languages are based on different types of logic. Propositional logic is an excellent tool to represent knowledge in many AI problems. Propositional logic is a branch of logic which deals with ways of accompanying or adjusting all propositions, statements or sentences to create more complicated (compound) propositions. The simplest statements are treated as individual units, so propositional logic does not study those logical properties.

The history of logic starts from Aristotle (384-322 BC), but Aristotle's practical writing about logic is related to a logic of categories and quantifiers like "all" and "some" which are not conducted in propositional logic. Aristotle adopted two rules for great importance in propositional logic which are called "Law of Excluded Middle" and "Law of Contradiction".

In later 3rd century BCE a serious study about logical operators like "and", "or" and "if…then…" was conducted by Stoic philosophers. Stoic philosopher Chrysippus

(280-205 BCE) represented a work related to propositional logic by making out a number of different ways of making complicated boundaries for arguments. The basic reasoning design like Diagenes Laertius, Aextus Empirius was presented by Chrysippus and other philosophers.

Some advanced works on Stoics were launched in the form of small steps in later centuries. Some works were done by Galen (129-210 BCE), Boethius (480-525 BCE), Peter Abelard (1079-1142 BCE), William of Ockham (1288-1947 BCE). Augustus DeMorgan (1806-1871 BCE), George Boole (1815-1864 BCE) and Gottlob Frege (1848-1925 BCE) presented some advanced works.

Propositional logic has specific syntax and semantic to be used for designing proposition and clarifying its logic and meaning.

Every language has different and unique syntax and semantic rules regardless it's a real life language or a computer language.

Syntax of proposition in propositional logic is the adjustment of words and expression to design and make well-formed sentences in any language; in other words, syntax is a set of rules to design a sentence in a particular language. Nouns, pronouns, adjectives, verbs etc. make a sentence but syntax is used to adjust every part to make proper sentence which has a meaning. To understand it better, let's consider the following combination of words: "I good at studying am university very a"; this combination of words does not form any meaning, but if syntax is to be applied on these words, a meaningful sentence will be like "I am studying at a very good university".

Syntax is also known as grammar for language and propositional logic; there are some components which are used in grammar for propositional logic: connectives (operators), grouping symbols etc.

The meaning of the sentence is defined by the semantics of a propositional logic. The interpretation is related to semantics of a propositional logic. The interpretation is determined by the sentence or propositional variable. The semantics can be used to represent different statements and their Boolean values - true or false values about the world.

The interpretation of the sentence or formula in a propositional logic can be defined in the following forms: a sentence becomes satisfiable if it is true in some interpretations; a sentence becomes unsatifiable if it is true in no interpretation; and a sentence is valid if it is true in all interpretations.

The atomic propositions which can be represented by true or false logical values can be combined for some reasons. The symbols like $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ are logical connectives or operators which are used in propositional logic. By using the logical connectives it is possible to establish propositional formulas. Well-Formed Formula (WFF) can be constructed by using single atomic propositions represented by capital letters A,B,C,…,Z, their negations, and joining any two single atomic propositions by using the logical connectives. WFF makes the statement meaningful and unambiguous.

One of the useful tools to analyze a propositional logic is the truth table which has a tabular form of representation of all possible combinations of true or false values of the input statements resulting in finding the truth value of the output compound statement; so several atomic statements can be combined into a single statement by using logical connectives, and there is a distinct truth table for each logical connective.

Truth table is also effective way to represent such compound propositions as tautology, contradiction, contingency and logical equivalence.

Modus Ponens and Modus Tollens are two rules of inference for propositional logic and they are described in Schematic Form.

Chapter 2

REVIEW OF EXISTING LITERATURE ON PROPOSITIONAL LOGIC

The paper [1] presents a finite predicate logic corresponding to the fragment of first order logic, and this fragment allows usage of equality and quantification. The model of this logic can be interpreted as Herbrand universe. The formulae of this logic can be effectively translated to the Bernays-Schonfinkel formulae. The proposed logic helps optimally encode planning problems. In comparison with the propositional approach, the reasoning with the presented logic is exponentially more efficient. Moreover, the proposed method is also effective to find optimal form of propositional encodings.

In [2] the resolution approach in a linguistic propositional based on truth values is proposed to establish main concept of symmetrical refined hedge algebra which is effectively used for the application to the intelligent reasoning system. The syntax and semantic of the linguistic propositional logic are determined. The linguistic information processing focusing on linguistic variables is supported. The resolution rule and procedure for soundness and completeness results are investigated.

A generalization of propositional logic approach which allows using the truth values in the continuous interval 0 and 1 is described in [3], and this generalized form is called Minimal Polynomial Logic (MPL). The representation of propositions in MPL is done using multi-variate polynomials. The simplest form of the polynomial logic denoted as PL_0 is also studied. This study shows that PL_0 is an efficient logic to prove some particular results about classical logic. Both MPL and PL_0 are implemented and the relationships of both logic techniques with the concepts of fuzzy logic and probabilistic logic are determined.

Argumentation has been one of the increasingly core studies in human reasoning in Artificial Intelligence. Argumentation has also been one of the objectives to be used for bringing together the propositional logic and non-monotonic reasoning in one unified frame. The paper [4] aims to investigate the connection between argumentation and propositional logic. This connection is useful for supporting a non-monotonic reasoning in Artificial Intelligence. A new logic of arguments which is called Argumentation Logic (AL) is formulated. The relation of argumentation logic with the propositional logic is studied, and the properties of argumentation logic are investigated.

The effectively propositional logic is a fragment of first-order predicate logic to be translated into propositional logic. Both propositional resolution and resolution with generalization are used for effectively propositional logic in [5]. A generalization rule shows that in comparison with resolution, the suggested resolution with generalization has exponentially shorter proofs. The sort assignments for the generalization process are provided.

Constructing the plausible mathematical model for understanding human reasoning and intelligence is important in Artificial Intelligence. The intelligent agents with abilities of deductive and inductive reasoning are developed by using computational model [6]. The agents can learn a generalized knowledge and arithmetic and propositional logic from examples. The agents are able to learn syntax and logical laws. The performance of the agent overcomes the average human performance in terms of accuracy. The proposed model is implemented using Haskell functional programming language. The learning process is described in case of propositional logic. The application areas of the presented agents are theorem proving, intellectual pedagogical systems etc.

The contexts were firstly introduced as a solution mechanism of the generality problem in artificial intelligence. The paper [7] studies the logical properties of contexts. The propositional language of context is investigated in terms of its syntax and semantics. The proof system based on Hilbert style is considered for general propositional language to provide the optimal results of the soundness and completeness of the system.

The paper [8] is about the controllability and definability as two strong forms of dependence involving variables in propositional knowledge base. The computational complexity issues are analyzed. It is mentioned that the controllability is a weaker form of dependence compare to definability. The applications of both forms are done, and one of the applications is decision under incomplete knowledge which is related to high complexity of controllability, and another application is the hypothesis discrimination.

To solve an extremely hard problem in the sports scheduling domain, a new and extremely competitive approach called the round robin problem is offered in [9]. The combination of a traditional propositional logic based problem encoding and local search satisfiability algorithms provides a determination of feasible schedules for the round robin problem much faster in comparison with other existing methods used for the same problem. The performed experiments for solving the round robin problem for different number of teams show the suitability of scheduling after combining the above two approaches.

The paper [10] introduces the linear propositional logic system using true value domain from linear symmetrical hedge algebra. The syntax, semantics, and inference of this logic system are presented. The resolution inference rule and resolution principle are defined for this logic system. The t-norm and t-conorm operations of Gödel are used to define the logical connectives for a linear propositional logic. The resolution inference rule is obtained by implementing the concept of reliability. The clauses of inference rule which have linguistic truth values with contradictory nature are defined. The maximal reliability of each clause of inference rule is determined.

The paper [11] considers the rules for developing strategies for students to solve propositional logical exercises in learning environment called LogEx. This learning environment intends rewriting a logical formula in normal form. LogEx provides a development of a domain reasoner for logic, and students can reach such learning goals as recognizing and applying rules correctly, proving the equivalence of formulae etc.

Dalal's approximation strategy has some limitations for full propositional logic. The family of logics Limited Bivalence is used for approximation of classical logic [12]. Each approximation step is realized in a polynomial time.

Cellular Automata with binary states on monoids is studied in [13], and it enables using formulae in propositional logic. The main properties of transition functions for Cellular Automata are defined. The multiplication of formulae on monoids is performed, and the basic properties of the multiplication are discussed.

The propositional discourse logic (PDL) is one of the modern normal forms of propositional logic establishing the facts about truth or falsity of the statements in a finite discourse, and is related to the structure of natural discourse logic [14]. The main properties of PDL is making reasoning in the presence of inconsistency by minimizing the number of assumptions, and avoiding a problematic behavior of logical connectives in case of having paradox situations. The directed graphs represent the referential structure of discourses.

In [15] the propositional logic of imperfect information also called IF-propositional logic is defined in terms of extensive form semantic games. The partitional structure of these games under imperfectness of information is studied. The time consistency plays an ambiguous role in analyzing the games, and this qualification is especially useful when the same information set is visited several times during playing the game. The application of this type of propositional logic in quantum theory and quantum logic is specified.

An intelligent tutorial program called P-logic Tutor which plays double roles as educational tool and a research environment is described in [16]. The important properties of this program are to teach students the basic concepts of propositional logic and providing them abilities in theorem-proving concepts. The implementation of this tutorial system program is done in Java programming language.

In the paper [17] a fuzzy propositional modal logic (FPML) and its semantics called fuzzy Kripke semantics are presented. Besides describing the properties of the fuzzy reasoning procedure, the possibility for the realization of the reasoning procedure on computer is studied. The soundness and completeness of the FPML system are studied.

Fuzzy logic is an important tool for representation of human reasoning in the presence of uncertainty. The fuzzy interpretation of such logical connectives as conjunction, disjunction, negation, implication and bi-implication is presented in [18]. The fuzzy interpretation of logical connectives leads to the cases in which the set of fuzzy tautologies exactly coincide with the set of classical tautologies. The propositional fuzzy logic is based on t-norm. The classical propositional logic can also be modeled by fuzzy connectives.

Chapter 3

PROPOSITIONS AND LOGICAL CONNECTIVES

3.1 Proposition

Proposition is an interpretative sentence which is used to express the given statement, suggestion, idea, plan or opinion to represent true or false values, but a proposition can't be both true and false at the same time. For example, the statement "Tomorrow it will be rainy" can be true or false but not both at the same time.

It's very important to note that not any sentence can represent a proposition, for example, "The movie is interesting" or "Black roses are beautiful"; in these sentences it's not possible to give a single answer as yes or no, true or false, because maybe movie for some persons is interesting but for some other persons it doesn't have any interest at all, and maybe some people like black roses but some other people dislike them; so it's not a proposition.

The notations used for propositions are A,B,C,..,a,b,c,…, to describe the propositional variables and to represent the statements which will get one of the logical values: true or false. So it is to mention that True (T) and False (F) are two constants used in propositional logic.

3.1.1 Compound proposition

A proposition that can further be distributed into many parts such that each individual part acts like a proposition, is known as a compound proposition. In compound proposition, every individual part has a unique value which is either true or false. Some examples of compound propositions are as follows: if it will be rainy tomorrow then I will not attend my friend, otherwise I will take some snacks with tea, and if it will be sunny tomorrow then I will go to the beach.

3.1.2 Standard form of proposition

A proposition can be expressed in standard form which is also known as a symbolic form. For example, in sentence used above it can be observed that there are some small parts like A = It will be rainy tomorrow, B = I will not attend my friend, C = I will take some snacks with tea, D = It will be sunny tomorrow, E = I will go to the beach. To connect or join A, B, C, D and E with each other, some specific words like if, then, otherwise are used.

The above proposition can be represented in standard form as if A then B, otherwise C, and if D then E.

3.1.3 Categorical proposition

A proposition which is defined as a basic concept of Aristotle logic, and dealing with two or more classes of objects, is called a categorical proposition. It is a proposition which accepts or refuses all or some objects from one group carried by another group.

Categorical proposition normally contains such parts as “Object Term” which is a first category, “Predicate Term” is a second category, “Copula” which connects different parts of categorical proposition, and “Quantifies” which are the words used to determine the class of object associated with predicate class.

It should be noted that while considering a categorical proposition, some specific terms like ”all”, “no” and “some” can be used for all categorical propositions. The standard form of all of them will be as follows: All S are P, No S is P, Some S are P, Some S are not P.

3.2 Logical connectives

To link, connect, join or attach two or more propositions or sub-propositions in grammatically accurate way in which the meaning and purpose of a compound sentence should not be meaningless, some specific symbols or words are used, and these words or symbols are known as logical connectives or operators. Binary connectives are widely and commonly used logical connectives and known as dyadic connectives. In propositional logic there are some specific symbols which are used to connect different propositions with each other to make compound proposition, and these symbols represent logical connectives. These logical connectives are given below:

1. **NOT** connective which is also called a negation connective, and in nature it’s a unary operator;

2. **AND** connective which is also called a conjunction connective, and in nature it’s a binary connective;

3. **OR** connective which is also called a disjunction connective, and in nature it’s a binary connective;

4. **Implication** connective which contains conditional “if…then” statement, and in nature it’s a binary connective;

5. **Equivalence** connective which contains necessary and sufficient condition “if and only if” (sometimes shortly described as iff), and in nature it’s a binary connective;

6. **NAND** connective which is NOT of all ANDs, and in nature it is a binary connective;

8. **NOR** connective which is NOT of any OR, and in nature it is a binary connective;

9. **XOR** connective which is OR but not both, and in nature it’s a binary connective.

It is important to note that NOT, AND, OR, implication, and equivalence are basic and regularly used logical connectives. The connectives NAND, NOR, and XOR are deriving connectives.

Among all others, only NOT connective is a unary logical connective that means it needs at least one statement or expression to work; all other logical operators are binary in nature which means that they need at least two statements or expressions to work.

The logical connectives with quantifiers have logical constants of two major types which are used as propositional logic and predicate logic in the academic system. The basic connectives which are used in propositional logic are represented in Table 1.

Table 1: Basic logical connectives

Negation	NOT	¬, ~
Conjunction	AND	∧
Disjunction	OR	∨
Implication	if….then	→
Equivalence	if and only if (iff)	↔

Some basic and deriving connectives can be used in examples represented in Table 2.

3.3 Notation history of logical connectives

The representation of True and False values of logical propositions can be done by using the following specific symbols (notations): ⊤ for True and ⊥ for False.

Negation is denoted by ¬ symbol, and it was firstly introduced by Heiting in 1929. The symbol ~ is another form to represent Negation, and it was firstly introduced by Russell in 1908. Before this, Negation was represented by prime (′) and bar (¯).

Conjunction is represented by the symbol ∧ and was firstly introduced by Heiting in 1929. Before this, the symbol ∩ (intersection) was used to represent AND.

Disjunction is represented by the symbol ∨ and was firstly introduced by Russell in 1908, and before this it was represented by the symbol ∪ (union).

Table 2: Some basic and deriving logical connectives in example

Logical connective	**Example** P = It is rainy Q = It is cool breeze
NOT (¬)	It is not rainy ¬P
AND (∧)	It is rainy and cool breeze P ∧ Q
OR (inclusive) (∨)	It is rainy or cool breeze (or both) P ∨ Q
Neither ... NOR	It is neither rainy nor cool breeze. ¬P ∧ ¬Q
OR (Exclusive) (XOR)	It is rainy or cool breeze (but not both). (P ∨ Q) ∧ ¬ (P ∧ Q)

Implication connective is represented by the symbol → and was firstly introduced in 1917 by Hilbert, and before this the symbol ⊃ was used by Russell in 1908, and the symbol => was used by Vax.

Equivalence connective is represented by the symbol ↔, and was denoted by different symbols in different times as follows: as symbol = in 1908 by Russell; as symbol ↔ by Tarksi in 1940; as symbol ⊃⊂ by Gentzen; as symbol ~ by Schonfinkell; and as symbol ⊂⊃ by Chazel.

Some authors also used letters to represent connectives at different times like Hilbert in 1904 used **u** as conjunction, and it is abstracted from German "und" meaning "and", **o** for disjunction derived from German "oder" meaning "or". In 1929, Lukasiewicz used **N** for negation, **K** for conjunction, **A** for disjunction, **C** for implication and **E** for equivalence connectives.

3.4 Truth tables for logical connectives

Truth table is based on binary values with combination of rows and columns and is used to check and verify logical operators' functionality, validity and scope. In broad sense, a truth table is a mathematical table that is used for logics' verification, and is especially concerned about Boolean algebra, Boolean functions and propositional calculus. Specifically, a truth table is used to calculate and check the truth and falsity of a propositional expression. In a truth table the possible binary values are written to predict and calculate the result under the condition that it should properly work for any combination of logical values.

A compound statement is formed from the combination of two or more statements.

The number of rows in a truth table can be determined by a number of variables and can be easily calculated using a simple formula "2^n". If there are two variables then number of rows will be $2^2 = 2\times2=4$, and if the number of variables is 5, then number of rows will be $2^5 = 2\times2\times2\times2\times2=32$. A truth table contains one column for every individual input variable A,B,…..,X,Y,Z, and there must be a column for all the possible results of the logical operations for what the truth table is designed.

To understand the truth tables properly, the following crucial background should be revised:

- A sentence is called a formula or a well-formed formula (WFF) in formal symbolic language of propositional calculus;

- Atomic and compound are two main types of classes in propositional calculus, and atomic formula is used to represent a simple declarative sentence. For example, “Einstein was a scientist” is denoted by a single sentential constant, whereas compound formula is a combination of atomic formulas which is formed by using the logical connectives AND, OR, NOT, XOR etc.;

- The important issue in a truth table is to remember the true or false values of a compound formula;

- A truth table is divided into two sides as right and left, where the left side is used to mention all possible combinations of truth values which have to be used for atomic formulas and propositions, and the right side of a truth table is used to show the resultant truth values of atomic formulas which is considered as a compound formula. So the right part allows determining for which conditions the compound formula is true or false;

- Logical languages are based on perfect clarification and certainty while logical connectives have their relations with natural language and can’t be defined or explained by any means of natural language until an ambiguity in natural language is eliminated.

All logical connectives can be proved by their functionality using truth tables.

3.4.1 Truth table for AND connective

While using AND or conjunction connective, there must be at least two statements to perform this operation. If A and B are statements, then the conjunction A ∧ B is true if and only if both of the statements A and B are true, otherwise it is false. If there are more than two simple statements with possible pairs of true/false values then the conjunction is true if all the statements have true values, otherwise if at least one of the values of the statements is false then conjunction is false. Table 3 shows the truth table for AND connective (with two statements).

Table 3: Truth table for AND connective

A	**B**	**A ∧ B**
T	T	T
T	F	F
F	T	F
F	F	F

3.4.2 Truth table for OR connective

Application of OR or disjunction connective requires at least two simple statements to work. If A and B are statements, then the disjunction A ∨ B is true if at least one of the statements has true value, i.e. if either A or B has true values, or both of the statements have true values, then the disjunction is true, otherwise it is false. If there are more than two simple statements with possible pairs of true/false values then the disjunction is true if at least one of the statements has true value. If all the values of the statements are false, the disjunction is false. Table 4 shows the truth table for OR connective (with two statements).

Table 4: Truth table for OR connective

A	**B**	**A ∨ B**
T	T	T
T	F	T
F	T	T
F	F	F

3.4.3 Truth table for NOT connective

NOT or negation connective is a unary connective; so it needs only one statement to perform operation to check the functionality. This connective gives true result if a value of the statement is false, and if a value of the statement is true then result will be false. Table 5 shows the truth table for NOT connective.

Table 5: Truth table for NOT connective

A	**¬ A**
T	F
F	T

3.4.4 Truth table for Implication connective

Implication connective represented in "if….then" form needs at least two statements to work. While using this connective, the result will be false only if precedent value is true and subsequent value is false; in all other combinations of logical values of statements the result will be true. Table 6 shows the truth table for Implication connective.

3.4.5 Truth table for Equivalence connective

Equivalence connective is implemented if there are at least two statements. This connective results true if precedent and subsequent parts have same status of being true or being false; it means if both values of the statements are true or both values of the statements are false, then Equivalence statement results true, otherwise this connective results false. Table 7 shows the truth table for Equivalence connective.

Table 6: Truth table for Implication connective

A	**B**	**$A \rightarrow B$**
T	T	T
T	F	F
F	T	T
F	F	T

Table 7: Truth table for Equivalence connective

A	**B**	**$A \leftrightarrow B$**
T	T	T
T	F	F
F	T	F
F	F	T

3.4.6 Truth table for NAND connective

The NAND connective is NOT of all AND's and is a binary logical connective of two statements. NAND operator is actually the complement or negation of AND connective, and is read as NOT of A AND B. Table 8 shows the truth table for NAND connective.

Table 8: Truth table for NAND connective

A	**B**	**A NAND B**
T	T	F
T	F	T
F	T	T
F	F	T

3.4.7 Truth table for NOR connective

The NOR connective is NOT of OR and is binary logical connective of two statements. This connective results true if both precedent and subsequent have same false values, and this connective results false for all other combinations of pairs of values of both statements. NOR is also read as NOT of A OR B. Table 9 shows the truth table for NOR connective.

Table 9: Truth table for NOR connective

A	**B**	**A NOR B**
T	T	F
T	F	F
F	T	F
F	F	T

3.4.8 Truth table for XOR connective

The XOR (exclusively OR) connective is a binary logical connective of two statements. This logical connective results true if exactly one of two statements has true value and these connective results false if either both statements have true values or both statements have false values. Table 10 shows the truth table for XOR connective.

Table 10: Truth table for XOR connective

A	**B**	**A XOR B**
T	T	F
T	F	T
F	T	T
F	F	F

3.5 Order of precedence of logical connectives

The sequential hierarchy of nominal priority of values or numbers is known as order of precedence which is normally used when there are multiple (more than one) connectives in a single statement to make it clear which sequence and flow expression will be processed and which connective will take first place and which will come at the end and what will be the order of other connectives used in expression. Connectives' precedence or hierarchy is commonly used for compound expressions.

The connectives' precedence has been very meaningful for a long time, and it was firstly called the "order of operations". The rule for connective hierarchy is simple and given below:

- Parentheses “( )” in expressions have more priority from inside to outside in every condition;

- Negation connective “¬” always has priority over all others;

- Conjunction or logical AND “∧” is processed after negation, if required;

- Disjunction or logical OR “∨” will take place afterwards;

- Implication “→” will take place after disjunction;

- Equivalence “↔” will take place at the end.

The precedence of logical connectives is represented in Table 11.

Table 11: Precedence of logical connectives

Connective	**Symbol**	**Precedence**
NOT	¬	1
AND	∧	2
OR	∨	3
Implication	→	4
Equivalence	↔	5

3.6 Tautology, satisfiability, contradiction, contingency, logical entailment and logical equivalence

3.6.1 Tautology

Tautology is a proposition which is always true for all the values of the propositional variables.

Let’s use truth table to verify that the formula A ∨ ¬ (A ∧ B) is a tautology which is represented in Table 12.

Table 12: Truth table for tautology verification of formula A ∨ ¬ (A ∧ B)

A	**B**	**A ∧ B**	**¬ (A ∧ B)**	**A ∨ ¬ (A ∧ B)**
T	T	T	F	T
T	F	F	T	T
F	T	F	T	T
F	F	F	T	T

It is possible to see from the Table 12 that all logical values under the main operator (last column) are true; it means that the compound proposition A ∨ ¬ (A ∧ B) is a tautology.

3.6.2 Satisfiability

A formula is said to be satisfiable if and only if the negation of the formula is not a tautology, or in other words, the formula is true under at least one interpretation.

Let’s use truth table to verify that the formula A ∧ ¬B is satisfiable (Table 13).

Table 13: Truth table for satisfiability verification of formula A ∧ ¬B

A	**B**	**¬B**	**A ∧ ¬B**
T	T	F	F
T	F	T	T
F	T	F	F
F	F	T	F

It is possible to see from the Table 13 that the second logical value under the main operator (last column) is true, so the negation of the formula A ∧ ¬B is not a tautology, so this formula is satisfiable.

3.6.3 Contradiction

A contradiction is a proposition which is always false for all the values of the propositional variables.

Let’s use truth table to verify that the formula A ∧ ¬B ∧ (A → B) is a contradiction which is represented in Table 14.

Table 14: Truth table for contradiction verification of formula A ∧ ¬B ∧ (A → B)

A	**B**	**¬B**	**A → B**	**A ∧ ¬B ∧ (A → B)**
T	T	F	T	F
T	F	T	F	F
F	T	F	T	F
F	F	T	T	F

It is possible to see from the Table 14 that all the logical values under the main operator (last column) are false; it means that the proposition A ∧ ¬B ∧ (A → B) is a contradiction.

3.6.4 Contingency

Another proposition which plays an important role in a reasoning process is a contingency. A contingency is a proposition which is neither a tautology nor a contradiction.

Let’s use truth table to verify that the formula ¬A → (A ∧ B) is a contingency which is represented in Table 15.

Table 15: Truth table for contingency verification of formula ¬A → (A ∧ B)

A	**B**	**¬A**	**A ∧ B**	**¬A → (A ∧ B)**
T	T	F	T	T
T	F	F	F	T
F	T	T	F	F
F	F	T	F	F

It is possible to see from the Table 15 that the propositional form ¬A → (A ∧ B) is a contingency because there is at least one false value (in fact there are two false values), and there is at least one true value (in fact there are two true values) under the main operator (last column).

3.6.5 Logical entailment

The logical entailment in propositional logic means that if one proposition A entails another proposition B, then for all true values of the proposition A the values of the proposition B must be also true (one should not care about the false values of the proposition A). The relationship "A entails B" in symbolic notation is represented as A |= B.

Table 16 shows that the entailment holds between A and (A ∨ B), but the entailment does not hold between A and (¬A ∨ ¬B).

Table 16: Entailment holding between A and (A ∨ B), and non-entailment holding between A and (¬A ∨ ¬B)

A	**B**	**¬A**	**¬B**	**A ∨ B**	**¬A ∨ ¬B**
T	T	F	F	T	F
T	F	F	T	T	T
F	T	T	F	T	T
F	F	T	T	F	T

3.6.6 Logical equivalence

The propositions A and B are called logically equivalent if the condition "A ↔ B is a tautology" is met. Expressing in another form, A and B are logically equivalent if "A entails B" and "B entails A".

De-Morgan's laws are popular in propositional logic. These laws can be described in the following forms: 1) the negation of the logical connective AND involving two statements A and B is logically equivalent to OR connective involving a negation of each statement A and B: ¬(A ∧ B) ≡¬A ∨ ¬B; 2) the negation of the logical connective OR involving two statements A and B is logically equivalent to AND connective involving the negation of each statement A and B: ¬(A ∨ B) ≡ ¬A ∧ ¬B.

De-Morgan’s laws can be better explained by using truth tables. Tables 17 and 18 describe truth tables for first De-Morgan’s law and second De-Morgan’s law, respectively.

Table 17: Truth table for first De-Morgan’s law

A	**B**	**¬A**	**¬B**	**A ∧ B**	**¬(A ∧ B)**	**¬A ∨ ¬B**
T	T	F	F	T	F	F
T	F	F	T	F	T	T
F	T	T	F	F	T	T
F	F	T	T	F	T	T

Table 18: Truth table for second De-Morgan’s law

A	**B**	**¬A**	**¬B**	**A ∨ B**	**¬(A ∨ B)**	**¬A ∧ ¬B**
T	T	F	F	T	F	F
T	F	F	T	T	F	F
F	T	T	F	T	F	F
F	F	T	T	F	T	T

Let's use truth table to verify the logical equivalence of the formula $\neg A \vee B \equiv A \rightarrow B$ which is represented in Table 19.

Table 19: Truth table for verification of logical equivalence of formula $\neg A \vee B \equiv A \rightarrow B$

A	**B**	**¬A**	**¬A ∨ B**	**A → B**
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

It is possible to see from the Table 19 that the logical values in the last two columns are identical; it means that the compound propositions $\neg A \vee B$ and $A \rightarrow B$ are logically equivalent.

3.7 Some properties of logical connectives

There are some characteristics associated with logical connectives and these characteristics are known as properties of logical connectives. Below some properties of logical connectives are given:

Associativity: Three statements A, B, and C should be involved, and the following associativity properties can be defined: $(A \vee B) \vee C \equiv A \vee (B \vee C)$, and $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$.

Commutativity: For the statements A and B, the following commutative properties can be defined: $A \vee B \equiv B \vee A$, and $A \wedge B \equiv B \wedge A$. It should be noticed that the ordering of operands does not make any sense in getting the final result of the above logical connectives.

Distributivity: This property also needs three statements A, B, and C to be involved, and this property can be described as $A \wedge (B \vee C) \equiv (A \wedge C) \vee (B \wedge C)$.

Identity: This property can be described in the following forms: $A \wedge T \equiv A$, and $A \vee F \equiv A$.

Idempotency: For the statement A, the following is always true: $A \wedge A \equiv A$, and $A \vee A \equiv A$.

Absorption: For two statements A and B, this property can be described in the following forms: $A \vee (A \wedge B) = A$, and $A \wedge (A \vee B) = A$.

Negation: This property can be also described in two forms: $A \wedge (\neg A) \equiv F$, and $A \vee (\neg A) \equiv T$.

Involution: This property is described as ¬(¬A) = A. Another name of this property is double complement, i.e. using double negation of any statement will result in original statement.

3.8 Properties of Implication

Assume that A and B represent statements, then A => B means "A implies B". The first statement is called premise, and the second statement is called conclusion. The implication A => B can be also written in the form A → B.

It should be noticed that the implication A => B should not necessarily produce the implication B => A.

If both statements A and B are negated, i.e. ¬A and ¬B are considered, then the implication should be reversed as ¬B => ¬A.

Suppose that two statements A and B have the arrangement (A ∧ (A → B)) → B, so the reasoning will look like:

- Make sure that if A is true and then check that A → B is fulfilled or not and then check B is true or not;

- If all the conditions are true, then first will be A ∧ (A → B) to imply B; so that (A ∧ (A → B)) → B gives true result ever;

- If A is false then the result will be (F ∧ (F → B)) → B which is true;

- F ∧ B = F such that F → B is true.

Chapter 4

ALGEBRAIC NORMAL FORMS AND TWO FORMS OF VALID INFERENCES

4.1 Algebraic normal forms

In normal form, the multiple statements or propositions can be used together by applying such logical connectives as AND, OR, XOR etc. In algebraic normal form two equivalent formulas can be converted into the same normal form by showing whether these formulas are same or not. Algebraic normal form (ANF) is not same as other normal forms, and can be represented as a simple list of multiple variables. The logical sentences can be converted into more standardized form by using logical equivalences which are very important in computer science applications.

There are three basic types of algebraic normal forms: Negation Normal Form (NNF), Conjunctive Normal Form (CNF), and Disjunctive Normal Form (DNF).

4.1.1 Negation Normal Form (NNF)

In negation normal form negation connective NOT (¬) is used in front of all variables and only applied to variables. Other allowed connectives in this normal form are conjunction (AND) and disjunction (OR), and it means that there is no implication in negation normal form. In negation normal form the connective NOT (¬) should go inside the statement as far as possible.

The rewrite rules are performed for computing the negation normal form, and some of these rules are given below:

$\neg(A \wedge B) \rightarrow \neg A \vee \neg B$

$\neg (A \vee B) \rightarrow \neg A \wedge \neg B$

$\neg \neg A \rightarrow A$

We can apply the negation normal form in the following form:

$\neg (A \vee \neg C) = \neg A \wedge \neg \neg C = \neg A \wedge C;$

$\neg (A \wedge \neg (B \wedge C) = \neg A \wedge \neg \neg (B \wedge C) = \neg A \wedge (B \wedge C)$

4.1.2 Disjunctive Normal Form (DNF)

The disjunctive normal form is a normal form which is a normalization disjunction of conjunctive clauses of logical formula, in other words, it is OR of AND connectives also known as products' summation. If there is disjunction of single or multiple conjunctions of literals and disjunctions is implicated strictly, then the logical formula is considered to be in DNF, and if all the variables of disjunction normal form occur strictly just once in every clause then it is called as Full Disjunctive Normal Form (FDNF). The logical connectives which can be used in disjunctive normal form are AND, OR and NOT. NOT operator can be processed as propositional variable and can be only used as a part of a literal.

The following properties are characteristic to DNF:

- variables used in each term must be connected with AND logical connective;

- OR connective is used to bring the terms together;

- either the original variable or its negation must be used in each term;

- expression should not contain any another operation.

The following formulas are in DNF:

$A \wedge B$;

$(A \wedge B) \vee (\neg A \wedge \neg B)$;

$(A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge C)$.

The following formulas are not in DNF because of different reasons:

$(A \vee \neg B) \vee (\neg A \wedge B)$ (because in the first term OR logical connective is used);

$(A \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C)$ (because the first term does not contain the variable B);

$(\neg A \vee B) \rightarrow (A \wedge \neg B)$ (because implication connective is used).

If the formula is not in DNF because of absence of the variable, it is possible to make the given formula in DNF. Let's consider the following formula which is not in DNF:

$$F(A,B,C) = (A \wedge B \wedge C) \vee (\neg A \wedge B)$$

This formula can be brought to DNF in the following form:

$F(A,B,C) = (A \wedge B \wedge C) \vee (\neg A \wedge B) = (A \wedge B \wedge C) \vee (\neg A \wedge B) \wedge (C \vee \neg C) = (A \wedge B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (\neg A \wedge B \wedge \neg C)$.

4.1.3 Conjunctive Normal Form (CNF)

The conjunctive normal form (CNF) is also known as Clausal Form because of involving a combination of clauses and this form can involve a literal or its complement in each term.

CNF can only involve the propositional connectives AND, OR and NOT, and no another connective; the only connective which is used as a part of a literal is NOT connective. CNF is also known as summations' product.

The following formulas are in CNF:

$A \vee B$;

$(A \vee B) \wedge (\neg A \vee B)$

$(A \vee B \vee \neg C) \wedge (A \vee \neg B \vee C) \wedge (A \vee B \vee \neg C)$.

The following formulas are not in CNF because of different reasons:

$(A \vee B) \vee (\neg A \vee \neg B)$ (because the logical connective between terms is OR, but it must be AND);

$(A \wedge B) \wedge (\neg A \wedge \neg B)$ (because the variables A and B are connected by AND logical connective);

$(A \vee B \vee \neg C) \wedge (A \vee \neg B \vee C) \wedge (A \vee B)$ (because the last term does not contain the variable C).

4.2 Horn Clauses

A logical formula which is in a specific rule-like form and provides good specifications and characteristics to be used in logical programming, formal specification and model theory is known as Horn Clause which is named after the invertor of this clause - the logic expert Alfred Horn who proposed this theory in 1951.

In a Horn clause the disjunction of literals is considered, and there can be at most one positive literal to be used in this disjunction. For example, the clauses $A\vee \neg B$, $\neg A\vee \neg B$, $A\vee \neg B \vee \neg C$, $\neg A\vee \neg B \vee \neg C$ are possible Horn clauses. The clauses $A\vee B$, $A\vee B \vee \neg C$ are not Horn clauses.

A Horn clause is a definite clause if it contains exactly one positive literal. For example, $\neg A \vee B$, $\neg A \vee B \vee \neg C$ are the possible definite clauses.

The definite clause represented in the form $\neg A \vee \neg B \vee C$ can be also described in another form as $A \wedge B \rightarrow C$.

A Horn clause can be described in the form of “condition => conclusion”. For example, the clause represented as

$$\text{parent }(X,Y) \wedge \text{parent }(X,Z) \Rightarrow \text{siblings }(Y,Z)$$

is a Horn clause.

A Horn clause $A \wedge B \wedge C \Rightarrow Q$ can be also described in another Horn clause form $\neg A \vee \neg B \vee \neg C \vee Q$.

It should be also noticed that the logic programming language Prolog which is popular in Artificial Intelligence is based on Horn clauses. Forward chaining which is a data-directed reasoning and backward chaining which is a goal-directed reasoning are based on Horn clauses.

4.3 Two forms of valid inferences

Two forms of valid inferences in propositional logic to be discussed below are Modus Ponens and Modus Tollens.

4.3.1 Modus Ponens

Modus ponens in a propositional logic is a form of valid argument which means that “A implies B, and A is true, so therefore B must be true”. Modus ponens is derived from Latin words MODUS PONENDO PONENS that means the way to make it confirmed, and is abbreviated to MP.

The history of modus ponens is associated with ancient era. Modus ponens is related to another valid argument known as modus tollens, but these arguments have different syntax and semantics.

Modus ponens is a rule of inference, and is also known as a “method of affirming” in propositional logic.

As an example of modus ponens argument form, the following premises and conclusion can be considered:

Premise 1: if student studies hard then he/she will get a good grade.

Premise 2: student will study hard.

Conclusion: student will get good grade.

Modus ponens can be symbolized as

$$\begin{array}{l} A \rightarrow B \\ A \\ \hline B \end{array}$$

or in another form $((A \rightarrow B) \wedge A) \rightarrow B$, where A and B are statements. The truth table for modus ponens is constructed in Table 20.

Table 20: Truth table for modus ponens

A	**B**	**A → B**	**(A → B) ∧ A**	**((A → B) ∧ A) → B**
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

4.3.2 Modus Tollens

Modus Tollens in propositional logic is another form of valid argument and means "A implies B, and not B, so therefore not A". Modus tollens is a deductive form with two premises and one conclusion, and so modus tollens is a syllogism.

Modus tollens is derived from Latin word MODUS TOLLENDO TOLLENS which means the way to refuse the consequent, and is abbreviated to MT. The history of Modus tollens is also related with ancient era.

Modus tollens is also known as a "method of denying" in propositional logic.

As an example of the modus tollens argument form the following premises and conclusion can be considered:

Premise 1: if the traffic is busy, Tom will be late for school.

Premise 2: Tom was not late for school.

Conclusion: the traffic was not busy.

Modus tollens can be symbolized as

$A \rightarrow B$

$\neg B$

———

$\neg A$

or in another form $((A \rightarrow B) \wedge \neg B) \rightarrow \neg A$, where A and B are statements.

The truth table for modus tollens is constructed in Table 21.

Table 21: Truth table for modus tollens

A	**B**	**A→B**	**¬B**	**(A → B) ∧ ¬B**	**¬A**	**((A → B) ∧ ¬B) → ¬A**
T	T	T	F	F	F	T
T	F	F	T	F	F	T
F	T	T	F	F	T	T
F	F	T	T	T	T	T

Chapter 5

CONCLUSION

Propositional logic is a very useful tool for reasoning process, and this logic provides the establishment of a reason to describe the truth or falsehood of the logical statement.

This master thesis represents the propositional logic and its functionality with uses in details and shows how this logic can be used for knowledge representation and reasoning process.

In this thesis such terminologies of a propositional logic as proposition which is a statement that can get one of truth values - true or false, logical connectives which are used to establish compound statements, different types of propositions as tautology, satisfiability, contradiction, contingency and logical equivalence, negation, disjunctive and conjunctive normal forms denoted as NNF, DNF, and CNF respectively, Horn clauses which are logical formulas with at most one positive literal, and two forms of valid inferences - modus ponens and modus tollens, are discussed with their properties.

REFERENCES

[1] Navarro-Perez, J. A., & Voronkov, A. (2013). Planning with Effectively Propositional Logic. *Programming Logics,* pp. 302-316.

[2] Nguyen, T-M-T., Tran, D.-K. (2016). Resolution Method in Linguistic Propositional Logic. *International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 7, No. 1,* pp. 672-678.

[3] Poli R., Ryan M., & Sloman A. (1995). A New Continuous Propositional Logic. *Portuguese Conference on Artificial Intelligence (EPIA 1995), Progress in Artificial Intelligence*, pp 17-28.

[4] Kakas, A., Toni, F., & Mancarella P. (2013). Argumentation for Propositional Logic and Nonmonotonic Reasoning. *In the 11th International Symposium on Logical Formalizations of Commonsense Reasoning, Proceedings.*

[5] Navarro-Perez, J. A., & Voronkov, A. (2008). Proof Systems for Effectively Propositional Logic. *Automated Reasoning. 4th International Joint Conference, IJCAR 2008 Proceedings,* pp. 426-440.

[6] Nizamani, A. R., Strannegard, C. (2014). Learning Propositional Logic From Scratch. *The 28th annual workshop of the Swedish Artificial Intelligence Society (SAIS).*

[7] Buvac, S., & Mason, I. A. (1993). Propositional Logic of Context. *AAAI-93 Proceedings,* pp. 412-419.

[8] Lang, J., & Marquis, P. (1998). Two forms of dependence in propositional logic: controllability and definability. *AAAI-98 Proceedings.*

[9] Bejar, R., & Manya, F. (2000). Solving the Round Robin Problem Using Propositional Logic. *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence,* pp. 262-266.

[10] Nguyen, T.-M.-T., Vu, V.-T., Doan, T.-V., & Tran, D.-K. (2013). Resolution in Linguistic Propositional Logic based on Linear Symmetrical Hedge Algebra. *Knowledge and Systems Engineering. Advances in Intelligent Systems and Computing, vol 244,* pp. 327-338.

[11] Lodder, J., Heeren, B., Jeuring, J. (2016). A Domain Reasoner for Propositional Logic. *Journal of Universal Computer Science, vol. 22, no. 8,* pp. 1097-1122.

[12] Finger, M. (2004). Towards Polynomial Approximations of Full Propositional Logic. *Advances in Artificial Intelligence – SBIA 2004: Proceedings of 17th Brazilian Symposium of Artificial Intelligence.*

[13] Ishida T., Inokuchi, S., & Kawahara Y. (2016). Propositional Logic and Cellular Automata on Monoids. *Journal of Cellular Automata, Vol. 12,* pp. 27-45.

[14] Dyrkolbotn, S., & Walicki, M. (2014). Propositional discourse logic. *Synthese, 191,* pp. 863–899.

[15] Pietarinen, A.-V. (2001). Propositional Logic of Imperfect Information: Foundations and Applications, *Notre Dame Journal of Formal Logic, Volume 42, Number 4,* pp. 193-211.

[16] Lukins, S., Levicki, A., & Stacy, J.B. (2002). A Tutorial Program for Propositional Logic with Human/Computer Interactive Learning. *Proceedings of the 33rd SIGCSE technical symposium on Computer science education,* pp. 381-385.

[17] Zhang, Z., Sui, Y., Cao, C., & Wu, G. (2006). A formal fuzzy reasoning system and reasoning mechanism based on propositional modal logic. *Theoretical Computer Science 368 (2006),* pp. 149-160.

[18] Rene, B., Bedregal, C., & Cruz, A.P. (2006)..Propositional Logic as a Propositional Fuzzy Logic. *Electronic Notes in Theoretical Computer Science, 143,* pp. 5–12.