Principia Mathematica and the Development of Automated Theorem Proving

Daniel J. O'Leary

1. Introduction

The present chapter describes the first two major published works in automated theorem proving, the *Logic Theory Machine* (LT) of Newell, Shaw, and Simon about 1956 and the work of Wang in 1958. Both works attempted to prove theorems in propositional logic found in *Principia Mathematica*. While the LT is the first published example of automated theorem proving, Martin Davis, in 1954, wrote a program to prove theorems in additive arithmetic (Loveland, 1984, p. 1). His results were not published.

Part I, Section A of *Principia Mathematica* (PM) (Whitehead and Russell, 1935) is entitled "The Theory of Deduction" and covers 35 pages of the sixteen-hundred page work. In this section, Whitehead and Russell state and prove about two hundred theorems in propositional logic. The LT attempted to prove the first 52 theorems and succeeded with 38. Wang's program proved them all. The two projects had the same goal, but quite different approaches. Even the concept of what constitutes a proof is different.

To help understand the differences in these approaches we adopt some definitions from Loveland (1984). By *automated theorem proving* we mean "the use of a computer to prove nonnumerical results ... often [requiring] human readable proofs rather than a simple statement 'proved'." Loveland identifies two major approaches to automated theorem proving which he calls the *logic approach* and the *human simulation* or *human-orientated* approach. "The logic approach is characterized by the presence of a dominant logical system that is carefully delineated and essentially static over the development stage of the theorem proving system."

The human simulation programs attempt to find proofs in the way a mathematician would. They usually have a body of facts, a knowledge base to which they refer, an executive director which controls the decision-making process, and the ability to chain both forward and backward. Newell, Shaw, and Simon's LT is a human simulation theorem prover; it attempts to find a proof a human would find using all the theorems proved previously. Wang's work follows the logic approach, and tends to give a result more akin to the word "proved" with reasons why the assertion can be made.

2. The Logic Theory Machine (LT)

Herbert Simon and Allen Newell of Carnegie-Mellon University and J.C. Shaw of the Rand Corporation wrote the LT programs in 1956 and 1957. They wrote this human simulation program to understand "how it is possible to solve difficult problems such as proving mathematical theorems, discovering scientific laws from data, playing chess, or understanding the meaning of English prose." (Newell et al., 1956, p. 109) They state:

> The research reported ... is aimed at understanding the complex processes (heuristics) that are effective in problem solving. Hence, we are not interested in methods that guarantee solutions, but which require vast amounts of computation. Rather, we wish to understand how a mathematician, for example, is able to prove a theorem even though he does not know when he starts how, or if, he is going to succeed.

2.1. Inside the Logic Theory Machine

The LT has three methods available to prove theorems: substitution, detachment, and chaining. The user gives the LT a theorem to prove and all theorems in PM up to that point. The LT may not have proved any of them. The theorems which precede the one to be proved are placed on a theorem list. The program refers to this theorem list as it attempts to apply various proof techniques.

The substitution method attempts to transform an entry on the theorem list to the desired theorem. It may substitute variables or replace connectives based upon definitions. The detachment method makes use of *modus ponens*. Assume we wish to prove B. The LT searches the theorem list for an entry of the form A implies B. If one is found A becomes a subproblem which the LT adds to a subproblem list. Chaining falls into two types, forward and backward. Both methods rely on the transitivity of implication. Assume the LT wishes to prove A implies C. In forward chaining the LT searches the theorem list for an entry of the form A implies B and if found, enters B implies C on the subproblem list. In backward chaining the LT searches the theorem list for B implies C and enters A implies B on the subproblem list.

The LT contains an executive director which controls the use of the three methods. The executive director uses the following scheme:

First use substitution.

Should substitution fail, try detachment. The result of each detachment is submitted to the substitution method. If substitution fails, then the result of detachment is placed on the subproblem list.

Try forward chaining next and submit the result to substitution. If a proof has not been found, the result of forward chaining is added to the subproblem list.

Try backward chaining in a manner similar to forward chaining.

When all the methods have been tried on the original problem, the executive director goes to the subproblem list and chooses the next subproblem. The techniques are applied in the same sequence, and if no proof is found the next problem on the subproblem list is tried.

The search for a proof stops when one of four conditions have been met:

1. the LT has found a proof;
2. the bounds set on time for the search have been exceeded;
3. the bounds set on memory for the search have been exceeded; or
4. no untried problems remain on the subproblem list.

2.2. An Example of a Logic Theory Proof

One of the theorems LT proved is

$$*2.45 \quad \vdash: \sim(p \vee q). \supset . \sim p.$$

It is instructive to compare the proofs of LT, PM, and Wang's program. In PM the proofs are often abbreviated. The authors (Whitehead and Russell, 1935, p. vi), state "The proofs of the earliest propositions are given without omission of any step, but as the work proceeds the proofs are gradually compressed, retaining however sufficient detail to enable the reader by the help of the references to reconstruct proofs in which no step is omitted."

The proofs of *2.45 in PM is presented as:

$$*2.45 \quad \vdash: \sim(p \vee q). \supset . \sim p \qquad [2.2, \text{Transp}].$$

Transp refers to one of seven theorems which carry that name. The proof given by LT and the reconstruction from PM are identical. (Wang's proof is given below.) The reconstruction of the proof is:

*2.45	$\sim(p \vee q). \supset . \sim p$	
1.	$p. \supset . p \vee q$	*2.2
2.	$p \supset q. \supset . \sim q \supset \sim p$ $q/p \vee q$	*2.16 [Transp]
3.	$p \supset . p \vee q: \supset : \sim(p \vee q). \supset . \sim p$	Sub. in 2
4.	$\sim(p \vee q). \supset . \sim p$	*Modus ponens* 3, 1

By examining the proof, one may trace the successful path taken by LT. The theorem list contains all theorems up to *2.45. Since the consequent of *2.16 has the same structure as the theorem to be proved, a substitution will make them identical (see lines 2 and 3 in the proof). The LT then tries to find a match for the antecedant and locates *2.2 on the theorem list.

3. Wang's Techniques

Newell, Shaw, and Simon contrasted algorithmic and heuristic methods in searching for a proof. The algorithmic methods guarantee that if a problem

has a solution it will be found. No claim, however, is made on how long it will take or the resources required. In the heuristic method we give up the guarantee, but solutions may be found at reduced cost.

To illustrate the algorithmic method, Newell, Shaw, and Simon put forward the British Museum Algorithm—look everywhere. All axioms are proofs of length 1. All proofs of length n are formed by making all the permissible substitutions and replacements in proofs of length $(n-1)$ and all permissible detachments of pairs of theorems generated up to this point. This method will, of course, generate every theorem of PM, but at a high cost.

Wang thought the argument against the algorithmic method, taking the British Museum Algorithm as an example, was weak. He wanted to show that there were other algorithmic methods available and the British Museum Algorithm only a straw man. Wang sought a method that guaranteed a proof and did not require extensive running time.

3.1. The Wang Program

Wang's algorithm, based on sequent logic, is much better than the British Museum Algorithm. Five connectives are used in PM: negation, implication, disjunction, conjunction, and equivalence. Wang's system uses eleven rules of inference. One rule identifies a theorem, five introduce a connective on the left-hand side of the sequent arrow, and five introduce a connective on the right-hand. These rules are stated below:

P1. Initial rule: If λ, ζ are strings of atomic formulas, then $\lambda \to \zeta$ is a theorem if some atomic formula occurs on both sides of the arrow.

In the following ten rules, λ and ζ are always strings (possibly empty) of atomic formulas.

P2a. Rule $\to\sim$: If $\phi, \zeta \to \lambda, \rho$, then $\zeta \to \lambda, \sim\phi, \rho$.
P2b. Rule $\sim\to$: If $\lambda, \rho \to \pi, \phi$, then $\lambda, \sim\phi, \rho \to \pi$.
P3a. Rule $\to\&$: If $\zeta \to \lambda, \phi, r$ and $\zeta \to \lambda, \psi, \rho$, then $\zeta \to \lambda, \phi \,\&\, \psi, \rho$.
P3b. Rule $\&\to$: If $\lambda, \phi, \psi, \rho \to \pi$, then $\lambda, \phi \,\&\, \psi, \rho \to \pi$.
P4a. Rule $\to\vee$: If $\zeta \to \lambda, \phi, \psi, \rho$, then $\zeta \to \lambda, \phi \vee \psi, \rho$.
P4b. Rule $\vee\to$: If $\lambda, \phi, \rho \to \pi$, then $\lambda, \phi \vee \psi, \rho \to \pi$.
P5a. Rule $\to\supset$: If $\zeta, \phi \to \lambda, \phi, \rho$, then $\lambda \to \phi \supset \psi, \rho$.
P5b. Rule $\supset\to$: If $\lambda, \psi, \rho \to \pi$ and $\lambda, \rho \to \pi, \phi$, then $\lambda, \phi \supset \psi, \rho \to \pi$.
P6a. Rule $\to\equiv$: If $\phi, \zeta \to \lambda, \psi, \rho$ and $\psi, \zeta \to \lambda, \phi, \rho$, then $\zeta \to \lambda, \phi \equiv \psi, \rho$.
P6b. Rule $\equiv\to$: If $\phi, \psi, \lambda, \rho \to \pi$ and $\lambda, \phi \to \pi, \phi, \psi$, then $\lambda, \phi \equiv \psi, \rho \to \pi$.

In sequent logic a theorem is proved by taking a finite set of strings in the form specified by rule P1 and applying the other rules until the theorem is proved. Wang's program turns this process around. The applications of the inference rules were deduced by starting at the theorem until a finite number of occurrences of rule P1 are reached. Given a theorem in PM to prove, one places a sequent arrow in front of it and proceeds. If, however, the main

connective is an implication, Wang often replaces it with the sequent arrow and then starts.

3.2. Examples Of Proofs

The proof of *2.45 generated by this method is shown below:

$$*2.45 \quad \tilde{}(p \vee q) \rightarrow \tilde{}p \tag{1}$$

$$(1) \quad \rightarrow \tilde{}p, p \vee q \tag{2}$$

$$(2) \quad p \rightarrow p \vee q \tag{3}$$

$$(3) \quad p \rightarrow p, q \tag{4}$$

The numbers on the right-hand side are line numbers. The numbers on the left-hand side show which line was used to produce the line of interest. The connectives are dealt with in order, from left to right. Negation, the left-most connective of line (1) can be introduced on the left-hand side of a sequent arrow by rule P2b, so line (1) could be generated by something of the form in line (2). The lines are examined until the conditions of rule P1 are satisfied.

Another example is given by

$$*5.21 \quad \vdash\colon \tilde{}p \,\&\, \tilde{}q \,.\, \supset \,.\, p \equiv q$$

a proof of which is shown below.

$$*5.21 \quad \rightarrow \tilde{}p \,\&\, \tilde{}q \,.\, \supset \,.\, p \equiv q \tag{1}$$

$$(1) \quad \tilde{}p \,\&\, \tilde{}q \rightarrow p \equiv q \tag{2}$$

$$(2) \quad \tilde{}p, \tilde{}q \rightarrow p \equiv q \tag{3}$$

$$(3) \quad \tilde{}q \rightarrow p \equiv q, p \tag{4}$$

$$(4) \quad \rightarrow p \equiv q, p, q \tag{5}$$

$$(5) \quad p \rightarrow q, p, q \tag{6}$$

$$(5) \quad q \rightarrow p, p, q \tag{7}$$

The connectives are dealt with from left to right, across the proof line as before. Finally, at line (5) there is an equivalence on the right-hand side of the arrow. By Rule P6b this must have come from two formulas. One is found in line (6) and the other in line (7).

The proof is always in the form of a finite tree, and the program conducts a depth-first search through the tree. A branch is explored until all connectives are eliminated. The program then checks that rule P1 is satisfied. The program stops when all branches have been explored. The formula submitted is a theorem if rule P1 is satisfied at the bottom of each branch of the search tree. The program does not produce proofs in the sense of PM, since it does not refer to previously proved theorems, nor is there a progression of theorems from a set of axioms.

4. Sidelights of the Logic Theory Machine

Simon (1956) wrote to Russell in late 1956 and described the work of the LT. Russell (1956) replied, "I am delighted to know that *Principia Mathematica* can now be done by machinery. I wish Whitehead and I had known of this possibility before we wasted ten years doing it by hand. I am quite willing to believe that everything in deductive logic can be done by machinery." The LT was named as a co-author on a paper submitted to the *Journal of Symbolic Logic*, but the paper was refused (Tiernay, 1983).

The proof of *2.85 given in PM, shown below, contains an error. The LT was able to find a remarkable proof of *2.85.

*2.85 $\vdash :. p \vee q . \supset . p \vee r : \supset : p . \vee . q \supset r$

Dem.

[Add . Syll] $\vdash :. p \vee q . \supset . r : \supset . q \supset r$ (1)

$\vdash .$ *2.55. $\supset \vdash : : \sim p . \supset :. p \supset r . \supset . r :.$

[Syll] $\supset :. p \vee q . \supset . p \vee r : \supset : p \vee q . \supset . r :.$

[(1) . *2.83] $\supset :. p \vee q . \supset . p \vee r : \supset : q \supset r$ (2)

$\vdash .$ (2) . Comm . $\supset \vdash :. \quad p \vee q . \supset . p \vee r : \supset : \sim p . \supset . q \supset r :$

[*2.45] $\supset : p . \vee . q \supset r :. \supset \vdash .$ Prop

The line numbered (2) cannot be obtained as indicated. The intent of Whitehead and Russell here is not clear, as explained in an unpublished paper from the Russell Conference of 1984. Simon (1956) wrote to Russell stating that the LT had "created a beautifully simple proof to replace a far more complex one in [*Principia Mathematica*]." The proof sent to Russell, however, contains a simple error. The correct proof is shown below:

*2.85 $p \vee q . \supset . p \vee r : \supset : p . \vee . q \supset r$	
1. $p \supset q . \supset : q \supset r . \supset . p \supset r$	*2.06
$p/q, q/p \vee q, r/p \vee r$	
2. $q \supset p \vee q . \supset : p \vee q \supset p \vee r . \supset . q \supset p \vee r$	Sub. 1
3. $q \supset p \vee q$	*1.3
4. $p \vee q \supset p \vee r . \supset . q \supset p \vee r$	*Modus ponens* 3, 2
5. $p \vee q \supset p \vee r . \supset . \sim q \vee (p \vee r)$	Def. of $\supset$
6. $p \vee (q \vee r) . \supset . q \vee (p \vee r)$	*1.5
$p/\sim q, q/p$	
7. $\sim q \vee (p \vee r) . \supset . p \vee (\sim q \vee r)$	Sub. 6
8. $p \vee q \supset p \vee r . \supset . p \vee (\sim q \vee r)$	Chain with 5 and 7
9. $p \vee q \supset p \vee r . \supset . p . \vee . q \supset r$	Def of $\supset$

In his letter Simon states, "The machine's proof is both straightforward and unobvious, we were much struck by its virtuosity in this instance."

5. Conclusion

Principia Mathematica is clearly a human creation. Its formalism, however, almost begs for a mechanical procedure. There is no surprise that, when looking for some human endeavor to simulate with a computer, PM is chosen. Both methods described here have had a profound effect on automated theorem proving. The work of Newell, Shaw, and Simon laid a strong foundation for the modern techniques of artificial intelligence. The work of Wang immediately opened up a new front in the battle to understand what computers can do.

Bibliography

Loveland, D.W. (1984), Automated theorem proving: a quarter-century review, in *Automated Theorem Proving* (Bledsoe and Loveland, eds.), American Mathematical Society, Providence, RI, pp. 1–45.

Newell, A., Shaw, J.C., and Simon, H.A. (1956), Empirical explorations of the logic theory machine: A case study in heuristics. *Proceedings of the Western Joint Computer Congress*, 1956, pp. 218–239. Also in *Computers in Thought* (Fiegenbaum and Feldman, eds.), McGraw-Hill New York, 1963, pp. 134–152. Page numbers cited are from this reference.

O'Leary, D.J. (1984), The Propositional Logic of *Principia Mathematica*, and Some of its Forerunners. Presented at the Russell Conference, 1984, Trinity College, University of Toronto. Unpublished.

Russell, B. (1956), Letter to Herbert Simon dated 2 November 1956. Located at the Russell Archives, McMaster University.

Simon, H. (1956) Letter to Bertrand Russell dated 9 September 1957. Located at the Russell Archives, McMaster University.

Tiernay, P. (1983), Herbert Simon's simple economics. *Science* 83, **4**, No. 9, November 1983.

Wang, H. (1960), Toward mechanical mathematics, *IBM J. Res. Develop.*, 1960, 2–22. Also in *Logic, Computers and Sets*, Chelsea, New York, 1970.

Whitehead, A.N. and Russell, B. (1935), *Principia Mathematica*, 2nd ed., Cambridge University Press, Cambridge, UK.