What is mathematical logic? A survey

John N. Crossley[1]

School of Computer Science and Software Engineering,
Monash University, Clayton, Victoria, Australia 3800
John.Crossley@infotech.monash.edu.au

1 Introduction

What is mathematical logic? Mathematical logic is the application of mathematical techniques to logic.

What is logic? I believe I am following the ancient Greek philosopher Aristotle when I say that logic is the (correct) rearranging of facts to find the information that we want.

Logic has two aspects: formal and informal. In a sense logic belongs to everyone although we often accuse others of being illogical. Informal logic exists whenever we have a language. In particular Indian Logic has been known for a very long time.

Formal (often called, 'mathematical') logic has its origins in ancient Greece in the West with Aristotle. Mathematical logic has two sides: syntax and semantics. Syntax is how we say things; semantics is what we mean.

By looking at the way that we behave and the way the world behaves, Aristotle was able to elicit some basic laws. His style of categorizing logic led to the notion of the *syllogism*.

The most famous example of a syllogism is

$$\frac{\text{All men are mortal} \qquad \text{Socrates is a man}}{\text{[Therefore] Socrates is mortal}}$$

Nowadays we mathematicians would write this as

$$\frac{\begin{array}{c}\forall x(\mathit{Man}(x) \to \mathit{Mortal}\,(x)) \\ \mathit{Man}(S)\end{array}}{\mathit{Mortal}\,(S)} \tag{1}$$

One very general form of the above rule is

$$\frac{A \qquad (A \to B)}{B} \tag{2}$$

otherwise know as *modus ponens* or *detachment*: the A is 'detached' from the formula $(A \to B)$ leaving B. This is just one example of a logical rule.

This rule, and other rules of logic such as

$$\frac{A \quad B}{(A \wedge B)} \qquad \text{or} \qquad \frac{(A \wedge B)}{A}$$

where $\wedge$ is read 'and', have obvious interpretations. These rules come from observing how we use concepts.

This analytic approach was that taken by George Boole, an Irish mathematician in the nineteenth century. It is from his work that we have *Boolean algebra* or *Boolean logic* or, as it is often known today, *propositional calculus*.

Later in the nineteenth century Gottlob Frege developed a small suite of logical laws that are with us today and suffice for all of mathematics. These are the rules of the *predicate calculus*. The rules relate only to the syntax. Although they are abstracted from the way we talk and think, the meaning, the *semantics*, is something quite separate.

The most familiar example of semantics is given by *truth-tables* such as the one for conjunction (or 'and', $\wedge$):

$\wedge$	T	F
T	T	F
F	F	F

Here we are given the *truth-values* of the formulae A and B and we work out the truth of the conjunction $(A \wedge B)$ by taking the value for A at the side and the value for B at the top and finding the value where column and row intersect. In general, determining the truth of a syntactic expression (formula) A requires looking carefully at its constituents.

At this point we should pause because we are now dealing with two completely different things, two different aspects of languages.

On the one hand we have the rules for working with strings of symbols – rules such as *modus ponens* above; on the other hand we are looking at meanings. This latter is the study of *semantics*. The former is *syntax* – the study of the way that language is put together from symbols. In English this latter includes the way we put words together to make a sentence.

Consider something we, in essence, wrote earlier (in (1)), or rather a slight variant of it.

$$\forall x(M(x) \rightarrow D(x)) \tag{3}$$

If $M(x)$ is interpreted as 'x is a man' and $D(x)$ as 'x will die', then this formula is obviously true under this interpretation. However, if we interpret $M(x)$ as 'x is an animal' and $D(x)$ as 'x has at most two legs', then it is obviously false under this different interpretation.

Note that the formula itself is neither true nor false, it is a *formula* – a piece of syntax.

2 Syntax and Semantics

The first concern of mathematical logic is the relation between syntax and semantics.

First let us consider syntax a little more closely.

Any logic starts from certain basic symbols. In ordinary mathematical logic, and I will say what 'ordinary' means later in Section 4, we have symbols (letters) for variables: $x, y, z, \ldots$ and letters for predicates or properties, such as M and D above. We also have letters for relations and functions. For example, $L(x)$ might arise in a context: 'there is a line between the points x and y', or in a context 'x is married to y'. We use small letters for functions, $f_1, f_2 \ldots$. Such arise in arithmetic where we may use f_1 for $+$, or have a function i, say, in group theory that is intended to denote the inverse of an element. Applying these function letters (perhaps repeatedly) gives rise to (individual) *terms*.

Together all of these give us *atomic formulae* – basic formulae such as $L(x, y)$ or $M(f_3(x, f_1(y_2)))$ or $D(y)$.

Atomic formulae can the be joined together by *logical connectives* to form more complicated *formulae*, sometimes called '*well-formed formulae*', such as

$$(M(x) \wedge D(y)) \qquad \text{or} \qquad (M(x) \to D(x)) \qquad \text{or} \qquad \forall x(M(x) \to D(x)).$$

These have been joined using *connectives*: *propositional connectives* $\wedge$ (read as 'and'), $\vee$ (read as 'or'), $\to$ (read as 'implies'), $\neg$ (read as 'not'); and the *quantifiers*: $\forall$ (read as 'for all') and $\exists$ (read as 'there exists').[1] The precise rules for constructing formulae can be found in any logic textbook, such as [21].

We then have rules for generating more formulae. We have already met *modus ponens* in (2). This rule is also know as *implication elimination*, ($\to$-E): the $\to$ above the line is eliminated below it.

In Fig. 1 we give the rules[2] which were discovered in the late nineteenth century by Frege (though Leibniz knew many of them long before in the seventeenth century). The formulation here is known as *Natural Deduction* since the rules (with two exceptions) look very close to our actual practice in reasoning. In this formulation we do not have axioms but we may have hypotheses, i.e. formulae that we assume. Here an expression such as $\Delta, A \vdash B$ is read 'from the formulae in Δ and the formula A we can prove the formula B'.

The rules should be easy to read with at most two exceptions. These are ($\vee$-E) and ($\exists$-E). The former corresponds to proof by cases. We can paraphrase it as follows: 'If, from A we can prove C and from B we can prove C, then we can prove C from $(A \vee B)$'. Likewise ($\exists$-E) can be paraphrased as: 'If we can prove C from some particular x such that P (with x replacing y), and we can also prove that there is some y such that P, then we can prove C'.

[1] Some people avoid using negation, $\neg$. They employ a constant $\perp$ for the *false* formula. Then they use the formula $(A \to \perp)$ instead of $\neg A$.

[2] The phrase 'x is free/not free in [some formula]' is a technical condition that avoids misunderstandings.

$$\frac{}{\vdash A}\ (\text{Axiom-I}) \qquad \frac{}{A \vdash A}\ (\text{Ass-I})$$

$$\frac{\Delta, A \vdash B}{\Delta \vdash (A \to B)}\ (\to\text{-I}) \qquad \frac{\Delta \vdash A \quad \Delta' \vdash (A \to B)}{\Delta, \Delta' \vdash B}\ (\to\text{-E})$$

$$\frac{\Delta \vdash A}{\Delta \vdash \forall x.A}\ (\forall\text{-I}) \qquad \frac{\Delta \vdash \forall x.A}{\Delta \vdash A[c/x]}\ (\forall\text{-E})$$

x is free in A, not free in Δ

$$\frac{\Delta \vdash P[a/y]}{\Delta \vdash \exists y.P}\ (\exists\text{-I}) \qquad \frac{\Delta_1 \vdash \exists y.P \quad \Delta_2, P[x/y] \vdash C}{\Delta_1, \Delta_2 \vdash C}\ (\exists\text{-E})$$

where x is not free in C

$$\frac{\Delta \vdash A \quad \Delta' \vdash B}{\Delta, \Delta' \vdash (A \wedge B)}\ (\wedge\text{-I})$$

$$\frac{\Delta \vdash (A_1 \wedge A_2)}{\Delta \vdash A_1}\ (\wedge\text{-E}_1) \qquad \frac{\Delta \vdash (A_1 \wedge A_2)}{\Delta \vdash A_2}\ (\wedge\text{-E}_2)$$

$$\frac{\Delta \vdash A_1}{\Delta \vdash (A_1 \vee A_2)}\ (\vee\text{-I}_1) \qquad \frac{\Delta \vdash A_2}{\Delta \vdash (A_1 \vee A_2)}\ (\vee\text{-I}_2)$$

$$\frac{\Delta \vdash (A \vee B) \quad \Delta_1, A \vdash C \quad \Delta_2, B \vdash C}{\Delta_1, \Delta_2, \Delta \vdash C}\ (\vee\text{-E})$$

$$\frac{\Delta \vdash \bot}{\Delta \vdash A}\ (\bot\text{-E})$$

Fig. 1. The basic rules of predicate calculus.

It is obvious that these rules are ‘good’ rules if we just interpret them in an intuitive way. That is to say, when so interpreted they lead from true assertions to other true assertions.

We build proofs by repeatedly applying the rules. Since some of the rules have two *premises* (top lines) we actually get a tree. The tree is a *proof* of the formula at the root of the tree.

3 The Completeness Theorem and Model Theory

It should be quite surprising that when we analyze the sort of language we use in mathematics, and elsewhere too, that the few basic rules, given above and originally due to Frege, suffice to yield all those syntactic expression that are always true – and no others. This is the most dramatic result that mathematical logic produced in the first half of the twentieth century:

Theorem 1 (The Completeness Theorem). *There is a finite (small) set of logical rules which will yield all, and only, those formulae which are true under any interpretation.*

What is an *interpretation*? We have given a hint when discussing (3) above.

First we have to establish what domain we are talking about, e.g. people, or natural numbers. Then we have to interpret the predicates in the language and these will usually be interpreted as relations, one such example in the case of numbers is the relation $\leq$. Here is a different example. If we have a language involving $P(x, y)$ and we consider interpreting this predicate P as 'x divides y and we only allow x, y, etc. to be interpreted as natural numbers, then we can see that

$$(P(x, y) \wedge P(y, z)) \rightarrow P(x, z)$$

is always true.

On the other hand

$$(P(x, y) \vee P(y, x)) \tag{4}$$

is sometime true and sometimes false, while

$$\forall x \neg P(x, x) \tag{5}$$

is false since every number divides itself.

However if we interpret $P(x, y)$ as 'y is strictly greater than x', then (4) is sometimes true and sometimes false and (5) is true.

If we go to a completely different interpretation and let the variables range over human beings and now interpret $P(x, y)$ as 'x is married to y' then we get different results.

The actual formal definition of 'true in an interpretation' is quite complicated (see e.g. [21]), so we omit it here.

Those formulae, which are true under all possible interpretations, are called *universally valid* or, sometimes, simply 'valid'. One example is any formula of the form $(A \vee \neg A)$.

Formally we say that an interpretation, $\mathcal{M}$, is a *model* of a formula A (or a set of formulae Δ) if A is true in $\mathcal{M}$ (if every formula in Δ is true in $\mathcal{M}$).

In [16], Kreisel pointed out that the Completeness Theorem actually catches our intuitive notion of truth. The argument is simple. The Completeness Theorem says that every formula true in all (formal, mathematical) interpretations is provable. Clearly, anything provable is true in all interpretations (including informal ones). Finally anything true in all interpretations is true in all formal interpretations. Thus these three classes: A: provable in predicate calculus, B: true in all interpretations, and C: true in all formal interpretations, are such that $A \subseteq B \subseteq C \subseteq A$ and therefore A, B and C all coincide.

One half of the Completeness Theorem is more powerful in the following form.

Theorem 2 (The Compactness Theorem). *If a set, Σ, of formulae is consistent, then it has a model, i.e. an interpretation in which all formulae in Σ are true.*

Here *consistent* means, as you would expect, that we cannot prove a contradiction (such as $(A \wedge \neg A)$) from Σ.

The idea of model gives rise to *Model Theory.* This got a great impetus from the Completeness Theorem. Model Theory is the study of interpretations of a set of sentences of logic. The area is basically a grand exploitation of the semantics of logic. Surprisingly, one can obtain significant results simply by looking at the style of the sentences. At its simplest level it gives us beautiful results such as the following.

Theorem 3. *If a formula $\forall x_1 \forall x_2 \ldots \forall x_n A(x_1, \ldots, x_n)$ with no quantifiers inside the formula A is true in an interpretation then it is true in any (non-empty) sub-interpretation.*

A simple application of this is the following. If we write down axioms for a group involving the inverse function (as mentioned above) and a function m, say for group multiplication, then all these axioms can be written in the form $\forall x_1 \forall x_2 \ldots \forall x_n A(x_1, \ldots, x_n)$ with no quantifiers inside the formula A. It follows that if $\mathcal{G}$ is a model of these axioms, i.e. a group, then any non-empty subset of the elements of $\mathcal{G}$ closed under inverses and multiplication, is actually a sub-group.[3]

But Model Theory has much more powerful results too and allows us to do calculus in a new way: the *Non-standard analysis* of Abraham Robinson [22]. It has given us deep insights into group theory and into models of set theory, and has become an autonomous discipline (see [1]).

4 Intuitionist or Constructive Logic

Now there is not just one 'true' logic. At the beginning of the twentieth century Brouwer questioned the law of the excluded middle and gave a new interpretation to the syntax that we use in mathematics. He regarded a proof as telling us how to make a construction. For Brouwer, a proof of $(A \vee \neg A)$ was only acceptable if one could give *either* a proof of A *or* a proof of $\neg A$. In the ordinary mathematics of that time, as used by Hilbert, $(A \vee \neg A)$ was trivially true. This was, roughly speaking, based on the idea that A was either true or not, even if we do not know which: there was no middle alternative. This was unacceptable, indeed meaningless, for Brouwer.

Now the ordinary logic that is (still!) commonly used by mathematicians and most philosophers[4] relies on the law of the excluded middle:

$$(A \ \vee \ \neg A)$$

or, equivalently, the law of double negation:

$$\frac{\neg\neg A}{A}$$

[3] Some people take this as the definition of a sub-group but other examples can be given, see [20] or [4].

[4] But not as much by computer scientists.

Both of these are equivalent to our last rule in Fig. 1:

$$\frac{\Delta \vdash \bot}{\Delta \vdash A}\ (\bot\text{-E})$$

For many computer scientists and philosophers, it is better *not* to use this rule. [5] This gives so-called *constructive* or *intuitionist* logic.

When I was a student this kind of logic was regarded as odd.

In the 1960s Saul Kripke, then a young student, produced a different kind of interpretation in order to prove a Completeness Theorem for intuitionist logic. His ‘interpretations’ were not single models in the sense we met above, they were families of such interpretations with relations between them. They are know as *possible world* interpretations. Intuitively speaking, a formula is then provable if (and only if) it is true in all possible worlds. For a detailed treatment see [18] or [7].

Nowadays there are lots of different logics that all have their value and application. These logics include various kinds of modal logics. Modal logics have *modalities* which are indicated by new symbols. Thus we may have $\Diamond$ for *possibly* and $\Box$ for *necessarily*. One of the most famous of these is the logic called $S5$. This is a propositional logic in which the formulae are built up from *propositional variables* $p, q, \ldots$ using the propositional connectives (see above Section 2) and also allowing prefixing of formulae by $\Box$ or $\Diamond$. (Thus $(p \vee \Diamond(q \wedge \Box r))$ is a formula of $S5$. It has all the true formulae of ordinary propositional calculus as axioms together with the rules shown in Fig. 2.

Many of these logics also have completeness theorems analogous to Theorem 1. The proofs of these have similarities with the proof of the completeness of intuitionist logic. Indeed, Kripke proved completeness results for modal logics first [17] and only subsequently used his ideas there to prove the completeness of intuitionist logic. Details of such theorems may be found in [7].

$$\boxed{\begin{array}{cc} \text{If } \vdash A \text{ then } \vdash \Box A & \vdash \Box(A \to B) \to (\Box A \to \Box B) \\ \Box A \vdash A & \Diamond A \to \Box\Diamond A \end{array}}$$

Fig. 2. The rules for the modal logic $S5$.

[5] This is not a question of the rule being right or wrong, it is a question of what one can say about what computers do. There are certainly problems which a compute cannot decide (see below, Section 5, so the computer does not necessarily ‘know’ whether A is true or $\neg A$ is true.

5 Recursive Functions

When one turns to specific domains, for example the natural numbers, $0, 1, 2, \ldots$, the power of the logic changes. In order to study the natural numbers, the theory of formal arithmetic was described by axioms. These axioms are known as the Peano axioms but they are really due to Richard Dedekind (see [12]).

Kurt Gödel showed, in 1931, that there is no finite system of axioms that will give you all the true statements of arithmetic.[6] This is his famous *Incompleteness Theorem*.

So logic, in one sense, fails. But this weakness is also a strength. In proving the theorem Gödel developed the notion of *recursive* or *computable* function.[7] These functions are those that are *representable* in formal arithmetic. That is to say there are predicates that exactly mirror these functions. Later, however, it was found that there are many other ways of describing exactly the same functions.

Alan Turing [25] showed that the functions that can be computed on his idealized machines, now known as *Turing machines*, are exactly the same as Gödel's recursive functions. All our computers developed from this notion of Turing's.[8] It is now almost universally believed that a function is computable if, and only if, it can be programmed on some computer and, for every type of computer so far developed, it has been shown that those functions that can be computed are amongst Turing's computable functions. So recursive functions are *exactly* the functions we can compute. This work also showed that some functions can*not* be computed. In particular it is not possible to compute whether a given Turing machine with a given program will stop or not. This is the *Halting problem*.

The development of recursion theory has spawned a whole sub-industry of mathematical logic in which I have played a part.

In advancing the theory of computable functions several different approaches have been adopted including the *Turing machines* mentioned above and the *Lambda calculus* which we shall treat below (see Section 7).

One of the nicest approaches is that of *Shepherdson-Sturgis machines* or *Unlimited Register Machines* (see [24]). An excellent description of these can be found in Davis, Sigal and Weyuker [11]. We consider an abstract machine, very much like a modern computer. It has a finite number of *registers* in which natural numbers are stored. The registers are identified by variables $X_1, X_2, \ldots, Y, Z_1, Z_2, \ldots$. The X_i are the input registers. There is also an output register, where the answer will be found. The Z_j are auxiliary ones. Initially each register has zero in it. The whole programming language is very simple and is shown in Fig. 3. All recursive functions can be computed using programs in this language!

[6] In fact he even showed that there is no finite complete system of axiom schemes for formal arithmetic.

[7] To be precise, attention actually focussed on *partial* functions, those that may not be defined for all arguments.

[8] At least as far as we can tell. It seems obvious that John von Neumann used Turing's ideas but there is no record of him admitting to that! See Martin Davis [10].

$$\boxed{\begin{array}{c} Z \to 0 \\[1ex] Z \to Z+1 \\[1ex] \text{If } Z \neq 0 \text{ GO TO } L \end{array}}$$

Z stands for an arbitrary register name. Instructions are labelled with *labels* $L, L_1, L_2, \ldots$.

Fig. 3. The programming instructions for Shepherdson-Sturgis machines.

It was some time before people were able to prove that there are different *degrees* of computability. However this has now become a large industry, strongly developed by Sacks in the latter half of last century. (See Barwise [1].)

One of the areas which still requires considerable development, in my view, is that of *higher order* recursive functions. Although a great deal of work was done last century by Kleene, Sacks and others, our understanding of the kinds of functions that will take a program as an argument and always yield another program is not very well understood (even though such higher order functions are extensively used).

6 Set Theory

Besides arithmetic logic has also been applied to set theory. Indeed, one of the major impetuses for mathematical logic was the problem, in the nineteenth century, of the foundations of analysis, including, in particular, the infinitesimal calculus.

Cantor [3, 2] started investigating Fourier series and came across difficulties. The first thing was that one could count beyond infinity. This had been remarked long ago by Galileo in his *Discorsi e dimostrazioni matematiche intorno a due nuove scienze* (1638) (which I have not seen) where he showed there are the same number of square numbers as there are numbers. Here is an even simpler example. Suppose we arrange all the even numbers first and then all the odd numbers, then we get a list

$$0, 2, 4, \ldots, 1, 3, 5, \ldots.$$

So if we start counting we count $1, 2, 3, \ldots$ and then run out of counting numbers before we get to the end. Cantor introduced infinite (ordinal) numbers to be able to count this far. He counted

$$1, 2, 3, \ldots, \omega, \omega+1, \omega+2, \ldots.$$

And this process could be carried much further to $\omega \times \omega$ or ω^ω or even to $\omega^{\omega^{\omega^{\omega^{\vdots}}}}$ which is now known as ϵ_0 and even that was not the end. There is no end.

In developing his theories of large cardinal and ordinal numbers he introduced set theory. Sets can be thought of as being formed in two different styles. The first is by building them up, e.g. by taking unions of smaller sets, etc.. The other is by defining them by what they comprehend: defining a set as the set of x such that $A(x)$. The *naïve axiom of comprehension* says that $\{x : A(x)\}$ always exists. This gives rise to *Russell's paradox* (6):

If R is the set of x such that $x \notin x$, then we have both $x \in R$ and $x \notin R$.

The basic axioms of set theory were not troublesome and were formulated by Zermelo and Fraenkel. The axiom of comprehension, however, had to be circumscribed.

Thanks to the formulations available through the mathematization of logic, set theory has developed enormously.

Nevertheless this (finite) set of axiom schemes did not succeed in resolving all the open questions. Not merely was it incomplete (as it had to be because it was possible to develop arithmetic in set theory, see my remarks on incompleteness above in Section 5) but it did not resolve the status of the Axiom of Choice. The Axiom of Choice says that, given a non-empty set of non-empty sets, there is a set that has exactly one element from each of those sets. Russell [23] gives the nice example: given an infinite set of pairs of socks, how do you pick one sock from each pair?

The first problem was the problem of consistency. This has been attacked by building models of set theory. Gödel was the first to do this in spectacular fashion by building models of set theory within set theory, see [14]. However, such models do not solve all the problems.

Now Gödel [14] had established the consistency, but not the independence, of both the Axiom of Choice and the Continuum Hypothesis[9] back in 1940.

About the time I obtained my PhD (1963) Paul Cohen [5, 6] showed the independence of the Axiom of Choice (and the Continuum Hypothesis) from the other axioms of set theory. He applied a kind of model theory which we now know is closely related to the model theory that Saul Kripke used for modal and intuitionist logic. The technique is known as *forcing* and depends on being able to create possible worlds that behave in unusual ways. Since that time very many other statements with mathematical import have been proved independent of the other axioms of set theory.

The search for "nice" axioms for set theory continues. Although the concept of set appears simple from an intuitive point of view, we have no precise conception of what a set is. Moreover, since about 1970, the field of Set Theory has become extremely complicated and difficult. It is perhaps not surprising that even its practitioners use words such as 'morass'.

[9] The Continuum Hypothesis says that there are no infinite cardinal numbers between the smallest infinite cardinal number, that of the set of natural numbers and the cardinal number of the set of all subsets of the natural numbers.

7 Proof Theory

Finally there is *Proof Theory.* This studies the formal proofs in mathematical logic in the same way that one studies the real numbers, for example. Originally this was done by Gerhard Gentzen in the 1940s when he tried to prove that the Peano axioms for arithmetic are consistent.

Gödel had not only proved the Incompleteness Theorem (see above Section 5) but had also shown that it was impossible to prove the consistency of formal arithmetic in the theory itself. Gentzen devised a new way of presenting proofs, in fact closely related to the system of natural deduction we used in Section 2. He was then able to show that one could simplify certain proofs.

For example, suppose we are given two proofs: where the first is a proof of A:

$$\begin{array}{c}\vdots\\ A\end{array} \tag{6}$$

and the second is a proof of B from A from which we get a proof[10] of $A \to B$:

$$\dfrac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{(A \to B)} \tag{7}$$

Then suppose that we use *modus ponens* to remove the A from $(A \to B)$ thus: We now have a proof:

$$\dfrac{\begin{array}{c}\vdots\\ A\end{array} \qquad \dfrac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{(A \to B)}}{B} \tag{8}$$

But if instead we had simply put the first proof, (6) of A, on top of the second proof, (7) of $(A \to B)$, then we no longer need the hypothesis $[A]$ in the second proof in order to get a proof of B. Previously we had made an unnecessary detour since we already had a proof of B.

That is to say, we can *reduce* the proof in (8) to a simple proof of B of the form

$$\begin{array}{c}\vdots\\ A\\ \vdots\\ B\end{array}$$

This removal of unnecessary detours is known as *cut elimination.*

[10] The square brackets indicate that A can be discharged, i.e. is not needed for the proof of B, though it is for the proof of B, of course.

Gentzen [13] showed, by using a special form of induction, transfinite induction,[11] that there could be no proof of a contradiction, i.e. that arithmetic was consistent. The actual technique was to assume that there was a proof of a contradiction and then to reduce that proof, by cut elimination, until it was, in fact, of a very simple form. From there it was obvious that there could be no such proof.

Gentzen's techniques have been greatly developed by Feferman in the USA and Schütte and his group in Germany. These people have extended his results to much more complicated systems of logic than simple arithmetic.

However, the work started by Gentzen about sixty years ago has started a new and perhaps surprising industry in computer science. Although Gentzen was aware of the information contained in a proof, it was not until Bill Howard showed in his [15] that propositional calculus (which is even simpler than predicate calculus and can be, at least partially, identified with Boolean algebra) reflects the lambda calculus. Thereby the usefulness of Gentzen's work for producing computer programs was realized.

This is despite the fact that the lambda calculus was one of the ways that recursive functions were developed. It was an alternative to Turing machines. More recently lambda calculus has formed the basis for the programming language, LISP, and in fact one can obtain programs directly from formal proofs by developing Howard's ideas further. For more about this you may care to listen to my lecture later in this conference [8]: *What is the difference between proofs and programs?* which is devoted to this topic of extracting programs from proofs.

8 Conclusion

Over little more than a century mathematical logic has developed from nothing to a very multi-faceted subject. It has thrown a great deal of light on many areas of philosophy: particularly through modal logics; mathematics: especially through set theory; and computer science: through the analysis it has permitted and the (correct) programs it has allowed to be generated. It has also shown the limits of computability.

At present, mathematical logic encompasses model theory, set theory, recursion theory and proof theory. Although modal logics have long been used, especially by philosophers, in my lifetime I believe that the most important change in mathematical logic has been the development of many, many other kinds of logics, which have supplemented the standard or classical one used in mathematics. I have touched on but a few of these. Some of the others are the subjects of other lectures in this conference. There is still more work to be done and I hope I have encouraged you to find out what there is to do in the areas in which you yourselves are interested.

[11] In fact he only needed transfinite induction up to ϵ_0, see Section 6, in order to prove his result.

References

1. Jon Barwise, editor. *Handbook of mathematical logic.* North-Holland Pub. Co., 1977.
2. Georg Cantor. Über einen die trigonometrischen Reihen betreffenden Lehrsatz. *Journal f. reine und angew. Math.*, 72:130–138, 1870.
3. Georg Cantor. Über die Ausdehnung eines Satzes aus der Theorie der trigonometrischen Reihen. *Mathematische Annalen*, 5:123–132, 1872.
4. Chen Chung Chang and H Jerome Keisler. *Model theory.* North-Holland Pub. Co., 1973. 3rd ed. 1990.
5. Paul Joseph Cohen. The independence of the continuum hypothesis. *Proc. Nat. Acad. Sci. U.S.A.*, 50:1143–1148, 1963.
6. Paul Joseph Cohen. The independence of the continuum hypothesis. II. *Proc. Nat. Acad. Sci. U.S.A.*, 51:105–110, 1964.
7. Maxwell John Cresswell and George Edward Hughes. *A New Introduction to Modal Logic.* Taylor & Francis Inc., 1996.
8. John Newsome Crossley. What is the difference between proofs and programs? Lecture at this conference.
9. John Newsome Crossley, Chris Brickhill, Christopher John Ash(†), John Colin Stillwell, and Neil Hale Williams. *What is mathematical logic?* Oxford University Press, 1972. Latest edition, Dover, 1990.
10. Martin Davis. *Engines of logic : mathematicians and the origin of the computer.* W. W. Norton, 2001.
11. Martin D. Davis, Ron Sigal, and Elaine J. Weyuker. *Computability, complexity, and languages: fundamentals of theoretical computer science.* Academic Press, Harcourt, Brace, Boston, MA, 2nd edition, 1994.
12. Richard Dedekind. The nature and meaning of numbers. In *Essays on theory of numbers.* Dover, 1901 (1963). Translation of *Was sind und was sollen die Zahlen?*
13. Michael E.Szabo, editor. *The collected papers of Gerhard Gentzen.* North-Holland Pub. Co., Amsterdam, 1969.
14. Kurt Gödel. *The consistency of the axiom of choice and of the generalized continuum-hypothesis with the axioms of set theory.* Princeton University Press, 1940.
15. William Howard. The formulae-as-types notion of construction. In John Roger Hindley and Jonathan Seldin, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1969.
16. Georg Kreisel. Mathematical logic. In Thomas Lorie Saaty, editor, *Lectures on modern mathematics*, volume 3, pages 95–195. Wiley, 1965.
17. Saul Kripke. A completeness theorem in modal logic. *JSL*, 24:1–14, 1959.
18. Saul Kripke. Semantical analysis of intuitionistic logic I. In John Newsome Crossley and Michael Anthony Eardley Dummett, editors, *Formal Systems and Recurive Functions.* North-Holland, Amsterdam, 1965.
19. Edward John Lemmon. *Beginning logic.* Nelson, 1971.
20. Roger C Lyndon. Properties preserved under homomorphism. *Pacific J. of Mathematics*, 9:143–154, 1959.
21. Elliott Mendelson. *Introduction to mathematical logic.* Chapman & Hall, 4th edition, 1997.
22. Abraham Robinson. *Non-standard analysis.* North-Holland Pub. Co., 1966.
23. Bertrand Russell. *Introduction to Mathematical Philosophy.* G. Allen and Unwin, London, 1970. First published in 1919, thirteenth impression, 1970.

24. John Cedric Shepherdson and H.E. Sturgis. Computability of recursive functions. *J. Assoc. Comput. Mach.*, 10:217–255, 1963.
25. Alan Matheson Turing. Computability and lambda-definability. *JSL*, 2:153–163, 1937.
26. Jean van Heijenoort. *From Frege to Gödel*. Harvard University Press, 1967.