

Automated generation and analysis of logical statements

Thesis proposal

For the degree of Master of Science in Computer Science

At Southern Connecticut State University

Anthony Pasqualoni

August 2006

Thesis Advisor: Dr. Hrvoje Podnar

**Major-Field Approval** – The advisor and the department chairperson

---

Advisor

---

Date

---

Chairperson

---

Date

**A. Title**

Automated generation and analysis of logical statements

**B. Statement of Purpose**

The aim of the thesis project will be to implement a data structure and an algorithm to generate and analyze statements in formal logic. The results of the project would be relevant to a wide variety of problems related to formal logic, such as automated theorem proving, and would shed light on the relationship between syntax and semantics in logical statements.

Formal logic is used to construct statements about the natural world and mathematical relationships. One of the most elementary forms of formal logic is propositional logic, where statements are expressed in terms of propositions joined by logical connectives. For example, consider the following statement:

*If we arrive at the station by dusk, we'll catch the train.*

In propositional logic, this can be represented as follows:

$$P \rightarrow Q$$

In the above form, the symbol 'P' represents the proposition 'we arrive at the station by dusk', 'Q' stands for the proposition, 'we'll catch the train', and ' $\rightarrow$ ' represents the 'if...then' connective.

Thus, if 'P' is true, it follows that 'Q' is true. The symbol '->' is known as a 'binary connective', since it connects two propositions to form a new proposition. Other binary connectives include 'and' (represented by the symbol '&') and 'or' (represented by 'v'). The following example uses the three logical connectives mentioned so far:

$$(P \ \& \ Q) \ -> \ (R \ \vee \ S)$$

This expression can be read as: 'if P and Q, then R or S.'

Formal logic has a wide range of applications. For example, electronic components that function as logical connectives are used to construct circuits. As a result, electronic circuits can be expressed as statements in propositional logic. Statements that are logically equivalent yield circuits that have the same function (Enderton, 2001). Thus, designing new circuits is aided by the fact that representative statements in propositional logic can be used to determine a circuit's function.

It turns out that in addition to circuits, propositional logic and its mathematical formulation, Boolean algebra, can be applied to solve problems in a wide range of physical processes, including fluid control systems and mechanical devices (Gardner, 1982).

Logical connectives and the ways in which they are grouped express the syntax of logical statements. In addition to syntax, logical statements can be understood in terms of semantics. Semantics is the relationship between the formal structure of a logical statement and the possible truth values that a statement can express. For example, the statement 'P & ~P', which in English stands for 'P and not P', is always false, since a proposition cannot be both true and false

simultaneously. In formal logic, statements that are always false are called *contradictions*; statements that are always true are *tautologies*. Since they share the same truth values, tautologies and contradictions represent distinct logical classes known as *equivalence classes*.

Formal logic is used to form valid arguments and proofs of theorems in mathematics, philosophy, and a wide range of other subjects. One of the benefits of propositional logic in particular is that there is a simple procedure for demonstrating that a theorem is valid using a truth table. If the table shows that a conclusion of a theorem is true whenever its premises are true, the theorem is valid (Sakharov & Weisstein, 2005). Not all theorems can be expressed in propositional logic, but those that can are amenable to automated theorem proving.

Truth tables can be constructed using an algorithm. However, as the number of basic propositions increases, the size of the table increases exponentially (Enderton, 2001). For example, a truth table for a statement with only two propositions, P and Q, has  $2^2 = 4$  rows in its truth table. A statement with 10 propositions has  $2^{10} = 1024$  rows. Given exponential growth of truth tables, there is a need to develop analysis of logical statements with more efficient methods. One of the more important problems in computer science is to develop a method for determining if a statement is true for at least one possible set of truth values. This is known as the *satisfiability problem*.

Tautologies in particular play an important role in logic, mathematics, and engineering. For example, an important problem in mathematical logic is to decide if tautologies can be deduced using formal proofs in fewer steps than it takes using a truth table (Krajíček, 2001). In proof complexity research, tautologies produced with random number generators are used to evaluate the efficacy of proof systems (Krajíček, 2001). In engineering, algorithms that recognize tautologies are employed for hardware verification (Aagaard, Jones, & Seger, 1998).

Propositional logic can be extended and used as a basis for more powerful logical systems. For example, additional functionality can be added that allows quantities to be expressed. Modal logic is an extended version of propositional logic which includes logical connectives for expressing possibility and necessity. Modal logic is applied in a wide range of subjects, including philosophy, set theory, and computer science.

The proposed research project will aim to shed light on the relationship between syntax and semantics in formal logic by implementing a data structure that can be used to represent logical statements. The data structure will be amenable to an algorithm which would generate and classify logical statements according to equivalence classes. In order to generate statements of sufficient variation and complexity for analysis, the algorithm will be designed and adapted to the data structure.

### ***C. Literature Review***

There are many ways to represent logical statements and systems of logic as an abstract structure. Gawron (2004) shows how a logical system can be represented as a lattice defined by an inference rule. Logical connectives such as 'and' and 'or' are operations on objects in the lattice. Similarly, Cleave (1991) states that some logical systems can best be described by the theory of lattices. He states that logical systems "can be constructed ... from various classes of lattices, with the lattice operations becoming the connectives" (Cleave, 1991, p. 136).

The philosopher and logician C.S. Peirce invented a logical system based on existential graphs (Hartshorne, 1960). Existential graphs are a form of diagrammatic reasoning in which logical relationships are represented as diagrams. Venn diagrams are a simpler form of diagrammatic reasoning. Peirce's existential graphs are much more powerful, and comprise three

logical systems: Alpha, which is equivalent to propositional logic, Beta, equivalent to first-order logic, and Gamma, which is a form of modal logic. Hammer (1996) proved that Alpha is both sound and complete - i.e., only valid theorems can be proved, and a proof exists for all valid theorems. Roberts (1973) and Shin (2002) both provide reading methods for the Alpha and Beta systems. A reading method is a procedure for interpreting a logical graph into a logical statement. Roberts' reading method is more intuitive, but lacks rigor and consistency (Shin, 2002). As a result it is less amenable to implementation as an algorithm. However, Shin (2002) does provide an algorithm for interpreting Peirce's Alpha and Beta graphs, and her method can be implemented in a programming language.

John Sowa uses Peirce's Beta graphs and his theory of semiotics as a basis for his own logical system known as Conceptual Graphs (Sowa, 1983). Sowa's Conceptual Graphs are used in knowledge representation systems to represent logical relationships between concepts and classes of objects. Conceptual Graphs have been the subject of a wide range of studies; a text edited by Ganter and Mineau (2000) provides a collection of several of these studies.

Peirce also showed that the properties of logical connectives such as 'and', 'or', and 'if...then' can be represented by a square divided into four quadrants, with each quadrant representing the four possible truth values of the two propositions that are joined by the connective (Hartshorne & Weiss, 1960). Peirce showed that there are sixteen possible binary connectives, and each of these can be represented by an icon that indicates which quadrants of the square are asserted. Clark (1997) showed that there is an algorithm for constructing tautologies using the connectives, and stated that the sixteen logical connectives form an algebraic group. The table shown below, in figure 1, was used to construct tautologies using icons designed by Peirce for the

connectives. It shows that there are properties shared by the syntactic structures of tautologies, such as symmetry:

TABLE IV

Figure 1: Table for constructing tautologies using Peirce's connectives

Mathematical graphs can also be used to express logical relationships. Enderton (2001) shows how graphs can be used to model statements in first-order logic. Enderton (2001) and Quine (1982) show that a logical statement can be expressed as a tree, a type of graph in which cycles are not present.

Since the 1970's, there has been a rather strict separation between research in machine-based reasoning, which employs logic, and research in machine-based perception, which employs pattern recognition (Schomaker, 2002). Nonetheless, research has shown that pattern recognition and symbolic processing have been successfully combined. For example, pattern recognition systems have been designed that are able to account for logical inference and evaluate validity of

arguments (Bechtel and Abrahamsen, 2002). Logical connectives and propositional variables can be encoded so that a neural network can process an argument and determine whether it is valid.

Hao Wang, while a researcher at Bell Laboratories, published an article on proving theorems using pattern recognition (Wang, 1960). The article states that the method and ideas of pattern recognition can be extended to give a "quasi-decision procedure" in first-order logic (Wang, 1960, p. 221). He explains that this procedure would in theory give a proof if a given formula is a theorem.

Breunig (1995) demonstrates how a pattern recognition program can be designed by using genetic programming. He uses a simplified data set comprising a set of true-false values and demonstrates that for data sets of a limited size, simple programs can be evolved that detect patterns in a two-dimensional grid. A text by Bhanu (2005) includes material on how evolutionary computational techniques, such as genetic programming and genetic algorithms, can be used to implement feature extraction, a crucial element in pattern recognition systems.

Orovos (1999) describes a variety of data structures that can be used in conjunction with pattern recognition, including graphs, trees, webs, and plex structures. However, although these structures and associated algorithms have been applied successfully, they present a number of drawbacks, including the computational complexity of the algorithms, sensitivity to errors in the patterns, and lack of generality. Tombre (1996) reviews the limitations and potential of two basic variations of pattern recognition, syntactical and structural. He states that there are inherent limitations to syntactical and structural methods that can probably be overcome through applied research.

The data structures and pattern recognition techniques described above are only a sample of a much larger class. The limitations and possibilities of various methods for representing and classifying logical statements is the subject of continued research in computer science and related disciplines.

#### ***D. Methodology***

The main phases of the research project will be as follows:

- Design and implement a data structure to represent logical statements.
- Design and implement an algorithm for generating tautologies and contradictions.
- Perform analysis on the generated statements.

The project will focus on propositional logic. As explained above, statements and theorems in propositional logic are amenable to an effective procedure -- i.e., an algorithm can be designed to determine the truth value of statements and the validity of theorems.

There are a variety of ways in which logical statements can be represented by a data structure. Initially, statements will be represented by a graph or lattice, both of which will be evaluated for ease of use, efficacy, and suitability for analysis. Shown below in figure 2 is a graph that represents the statement ' $\sim(P \rightarrow (Q \rightarrow P))$ ' as a tree, with nodes serving as logical connectives and statement variables:

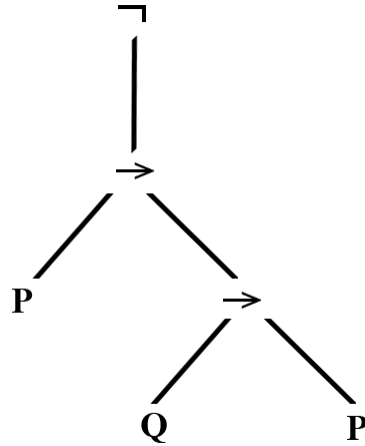


Figure 2: Graph of the logical statement  $\sim (P \rightarrow (Q \rightarrow P))$

Once a data structure is specified and implemented, logical statements will be generated using a randomized algorithm. Initially, statements will be generated recursively using logical connectives and propositional variables. For example, the statement  $(P \vee Q) \rightarrow (P \& \sim R)$  can be generated by randomly selecting the  $\rightarrow$  connective, and expanding the connectives and variables on both sides of the connective using a recursive function.

After a sufficient number of statements are generated, each statement will be evaluated in terms of its truth value. More specifically, the values 'true' and 'false' will be substituted for statement variables, and the statement's truth value will be evaluated. If a statement is true for all truth values, it will be classified as a tautology. If it is false for all truth values, it will be classified as a contradiction. Otherwise, it will be classified as a contingent statement.

In order to generate tautologies and contradictions of sufficient complexity, the algorithm for generating statements will be modified. One method by which the algorithm can be improved is through the use of an evolutionary strategy. For example, genetic programming can be used to evolve random statements into tautologies. Pattern generation algorithms will also be considered

for constructing statements. Once a method is implemented, it will be evaluated in terms of the variation of statements it generates and the speed in which it operates.

When a sufficient number of statements have been generated, the statements will be analyzed in terms of their syntactical structure and their possible truth values. Possible methods for analysis include statistical methods such as Bayesian classifiers and structural pattern recognition algorithms. Information-theoretic measurements such as entropy are also applicable.

The proposed research project will be carried out using conventional software and hardware. ANSI C will be the programming language of choice. Open-source software such as a neural network library may be employed for analysis of the generated logical statements.

### ***E. Contributions of the Project***

The project is expected to improve the understanding of the relationship between syntax and semantics - i.e. logical structure and truth value. Its results will be relevant to areas of research in which statements in propositional logic are generated and analyzed, such as hardware verification and proof complexity.

Finding an appropriate data structure that both expresses logical relationships and is amendable to analysis would help answer a number of difficult problems in formal logic, such as the satisfiability problem and problems in automated theorem proving.

### ***F. References***

Aagaard, M.D., Jones, R.B., & Seger, C.H. (1998). *Combining Theorem Proving and Trajectory Evaluation in an Industrial Environment*. Retrieved August 17, 2006 from [http://www.engineering.usu.edu/ece/faculty/bunker/rlist/32\\_4.pdf](http://www.engineering.usu.edu/ece/faculty/bunker/rlist/32_4.pdf).

- Bechtel, W. & Abrahamsen, A. (2002). *Connectionism and the Mind*. Oxford: Blackwell Publishers.
- Bhanu, B. (2005). *Evolutionary Synthesis of Pattern Recognition Systems*. New York: Springer.
- Breunig, M.M. (1995). *Location Independent Pattern Recognition using Genetic Programming*. Retrieved August 14, 2006 from <http://citeseer.ist.psu.edu/breunig95location.html>.
- Clark, Glenn. (1997). New Light on Peirce's Iconic Notation for the Sixteen Binary Connectives. In N. Houser, D. Roberts, & J.V. Evra (Eds.), *Studies in the Logic of Charles Sanders Peirce* (pp. 304-333). Bloomington: Indiana University Press.
- Cleave, J.P. (1991). *A Study of Logics*. Oxford: Clarendon Press.
- Enderton, H.B. (2001). *A Mathematical Introduction to Logic*. New York: Harcourt.
- Ganter, B. & Mineau, G.W. (Eds.). (2000). *Conceptual Structures: Logical, Linguistic, and Computational Issues*. New York: Springer.
- Gardner, M. (1982). *Logic Machines and Diagrams* (2nd ed.). Chicago: The University of Chicago Press.
- Gawron, J.M. (2004). *Partially ordered sets and lattices*. Retrieved August 13, 2006 from <http://www-rohan.sdsu.edu/~gawron/mathling/poset/posets.pdf>.
- Hammer, E. (1996). Peircean Graphs for Propositional Logic. In G. Allwein & J. Barwise (Eds.), *Logical reasoning with diagrams* (pp. 129-148). New York: Oxford University Press.
- Hartshorne, C. & Weiss, P. (Eds.). (1960). *Collected Papers of Charles Sanders Peirce* (Vol. 4). Cambridge, Massachusetts: Harvard University Press.
- Krajčec, Jan. (2001). *Tautologies From Pseudo-Random Generators*. Retrieved August 17, 2006 from <http://citeseer.ist.psu.edu/281596.html>.

- Orovas, C. (1999). *Cellular Associative Neural Networks for Pattern Recognition*. Retrieved August 14, 2006 from <http://www.cs.york.ac.uk/ftplib/reports/YCST-99-12.pdf>.
- Quine, W.V. (1982). *Methods of Logic* (4th ed.). Cambridge, Massachusetts: Harvard University Press.
- Roberts, D.D. (1973). *The Existential Graphs of Charles S. Peirce*. Paris: Mouton.
- Sakharov, A. & Weisstein, E. (2005). *Propositional Calculus*. Retrieved August 13, 2006 from <http://mathworld.wolfram.com/PropositionalCalculus.html>.
- Schomaker, L. (2002). *Patterns and Symbols: A World Through the Eye of the Machine*. Retrieved August 14, 2006 from <http://www.ai.rug.nl/~lambert/papers/oratie-schomaker-2002-eng.pdf>.
- Shin, S. (2002). *The Iconic Logic of Peirce's Graphs*. Cambridge, Massachusetts: The MIT Press.
- Sowa, J.F. (1983). *Conceptual Structures: Information Processing in Mind and Machine*. Boston: Addison-Wesley.
- Tombre, K. (1996). *Structural and Syntactic Methods in Line Drawing Analysis: To which Extent do they Work?*. Retrieved August 14, 2006 from <http://citeseer.ist.psu.edu/tombre96structural.html>.
- Wang, H. (1960). Proving theorems by pattern recognition I. *Communications of the ACM*, 3 (4), 220-234.