Automated Education Propositional Logic Tool (AEPLT): Used For Computation in Discrete Mathematics

J. Mbale *mbalej@yahoo.com*
Centre of Excellence in Telecommunications (CoE)
Department of Computer Science
Faculty of Science
University of Namibia
Windhoek, 340, Namibia

Abstract

The Automated Education Propositional Logic Tool (AEPLT) is envisaged. The AEPLT is an automated tool that simplifies and aids in the calculation of the propositional logics of compound propositions of conjuction, disjunction, conditional, and bi-conditional. The AEPLT has an architecture where the user simply enters the propositional variables and the system maps them with the right connectives to form compound proposition or formulas that are calculated to give the desired solutions. The automation of the system gives a guarantee of coming up with correct solutions rather than the human mind going through all the possible theorems, axioms and statements, and due to fatigue one would bound to miss some steps. In addition the AEPL Tool has a user friendly interface that guides the user in executing operations of deriving solutions.

Keywords: Compound proposition, propositional variables, propositional logic, truth table, connective and SEMINT specific parser.

1. INTRODUCTION

This work envisages a solution of automating the calculation of the propositional logic which is user friendly. This paper introduces the Automated Education Propositional Logic Tool (AEPLT) designed for the stated task. The AEPLT automatically calculates the propositional logics of compound propositions of conjuction, disjunction, conditional, and bi-conditional. The AEPLT architecture is composed of the following components: Proposition Process (PP), Operator, SEMINT Specific Parser, Assumption Statements, T/F and Truth Table. Once the user activates the system, the SEMINT Specific Parser automatically extracts the propositional variables from the PP and connectives from Operator. The SEMINT Specific Parser has the ability of forwarding the compound propositions or formulas to the right Assumption Statements. In these right Assumption Statements, the formulas are examined against various statements. After this examination, the system is ready to give results whether Truth False value, which is then recorded in the truth table. Once the results are recorded, the user can access the results from the truth table for the intended application. The work also demonstrates an algorithm that clearly illustrates the stages of calculating and implementing the tool. The system's application is comprehensively demonstrated by its interface, which guides the use of the system and makes this tool user friendly.

1.1 Statement of Problem

During the Discrete Mathematics classes, students struggle to calculate the propositional logics of compound proposition. Yet in this cutting edge era, the Information Communication Technology (ICT) tools have been employed and applied to automate all challenging mathematical, physics, engineering and any other scientific problems. Considering all the theorems, axioms and statements the student has to undergo in order to derive the logic that would help him come up with results, the process tends to be cumbersome. It is in view of this that the AEPLT was introduced to automate the calculation of the propositional logic of the conjuction, disjunction, conditional and bi-conditional.

1.2 Literature Review

The propositional logic is one of the topics under Discrete Mathematics course or discipline. Before tackling propositional logic, it is inevitable to first look at the Discrete Mathematics which is the overall course. In fact, the significance of Discrete Mathematics as the basis for formal approaches to software development has been noted by many scholars such as Dijkstra, Gries, and Schneider [1,2,3]. This position continues to be espoused by the ITiCSE working group [4] among others [5,6,7,8]. The idea that Discrete Mathematics can be viewed as software engineering mathematics has been popularized by the Woodcock and Loomes textbook [9]. In [1] it was emphasized that the application of Discrete mathematics to the software development problem has been the subject of extensive research. He added that much of the initial effort was directed to formal verification, the process of showing the equivalence of two software system presentations. He also indicated that Discrete Mathematics pedagogy has a rich background. Whereas in [10,11,12,13] they pointed out that Discrete Mathematics literature dates back even to the first proposed computing curricula. In [14, 15] they described Discrete Mathematics as the study of mathematical structures and objects that are fundamentally discrete rather than continuous. They gave some examples of objects with discrete values as integers, graphs, or statement in logic. From their description they also pointed out that concepts from Discrete Mathematics were useful for describing objects and problems in computer algorithms and programming languages. They further emphasized that these had applications in cryptography, automated theorem proving and software development. This description is also supported in [16] where they discussed Discrete Mathematics as the study of mathematical structures that do not support or require the notion of continuity. They outlined the topics of Discrete Mathematics to include: logic, sets, numbers theory, combinatorics, graphs, algorithms, probability, information, complexity, computability, etc. Also in [1] it was pointed out Discrete Mathematics underlying modern software engineering theory that included: propositional and first-order predicate logic, reasoning, proof techniques, induction, finite set theory, relations and graphs. the Discrete Mathematics classes, students struggle to calculate the propositional logics of compound proposition.

The research on the propositional logic was envisaged as far back as 1854 by a Mathematician George Boole [17, 18] who established the rules of symbolic logic in his book titled, The Laws of Thought. Since then, a lot of scholars had been researching on the significance of propositions towards building the reasoning capacity of the learners, engineers and other scientists. Many scholars have come up with various definitions to the concepts of propositional logic. Others have separated definitions of propositional and logic. [19] defined logic as the science of reasoning correctly. He further emphasised that this subject has a long history, and narrates that the person generally agreed to have founded formal logic was Aristotle who is method of formal reasoning was called the syllogism. He also pointed out that in nineteen century philosophers and mathematicians like Boole, De Morgan, Frege and others, became interested in modeling the laws of thought. In [14] he classified logic as the branch of philosophy concerned with analyzing the patterns of reasoning by which a conclusion is drawn from a set of premises, without reference to meaning or context. They further emphasized its importance as a formalization of reasoning, a formal language for deducing knowledge from a small number of explicit stated premises, hypotheses, axioms and facts. They also pointed out that logic was a formal framework for representing knowledge. They also defined proposition as the underlying meaning of a simple declarative sentence, which is either true or false. In this definition, the truth or falsehood of a proposition was indicated by assigning it one of the truth values T, for true and F for false. They also cited some examples of propositions as: mammals are warm blooded, the sun orbits the earth, four is a prime number, Joan is taller than John, etc. A practical example was given from a statement: "In 1938 Hitler seized Austria, *(and)* in 1939 he seized former Czechoslovakia *and* in 1941 he attacked the former USSR *while* still having a non-aggression pact with it." This statement was expressed in atomic propositions as: p = in 1939 Hitler seized Austria; q = 1939 he seized former Czechoslovakia; r = 1941 he attacked the former USSR, and s = in 1949 Hitler had a non-aggression pact with the USSR. This was formalized in propositional logic as: $s \wedge q \wedge r \wedge s$.

[20] pointed out that many mathematical statements were constructed by combining one or more propositions. He further stated that new prepositions called compound propositions are formed from the existing propositions using logical connectives or operators. Whereas [19] defined compound statements as the combination of primitive statements by means of logic connectives. [19, 20] stated that letters were used to denote propositional variables or statement variables. They gave the conventional letters used for propositional variables as: p, q, r, s, etc. The authors also defined the truth table which displays the relationships between truth values that are true (denoted T) if it is true proposition and false (denoted F) otherwise. They classified these propositions as: (i) negation of p and denoted $\neg p$. They defined as the truth value of the negation of p was the opposite of the truth value of p, read as "not p"; (ii) the compound proposition conjunction denoted $p \wedge q$, is true when both p and q are true and false otherwise; (iii) the compound proposition disjunction denoted $p \vee q$ is false when both p and q are false and true otherwise; the compound proposition conditional denoted $p \rightarrow q$, is false when p is true and q is false, and is true otherwise; (iv) the compound proposition exclusive or denoted $p \oplus q$, is true when exactly one of p and q is true and is false otherwise; (v) the compound proposition bi-conditional denoted by $p \leftrightarrow q$, is true when p and q have the same truth value, and is false otherwise.

2. THE AELPT SYSTEM ARCHITECTURE

The Propositional Computation architecture given in Figure 1 is an automated model which is used to calculate the values of compound propositions: conjunction, disjunction, conditional and bi-conditional. The architecture is composed of the following components: Proposition Process (PP), Operator, SEMINT Specific Parser, Assumption Statements, T/F and Truth Table.

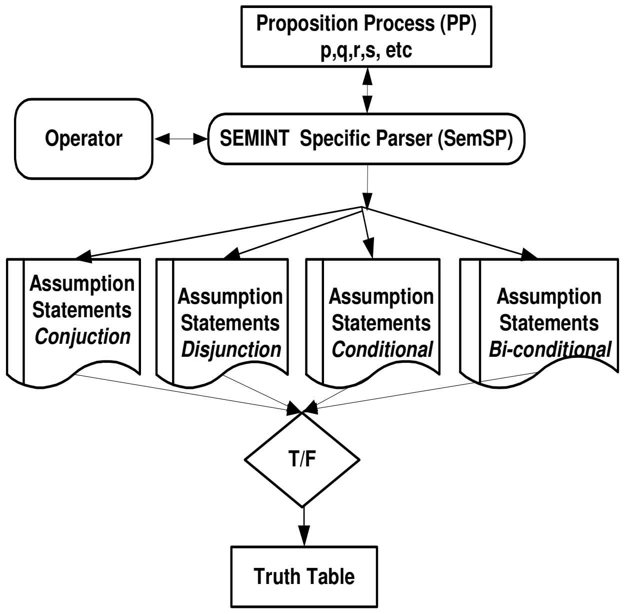

Proposition Process (PP)
p,q,r,s, etc
Operator
SEMINT Specific Parser (SemSP)
Assumption Statements Conjuction
Assumption Statements Disjunction
Assumption Statements Conditional
Assumption Statements Bi-conditional
T/F
Truth Table

FIGURE 1: Preposition computation architecture.

The user activates the Propositional Process (PP) which holds the propositional variables such as the p, q, r, s, etc. Once the PP is activated, the SEMINT Specific Parser (SemSP) automatically extracts the propositional variables and the logical connectives from the Operator, forming a compound statement or formula such as $p \wedge q$, $p \vee q$, $p \rightarrow q$, $p \oplus q$ and $p \leftrightarrow q$. The SemSP is intelligently designed to automatically pass the compound statement to the right Assumption Statements component. The Assumption Statement has pre-programmed statements that determine the final computed propositional statement whether true or false. Then either the true (T) or false (F) is selected from the T/F decision box and this result is recorded in underlying Truth

Table. This Truth Table then holds the results of the computed compound proposition such as conjuction, disjunction, conditional and bi-conditional. Then the user can pick the results his/her application purposes.

3. IMPLEMENTATION OF AUTOMATED EDUCATION PROPOSITIONAL LOGIC TOOL

The implementation of Automated Education Propositional Logic Tool is done by the algorithm illustrated in Figure 2.

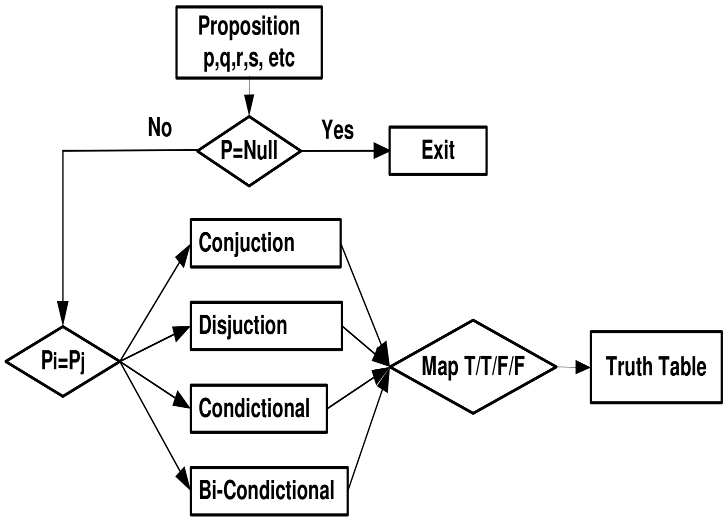


FIGURE 2: Preposition implementation algorithm.

From Figure 2, the propositional variables are activated into the decision box. In the decision box, if the propositional variables are null, then the process is exit. This would imply that the variables are empty and the process can not be continued. At the same instance, if the propositional variables are not null, the pair or set of such statements are passed on to the adjacent decision box. Let P_i be the first propositional variable and P_j be the second one. The pair or the set of propositional variables are mapped to a connective forming a compound statement or formula. Then the mapped propositional variables are forwarded to the right compound proposition where the formula would be computed through a series of statements. These statements make a complete algorithm that determines true and false solution. Below, the statements are discussed:

A. Conjuction ($p \wedge q$):
   - True ***and*** True is True, because both sides of the conjuction are True, then the proposition holds True
   - True ***and*** False is False, because a proposition cannot be both True and False at the same time, hence False
   - False ***and*** True is False, because a proposition cannot be both True and False at the same time, therefore False
   - False ***and*** False is False, because a proposition holds to be False on both sides of conjuction, hence False.

B. Disjunction ($p \vee q$):
   - True ***or*** True is True, because both sides of the disjunction are True, then the proposition holds True
   - True ***or*** False is True, because at least one side of the disjunction is True, therefore, the proposition is True
   - False ***or*** True is True, because at least one side of the disjunction is True, hence, the proposition is True

- False ***or*** False is False, because all the sides of disjunction hold False, then the proposition is False.

C. Conditional (p $\rightarrow$ q):

- True ***implies*** True is True, hence the proposition holds True
- True ***implies*** False is False, this result takes precedence to make the proposition False
- False ***implies*** True is True, this result takes precedence to make the proposition True
- False ***implies*** False is False, which is True from the statement, hence the proposition is True.

D. Bi-Conditional (p $\leftrightarrow$ q):

- True ***implies*** True and is ***implied*** by true, it gives True, hence the proposition holds True
- True ***implies*** False and False ***implies*** True, therefore the proposition is False because what is False is never True and vice versa
- False ***implies*** True and True ***implies*** False, hence the proposition is False because it is not True that what is False is True and vice versa
- False ***implies*** False and False ***implies*** False, hence the proposition is True, because False is False.

After the formulas examine the four compound statements, then from the decision box the results are produced and recorded in the Truth Table. Therefore, the user can pick the results for the intended application.

E. The Application Interface: Propositional Tool:

The AEPLT has a user friendly interface. It has a pull down menu, where the user can select what he/she wants to calculate such as conjuction, disjunction, conditional, and bi-conditional as illustrated in Figure 3.

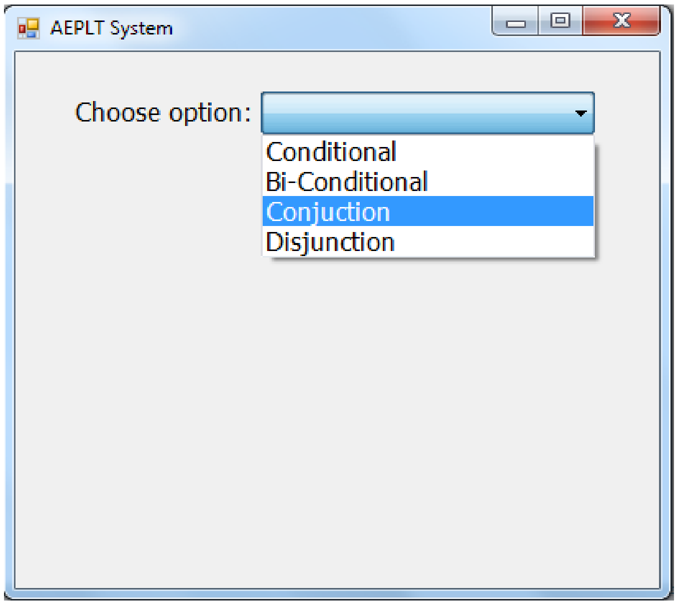
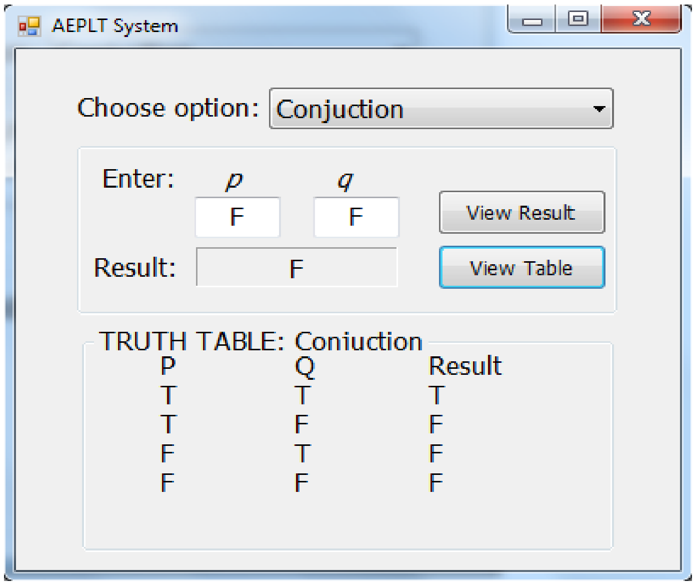


FIGURE 3: Compound propositional input window.

From Figure 3, if you select conjuction from a pull down menu, then the Figure 4 appears. This figure has entry or input spaces for entering the variables of propositions p and q that are True (T) and False (F). Every after entering values True (T) and False (F), one can still view the individual results without checking on the Truth Table by pressing on the button “View Result”. When you press on “View Button”, the system will display Truth Table conjuction results. Similar calculations can be done on others such as the disjunction, conditional and bi-conditional.



Enter: p = F, q = F

Result: F

TRUTH TABLE: Conjuction

P	Q	Result
T	T	T
T	F	F
F	T	F
F	F	F

FIGURE 4: The Conjuction Window for Input.

4. CONCLUSION

Development in all sectors of work, require correct planning in order to provide tools that will yield the intended results. As in [19], logic was defined as the science of reasoning correctly. Once the implementers reason correctly in strategizing and planning in executing their tasks, positive results would be achieved. Hence, in this work, the AEPLT was envisaged to come up with automated systems which will always give a precise propositional logic results. The system has an architecture where the user simply enters the propositional variables and whole calculation is done giving accurate results.

The envisaging of this model, Automated Education Propositional Logic Tool (AEPLT) has scored a number of achievements. First it has allowed the users, who are in this case the students to concretely use this automated model, rather than calculating the propositional logic of compound propositions of conjuction, disjunction, conditional and bi-conditional manually. Secondly, the automated model has a user friendly interface where the student enters the propositional variables and then the system automatically maps them with the right connectives to form compound proposition or formula that are calculated to yield the intended results. Thirdly, during the execution, this automated system gives a guarantee of producing correct results rather than when it is done manually whereby due to fatigue or exhaustion, the user may bound to key-in incorrect input and thereafter result into wrong output.

5. REFERENCES

[1] J. P. Cohoon and J. C. Knight. "Connecting Discrete Mathematics and Software Engineering," 36th ASEE/IEEE Frontiers in Education Conference, San Diego, CA, October 28 – 31, 2006.

[2] E. W. Dijkstra. "On the cruelty of really teaching computing science," Communications of the ACM, December 1989, pp. 1398-1404.

[3] D. Gries, and F. B. Schneider. "A logical approach to discrete math," Springer-Verlag, New York, 1993.

[4] V. L. Almstrum, C. N. Dean, D. Goelman, T. B. Hilburn, and J. Smith. "ITiCSE 2000 working group report: support for teaching formal methods," SIGCSE Bulletin, June 2001.

[5] A. E. Fleury. “Evaluating discrete mathematics exercises”, SIGCSETechnical Symposium on Computer Science Education, 1993, pp. 73-77.

[6] J. W. McGuffee. “The discrete mathematics enhancement project”, Journal of Computing in Small Colleges, 2002, pp. 162-166.

[7] H. Saiedian. “Towards more formalism in software engineering education”, SIGCSE Technical Symposium on Computer Science Education, 1993, pp. 193-197.

[8] K. Heninger. “Specifying Software Requirements Complex Systems: New Techniques and Their Application”, *IEEE* Transactions on Software Engineering, Vol. SE-6, No. 1, January 1980.

[9] J. Woodcock, and M. Loomes. “Software Engineering Mathematics” Software Engineering Institute, Series in Software Engineering, 1988.

[10] A. T. Berztiss. “The why and how of discrete structures”, SIGCSE Technical Symposium on Computer Science Education, 1976, pp. 22-25.

[11] R. E. Prather. “Another look at the discrete structures course”, SIGCSE Technical Symposium on Computer Science Education, 1976, pp. 247-252.

[12] J. P. Tremblay, and R. Manohar. “A first course in discrete structures with applications to computer science,” SIGCSE Technical Symposium on Computer Science Education, 1974, pp. 155-160.

[13] A. Tucker, (editor). “Computing curricula 1991: report of the ACM/IEEE-CS Joint curriculum task force”, *ACM Press*, 1991.

[14] http://www.cs.pitt.edu/

[15] http://gear.kku.ac.pitt.edu/

[16] mason.gmu.edu/~asamsono/teaching/math125/Lecture1.pdf · PDF file

[17] http://docs.google.com/

[18] www-groups.dcs.st-and.ac.uk/history/Mathematicians/Boole.html

[19] Robin Hirsch. www.cs.ecl.ac.uk/staff/r.hirsch//teaching/1b12/

[20] www.coursehero.com/file/2552944/s11propositionallogicBW