Teaching mathematics by computer

A. D. McGettrick

Department of Computer Science, University of Strathclyde, Livingstone Tower, 26 Richmond Street, Glasgow G1 1XH

Computer assisted learning, CAL, is a topic which has recently aroused considerable interest in this country. The British Government is currently spending some £2m on various projects which aim to demonstrate the feasibility and practicality of using computers as a teaching aid in education.

In this paper we shall be concerned with the theoretical background associated with the problem of teaching mathematics by means of CAL techniques. We shall look at several aspects of secondary school and early university mathematics and discuss the problems that arise.

(Received January 1976)

1. Background

Computer assisted learning, often abbreviated to CAL, is a topic which has recently aroused considerable interest in this country. The British Government through the National Development Programme in Computer Assisted Learning (Hooper, 1974) is currently spending some £2m on various projects which aim to demonstrate the feasibility and practicality of using computers as a teaching aid in education.

Assume an environment in which the computer as part of its task poses questions on some particular subject matter for students to answer. A question might appear on a VDU or on a teletype and the student has to reply by typing an appropriate answer. At this stage the computer must decide whether or not the supplied answer is acceptable. It is vitally important that appropriate action is taken. The whole credibility of a CAL system can depend on a sensible reaction on the part of the computer. In what follows this will be referred to as 'the answer recognition problem'.

Having reacted in an appropriate manner the computer can keep track of the progress of each individual student and gauge his speed of progress accordingly. At various moments in time it may be appropriate to

(*a*) provide easier or harder examples

(*b*) provide more or less problems on a particular topic

(*c*) provide revision exercises

(*d*) ask a student to see his teacher for some special purpose

(*e*) refer a student to a book or some other material.

In sophisticated CAL systems it might happen that other audio-visual aids are available perhaps being controlled by the computer. In such circumstances it may be appropriate to switch on a tape tecorder, film projector, etc. for the student to receive some prerecorded instruction.

In this paper we shall be concerned with the theoretical background associated with the problem of teaching mathematics by means of CAL techniques. We shall look at several aspects of secondary school and early university mathematics and discuss the problems that arise.

2. The role of the computer

Whenever one mentions CAL one must be prepared to expect questions about the role played by the computer. Answers inevitably involve discussions about the ability of the computer to:

(*a*) interact

(*b*) regulate the pace of lessons and exercises

(*c*) cope with several students simultaneously

(*d*) assess and evaluate both students and lessons and handle large volumes of information.

In teaching mathematics, and indeed some other subjects, one can arrange that other facets of the computer can be utilised. For example,

(*e*) using random number generators one can arrange that parameters be suitably chosen at run time and consequently expressions such as $ax^2 + bx + c$ might appear to the student as

$$x^2,\ 1 - x^2,\ 4,\ x, \text{ etc.}$$

(*f*) one can select one from a set of, say, functions each of which might have parameters (the selection might again be performed by a suitable random number generator); a greater degree of flexibility and variety can be achieved by the composition of several such functions (note, for example integration by change of variable)

(*g*) if a suitable VDU screen is available one can show for instance the rotation or transformation of triangles, conics, etc. caused by changes of axes; one could also show solids, the rotation of solids and their projection from various angles.

3. The answer recognition problem

Even in traditional methods of teaching mathematics there are often substantial difficulties in deciding whether or not an answer obtained by a student is correct. It should therefore come as no surprise that it also raises considerable problems from a CAL point of view.

Some of the more common approaches to solving this problem are given below together with some remarks about their merits or otherwise. Most of these solutions can be used in appropriate circumstances.

The reader should imagine that the answer takes the form of some kind of expression. Most CAL programs are designed in this way. The expressions will usually be arithmetic or algebraic expressions. This covers both constants or relatively complex expressions involving perhaps several variables together with the usual arithmetic operators and even sin, cos, log, exp, π, square roots, etc.

But the term 'expression' can also include expressions involving sets, the propositional calculus, the predicate calculus, etc. More complicated expressions could involve a mixture of the various types and might therefore include both quantifiers and arithmetic expressions.

3.1. *Accept only an exact sequence of characters*

This simple strategy is very limited but can be of use when the required answer itself is simple. It might be used to effect in CAL programs designed to teach primary school arithmetic, for example. But in more complex cases it is inadequate. In teaching mathematics by CAL this scheme has a limited relevance. Most results will take the form of expressions and this

strategy pays no attention to the mathematical structure of expressions.

The scheme can be modified in several ways. One can allow spaces to occur between characters, one can allow simple editing in the form of corrections to mistypes, etc. but these add little. One could also allow not just one sequence of characters but admit any one of several such sequences. The more sequences one admits the higher the probability that correct replies will be recognised. Even with these modifications there is still a considerable chance of a correct answer being marked wrong. Moreover, the task of preparing tables of acceptable answers can be tedious in the extreme.

3.2. *Accepting an answer in arbitrary form*

Consider now the possibility of allowing a student the freedom to supply his answer in whatever manner he chooses. In the first place note that such a degree of flexibility may not always be desirable. In a classroom a student, when asked to differentiate x^2, would not be expected and probably not permitted to supply his answer $2x$ in an arbitrary format. There are occasions when a certain degree of freedom is desirable, there are occasions when it is not.

But restrictions of another kind now come into the picture. If a student is permitted an arbitrary degree of freedom then in general terms the computer has the task of deciding whether two arbitrary expressions are equivalent, i.e. yield the same result for all values of their variables. Whether this is possible will depend on the kind of expressions under discussion. We shall look at this in more detail in Section 4. But if for the moment we consider arithmetical expressions it is not possible for the computer to decide whether two such expressions are equivalent. In fact Richardson (1968) has shown that it is impossible to obtain an algorithm to determine whether an arbitrary expression from a more limited class of expressions is identically zero. The expressions considered by Richardson are generated from the rational numbers, the two real numbers π and $\log_e 2$, and the indeterminate x using:

(*a*) the operations of addition and multiplication

(*b*) the sin function

(*c*) the exponential function

(*d*) the absolute value function

(*e*) substitution.

Indeed since it is now known that Hilbert's Tenth Problem (Hilbert, 1901) is unsolvable (Davis, 1973; and Matiasevich, 1970) a result of Caviness (1967) shows that the same result applies to the smaller class of expressions generated as above but with $\log_e 2$ and the exponential function omitted.

Other results similar to those described above will be discussed later in Section 4.

3.3. *Algebraic manipulation techniques*

At this stage there is a temptation to abandon all hope of being able to produce a satisfactory CAL system for teaching mathematics. But let us consider some further possibilities. Suppose we try to arrange that the computer, in attempting to recognise an answer, performs some mathematics. This will usually take the form of arithmetic calculations or algebraic manipulation but may also include logical operations, etc.

The computer could attempt to determine the equivalence of two arithmetic expressions by one of the following methods:

(*a*) evaluate both expressions using finite field arithmetic

(*b*) evaluate both expressions using floating point arithmetic

(*c*) reduce both expressions to canonical form

(*d*) reduce the difference of the two expressions to a standard form which is 0 for all expressions equivalent to 0.

In Martin (1971) these methods are examined and their relative merits discussed. Usually the acceptability or otherwise of a particular method will depend on the class of expressions being used. But from our point of view the relevant observations from Martin are as follows.

Method (*a*) breaks down badly when elementary functions such as sin, exp, etc. appear in expressions. Moreover there is a possibility that the method will decide erroneously that two nonequivalent expressions are in fact equivalent. Such an event we shall refer to as a random match.

For method (*b*) to be of use it is necessary to select the points at which evaluation has to be performed. Such problems as overflow and dangerous rounding errors must be avoided. Moreover, as in method (*a*) there is the possibility of a random match. But perhaps worse there is also the possibility that this method will decide that equivalent expressions are not equivalent.

The relevance of methods (*c*) and (*d*) will depend on the existence of suitable algorithms for reducing expressions to standard or canonical form. This topic is discussed in detail in Section 4. But note that, as a result of the work of Richardson and others, it will in general be necessary to limit the class of expressions one is prepared to consider.

Roughly speaking this concludes Martin's remarks. Note that results obtained by using methods (*a*) and (*b*) can be improved by performing the calculations for several values of the variables. Moreover, method (*b*) can be modified to allow evaluation of expressions using integers rather than floating point numbers in an attempt to avoid approximations, rounding errors, etc. But this is of limited applicability since it depends on the fact that constants in expressions are integers, that division leading to real numbers is not involved, that sin (x), exp (x), etc. do not appear, and so on. Even then overflow is still liable to occur.

Methods other than (*a*)–(*d*) can also be applied. Consider:

(*e*) evaluate both expressions using *p*-adic numbers; work in this area is being carried out by D. Y. Y. Yun at MIT.

(*f*) evaluate both expressions using interval arithmetic; work in this area has been carried out by Wittig (Fitch, 1973).

Martin's remarks were directed to the problem of determining the equivalence of algebraic expressions from the point of view of algebraic manipulation systems. We have a different point of view and therefore some further remarks should be made. Consider again method (*a*) possibly with the added ability to evaluate expressions for several values of the variables. In certain branches of mathematics, e.g. topics in number theory, this approach would be entirely relevant since one might be teaching some aspect of the theory of finite fields. In such cases this provides a complete answer to the answer recognition problem. Consider for example the theorem (Hunter, 1964)

if p is a prime and if $a_n \not\equiv 0 \pmod p$ then the algebraic congruence

$$a_n x^n + a_{n-1} + x^{n-1} + \ldots + a_1 x + a_0 \equiv 0 \pmod p$$

has not more than n incongruent roots modulo p.

Thus one has only to evaluate the polynomial for at most n different values of x to test if the polynomial is identically 0. Avoiding overflow will usually be relatively simple.

3.4. *Pattern matching*

One might approach the answer recognition problem by applying pattern matching techniques. Indeed this is not unrelated to the earlier discussion on standard and canonical forms.

As Fateman (1972) points out there are many problems associated with pattern matching. A pattern can be regarded as

(*a*) a lexical entity, as in SNOBOL for example (Farber *et al.*, 1964)

(*b*) a syntactic entity, as in FORMULA ALGOL (Perlis *et al.*, 1966), AMBITS (Christensen, 1965) or FAMOUS (Fenichel, 1966)

(*c*) a semantic entity, as in SIN (Moses, 1971b) or SAINT (Slagle, 1961).

From our point of view the first two of these alternatives will usually be unsatisfactory. Approach (*a*) might be adopted if we were interested in arbitrary strings of characters and were not concerned with their structure (the balanced strings of SNOBOL help to a limited extent). The strings encountered in our environment will be expressions with a structure and a meaning. Again approach (*b*) is not entirely satisfactory since expressions such as 0**0 are syntactically correct but semantically wrong.

Approach (*c*) attempts to take account of meaning as well as structure and thus tends to be more relevant. But nasty problems continue to appear. Does 1 match the pattern $A**B$? If so what are the values of A and B? Fateman (1972) discusses this and similar points in some detail but concludes that almost arbitrary decisions have to be made.

Pattern matching cannot theoretically provide a complete solution to our problem as witnessed by the earlier result of Richardson. Even in the MACSYMA system at MIT which provides a mechanism whereby the system can be 'taught' new transformations to be applied to expressions or new identities (see Fateman again) there are severe inadequacies.

3.5. *Other strategies*

One of the common ways in which student answers differ from expected answers is in the presence of extra brackets. If brackets can be removed this difficulty can be overcome. Reducing expressions to prefix or postfix notation will result in a bracket-free representation of expressions. But again this simple approach is usually inadequate.

In lessons on integration the student has usually to supply a constant of integration. This can often appear in a variety of ways, e.g.

$$\tfrac{1}{2}\log(x^2+1)+C \text{ and } \log A(x^2+1)^{\frac{1}{2}}$$

are equivalent in the sense that they are both indefinite integrals with respect to x of

$$\frac{x}{x^2+1}$$

To overcome this problem one can differentiate the answer supplied by the student thereby removing the constants of integration (hopefully!). The result thereby obtained and the integrand can then be tested for equivalence.

By means of multiple choice techniques one can attempt to deal with the answer recognition problem. Apart from the difficulties involved in preparing suitable sets of possible answers there are substantial dangers associated with this approach. In teaching integration for instance a student can reply correctly if he can differentiate each of the expressions from which he has to select an answer.

4. Theoretical results

Partial solutions to the answer recognition problem would appear to lie in the appropriate use of the methods discussed above. With this in mind we now look at theoretical results which might aid in the development of CAL material. In particular these will include results that have been obtained in the area of standard or canonical forms.

We begin by giving results involving arithmetic and algebraic expressions. We conclude by mentioning some relevant results from mathematical logic.

4.1. *On canonical forms*

Fitch (1973) discusses some results involving canonical forms. He shows that

> the class of constants formed by the addition, subtraction, multiplication and division of rational powers of rational numbers has a canonical form

and

> the class of expressions formed by the addition, subtraction, multiplication and division of rational powers of polynomials in n variables with rational coefficients has a canonical form.

The canonical forms are given and algorithms provided for determining whether or not two constants or expressions of the type mentioned are identical. Fitch's paper ends with a result that

> the class of expressions formed by the addition, subtraction, multiplication and division of rational powers of polynomials in x and x^x with rational coefficients has a canonical form.

Results of a different nature are obtained by Caviness (1967, 1970). He defines a class of expressions, his FOE or first order exponentials, as follows. These are generated from

(*a*) the rationals

(*b*) the complex number i

(*c*) the variables $x_1, x_2, \ldots, x_n$

using the operations of addition, multiplication and restricted composition and the exponential function. The composition is restricted in that nested exponentials are not allowed (hence first order). Caviness' canonical form is

$$P_1 \exp(S_1) + P_2 \exp(S_2) + \ldots + P_k \exp(S_k)$$

where each P_i and S_i is in canonical form, each P_i is non-zero and the S_i's are ordered in some sense. As a result of this, of course, Caviness is able to state results involving the trigonometric and hyperbolic functions. With the introduction of division further slight extensions are possible.

In the same work Caviness considers another class of expressions, his radical expressions. These are obtained from:

(*a*) the rationals

(*b*) the variable x

using the operations of addition, subtraction, multiplication and division and raising expressions to rational powers (these may not be nested). Again Caviness shows that these can be expressed in canonical form. However his algorithm is impracticable. More recently Fateman (1972) improved on Caviness' result by producing a practical result and extending the work to include the possibility of several variables.

4.2. *Conjectured results*

Many results on algebraic simplification have been proved subject to some as yet unproved conjectures. These conjectures tend to be of a similar nature in that they relate to the linear dependence or independence of a set of constants.

Brown (1969) defines a class of expressions, the REX or regular exponential expressions. These are obtained from

(*a*) the rational integers

(*b*) the complex number i and π

(*c*) the variables $x_1, x_2, \ldots, x_n$

(*d*) the exponential function exp

using the rational operations, i.e. $+$, $-$, $\times$ and $/$, and substitution. Note that these involve the trigonometric and hyperbolic functions. In his paper Brown gives a method of simplifying REX expressions. Provided the conjecture given below is true Brown proves that the only simplified REX expression equivalent to 0 is 0 itself. Brown's conjecture is:

> Let $p_1, p_2, \ldots, p_n$ be non-zero rational exponential expressions such that the set $\{p_1, p_2, \ldots, p_k, i\pi\}$ is linearly independent over the rationals. Then the set $\{\exp(p_1), \ldots,$

$\exp(p_k), x_1, x_2, \ldots, x_n, \pi\}$ is algebraically independent over the rationals.

Moses (1971a) discusses Brown's algorithm at some length.

Subject to another similar conjecture Caviness (1970) describes a simplification algorithm for his class of exponential expressions. These are obtained from

(*a*) the rational numbers

(*b*) the complex number i

(*c*) the variable x

(*d*) the exponential function

using the operations of addition and multiplication and unrestricted composition.

Note that there is no division and only a single variable is used. The above expressions are similar to the FOE expressions defined earlier but only a single variable is now permitted and arbitrary nesting of polynomials is allowed. The defined expressions might therefore be classified as exponential polynomials. Caviness gives a canonical form again looking like

$$P_1 \exp(S_1) + \ldots + P_k \exp(S_k) .$$

The various S_i are themselves distinct exponential polynomials and the P_i are non-zero polynomials. Each S_i and P_i is in canonical form and the S_i are ordered.

Moses (1971a) himself mentions another similar algorithm to allow the simplification of REX expressions which do not involve i or π or the nesting of exponentials. Again the result depends on an unproved conjecture.

All the expression classes so far discussed in this paragraph involve rational operations together with the exponential function. Richardson (1966; 1969) defines an algorithm for a class of expressions which differs from the REX expressions in that it involves:

(*a*) no complex number i

(*b*) only a single variable x

(*c*) the logarithmic function $\log |x|$.

Like the exponential function, the logarithmic functions can be nested to any depth. Richardson's algorithm will decide whether or not an arbitrary expression from this class is equivalent to zero. However it is necessary to know that the expressions it manipulates are totally defined throughout the range over which zero-equivalence is to be tested. Again Richardson's algorithm depends essentially on conjectures of the kind already mentioned.

4.3. *Undecidability results*

Apart from the already stated undecidability results there are other such results which are also of interest.

Wang (1974) defines G_n to be the class of expressions generated from:

(*a*) the rationals and π

(*b*) the indeterminates $x_1, x_2, \ldots, x_n$

using:

(*a*) the operations of addition and multiplication

(*b*) the sin function

(*c*) composition.

He then proves that the problem of deciding the existence of a real number r with the property that

$$f(r) = 0$$

for any f in G_1 is recursively undecidable. This result is then used to prove that the problem of deciding whether

$$\int_{-\infty}^{\infty} \frac{dx}{(x^2 + 1)f^2(x)}$$

converges is also recursively undecidable.

Richardson (1968) shows that for a certain class of expressions the problem of deciding whether or not the expressions of that class had integrals which could be expressed in closed form is undecidable. (Of course, this has implications for differential equations). The class of functions considered by Richardson is obtained from:

(*a*) the elementary constants

(*b*) the variable x

(*c*) e^x, $\sin(x)$ and $\log |x|$

using the operations of addition, subtraction, multiplication and division and substitution.

The elementary constants are members of the smallest set of real numbers which form a closed real field containing 1 and π and are closed under the application of the functions e^x, $\sin(x)$ and $\log |x|$.

However, if the class of functions is restricted then one can make certain assertions about the form of the integral and about algorithms for evaluating it. See the work of Risch (1968, 1969, 1970) which is based on earlier work done by Liouville.

The proof of the unsolvability of Hilbert's Tenth Problem (Hilbert, 1901) implies that it is not possible to produce an algorithm to determine whether an arbitrary polynomial with integer coefficients has integral roots. It is assumed that the polynomials have arbitrarily many variables and arbitrary degree. Similar results have been obtained for exponential Diophantine equations by Davis *et al.* (1961).

4.4. *Results from logic*

Logical expressions can be used in answer to several types of mathematical questions. As a result it is of interest to look at the possibility of dealing with such expressions in the context of CAL.

Expressions of the propositional calculus involve variables and a set of logical connectives. These connectives or operations can take various forms but all of these can be expressed in terms of $\wedge$, $\vee$ and $\neg$. Other operators include implication, equivalence, etc. Such expressions frequently appear in set theory, boolean algebra, switching theory and, of course, mathematical logic itself. It is always possible to decide whether or not two arbitrary expressions in the propositional calculus are identical for one has only to reduce them both to minterm or maxterm canonical form and compare the results. Essentially this involves comparing the truth tables.

If we move from logical expressions in the propositional calculus to more complex forms of logical expressions the matter is not so easily resolved.

The first order predicate calculus is a formal system which makes use of:

(*a*) logical constants, values in some domain D

(*b*) logical variables whose values extend over D

(*c*) functional constants, functions whose domain is D^n and whose codomain is D

(*d*) predicates, i.e. functions with domain D^n and codomain {**true, false**}

These objects are connected by means of the usual logical symbols

$$\wedge, \vee, \neg, \supset \text{ (implies) and } \equiv \text{ (equivalence)}$$

together with the quantifiers ∀ (the universal quantifier meaning 'for all') and ∃ (the existential quantifier meaning 'there exists'). In the first order predicate calculus these quantifiers can qualify only logical variables. However this is sufficient for many purposes. For a precise definition of the predicate calculus see Manna (1974). Unfortunately there is no decision procedure for deciding:

(*a*) whether an arbitrary expression in the predicate calculus is always true, i.e. is valid

(*b*) whether two arbitrary expressions are equivalent, i.e. yield the same value for all values of the variables.

Note that (*a*) and (*b*) above are not unrelated.

These results, of course, cause trouble from a CAL point of view. But as with arithmetic expressions there are some escape routes. Every first order expression can be reduced in a series of steps to prenex normal form (either prenex conjunctive or prenex disjunctive normal form). These forms are such that the quantifiers $\forall$ and $\exists$ always appear at the start of an expression and not at an arbitrary point within them. Let us call the sequence of $\forall$'s and $\exists$'s the prefix of the logical expression. If the prefix happens to be of a particular kind it does become possible to produce programs to determine the validity of first order expressions. If we let x_i, y_j, z_k denote the logical variables these prefixes include

$$\forall x_1 \ldots \forall x_m \exists y_1 \ldots \exists y_n$$
$$\forall x_1 \ldots \forall x_m \exists y \forall z_1 \ldots \forall z_n$$
$$\forall x_1 \ldots \forall x_m \exists y_1 \exists y_2 \forall z_1 \ldots \forall z_n$$

But they do not include prefixes of the form

$$\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n$$
$$\forall x_1 \ldots \forall x_m \exists y_1 \exists y_2 \exists y_3 \forall z_1 \ldots \forall z_n .$$

One can go from the first order predicate calculus to higher orders of logic by allowing the quantifiers $\forall$ and $\exists$ to qualify not just variables but also functions and predicates, by allowing other operators or connectives such as equality and so on. But such systems are even more complex and results are correspondingly less satisfactory (Manna, 1974).

5. Consequences of theoretical results

It must be the aim of the CAL programmer to avoid the possibility of encountering any of the theoretical limitations discussed above. Fortunately the programmer (teacher) has the ability to choose what questions he wishes to ask and also what form of answer he will accept. The correct choice is no easy matter. But at least the situation is under his control.

The results discussed in Section 4 imply, for example, that a student cannot be allowed the freedom to ask the system to integrate an arbitrary expression. But he could be allowed to ask for the integral of an expression of a particular form.

It is too severe to demand of any CAL system that it should always produce the correct response and, for example, always mark a correct answer correct. However the proper philosophy would appear to be that a correct answer should never be marked wrong. This implies that if a student responds by typing a result which is in some sense too complex (it might contain some 'unknown' functions) then the student can be asked to simplify his answer. A more helpful response might ask to simplify by removing functions, variables, etc. which can be named in the computer's message. This resembles closely the kind of response that might be provided by a helpful and tolerant teacher.

How should one tackle the problem of preparing lessons? One possible approach is to learn from the work of the pioneers of algebraic manipulation systems and to use this in conjunction with the randomness provided by random number generators. In dealing with:

(*a*) symbolic integration (indefinite integrals) see the work of Slagle (1961), Moses (1967 and 1971b), Manove (1968) and Risch (1968, 1969, 1970),

(*b*) symbolic integration (definite integrals) see the references included in (*a*) above together with the work of Wang (1971),

(*c*) symbolic integration (contour integration) see the references already given in (*a*) and (*b*),

(*d*) differential equations see Moses (1967) and Manove (1968),

(*e*) limits of sequences see Wang's (1971) work on DELIMITER.

6. Conclusion

In this paper we have attempted to outline some of theory and techniques which might underlie a CAL system for teaching mathematics.

There is much work to be done on this area. Apart from the fact that more theoretical results are required one of the large tasks is the production of useful material in the form of programs, i.e. lessons, exercises, etc. For the success of such systems it must be possible to tax good students and yet cater for weaker students. It should also be possible for a teacher to prepare his own lessons with relative ease.

These conflicting aims together with the substantial theoretical limitations make this a challenging field.

Acknowledgements

I am indebted to the members of a CAL group in Glasgow University for allowing me to share some of their experiences in producing CAL material for mathematics. I am particularly indebted to Dr John Hunter and Mr Dennis Daly for many interesting discussions.

References

BROWN, W. S. (1969). Rational exponential expressions and a conjecture concerning π and e, *Amer. Math. Monthly*, Vol. 76, Jan. 1969, pp. 28-34.

CAVINESS, B. F. (1967). On canonical forms and simplification, Carnegie Institute of Technology, Ph.D. Thesis.

CAVINESS, B. F. (1970). On canonical forms and simplification, *JACM*, Vol. 17, No. 2, pp. 385-396.

CHRISTENSEN, C. (1965). Examples of symbol manipulation in the AMBIT programming language, Proc. ACM 20th National Conference, Cleveland, Ohio, pp. 247-261.

DAVIS, M. (1973). Hilbert's Tenth Problem is unsolvable, *Amer. Math. Monthly*, Vol. 80, No. 3, pp. 233-269.

DAVIS, M., PUTMAN, H., and ROBINSON, J. (1961). The decision problem for exponential diophantine equations, *Annals of Math*, Vol. 74, No. 3, pp. 425-436.

FARBER, D. *et al.* (1964). SNOBOL, A string manipulation language, *JACM*, Vol. 11, No. 1, pp. 21-30.

FATEMAN, R. J. (1972). Essays in algebraic simplification, *Project MAC*, MAC-TR-95, MIT, Cambridge, Mass., April 1972.

FENICHEL, R. (1966). An on-line system for Algebraic Simplification, doctoral thesis, Harvard University, July 1966.

FITCH, J. (1973). On algebraic simplification, *The Computer Journal*, Vol. 16, No. 1, pp. 23-27.

HILBERT, D. (1901). Mathematische Probleme, *Bull. Amer. Math. Soc.*, Vol. 8, (1901-2), pp. 437-479.

HOOPER, R. (1974). The National Development Programme in Computer Assisted Learning—Origins and Starting-Point, *Programmed Learning and Educational Technology*, Vol. 11, No. 2, pp. 59-73.

HUNTER, J. (1964). *Number Theory*, Oliver and Boyd, 1964.

MANNA, Z. (1974). *Mathematical Theory of Computation*, McGraw-Hill, 1974.

MANOVE, M. *et al.* (1968). Rational Functions in MATHLAB, Proc. *IFIP Conf. on Symbolic Manipulation Languages*, PISA, Italy 1968.

MARTIN, W. A. (1971). Determining the Equivalence of Algebraic Expressions by Hash Coding, *JACM*, Vol. 18, No. 4, pp. 549-558.

MATIASEVICH, J. V. (1970). Diophantine nature of enumerable sets, *Akademiia Nauk. SSSR Doklady* 191, No. 2, pp. 279-282.

MOSES, J. (1971a). Algebraic Simplification: A Guide for the Perplexed, *CACM*, Vol. 14, No. 8, pp. 527-537.

MOSES, J. (1971b). Symbolic Integration: The Stormy Decade, *CACM*, Vol. 14, No. 8, pp. 548-560.
MOSES, J. (1967). Symbolic Integration, *Project MAC*, MAC-TR-47, MIT, Cambridge, Mass., December 1967 (also available from the Defence Document Centre, AD No. 662666).
MACSYMA reference manual (1974). MATHLAB group, MIT, Project MAC, Cambridge, Mass., January 1974.
PERLIS, A., ITTURIAGA, R., and STANDISH, T. (1966). A definition of FORMULA ALGOL, paper presented at first symposium on *Symbolic and Algebraic Manipulation of ACM*, Washington DC.
RICHARDSON, D. (1966). Some unsolvable problems involving functions of a real variable, Ph.D. Diss., University of Bristol, England.
RICHARDSON, D. (1968). Some unsolvable problems involving elementary functions of a real variable, *J. Symbolic Logic*, Vol. 33, pp. 511-520.
RICHARDSON, D. (1969). Solution of the identity problem for integral exponential functions, *Zeitschr. Math. Logik*, Vol. 15, pp. 333-340.
RISCH, R. (1968). On the integration of elementary functions which are built up using algebraic operations, Rep. SP-2801-002, Syst. Develop. Corp., Santa Monica, California.
RISCH, R. (1969a). The problem of integration in finite terms, *Trans. AMS*, Vol. 139, May 1969, pp. 167-189.
RISCH, R. (1969b). Further results on elementary functions, Rep. RC 2402, IBM Corp., Yorktown Heights, NY, March 1969.
RISCH, R. (1970). Solution of the problem of integration in finite terms, *Bull. AMS*, Vol. 76, No. 3, pp. 605-608.
SLAGLE, J. (1961). A Heuristic Program that Solves Symbolic Problems in Freshman Calculus, Symbolic Automatic Integrator (SAINT), Doctoral Dissertation, MIT.
WANG, P. S. (1971). Evaluation of definite integrals by symbolic manipulation, Doctoral Dissertation, Department of Mathematics, MIT, Cambridge, Mass., Also MAC-TR-92, *Project MAC*, MIT.
WANG, P. S. (1974). The Undecidability of the Existence of Zeros of real elementary functions, *JACM*, Vol. 21, No. 4, pp. 586-589.

Medical Images: Formation, Perception and Measurement, proceedings of the Seventh L. H. Gray Conference held at the University of Leeds, 13-15 April 1976, edited by George A. Hay, 1977; 368 pages. (*The Institute of Physics and John Wiley*, £12·50)

It is always difficult to review conference proceedings when one has not been present at the conference. However, it should be said at once that this is a well produced book containing a collection of high quality papers. Only the statements about the papers' origin and the reproduction of some of the discussion discloses how they came to be written. The handling of complex images from a wide variety of sources has become very much part of the stock in trade of the medical physicist from both the academic and the service point of view. This has taken them into instrumentation technology, pattern recognition and computing. This book provides papers on various aspects of the present and future development of multifarious aspects of this complex subject and provides a valuable review of the state of the art—as one might expect from an L. H. Gray Conference.

The main sections in the book are concerned with recent advances in imaging devices, methods of assessment of instrument performance, the detection and perception of image formation, the quantitative assessment of image processing and the extraction of numerical information from images. The papers vary from ones dealing with instrumentation and experimental topics to those concerned with the mathematical analysis of system function. A wide variety of physical systems are included from ionography, computerised transaxial tomography, radioisotope imaging with gamma cameras, thermography, acoustic and ultrasonic imaging.

The papers individually are up to full academic standards in presentation, content and bibliography. The provision of the proceedings of such a standard and within a year of the original conference is a practice that one wishes would be more widely followed. This book will be a valuable addition to personal and departmental libraries.

B. BARBER (London)

Computer Science and Multiple-valued logic, edited by D. C. Rine, 1977; 548 pages. (*North Holland*, US$ 50.95)

Although multiple-valued logic (which admits more numerals than the 0/1 alternatives of the binary system) has held the interest of mathematicians since the 1920's, its relevance to computer science has had a rather bumpy history. In 1947 Norbert Wiener produced a brilliant argument for the optimality of the binary system for *storing* information. There have been other arguments (in my opinion bogus) for the optimality of a base of e which is most closely approximated by 3. However, when it comes to hardware for *processing* as opposed to storage it may be that multiple-valued systems offer interesting and possibly even economical alternatives.

However, the author claims that the recent revival of interest heralded by this book is partly underpinned by the idea that multiple-valued decisions are often made in programming (when one wants to branch with one or several 'maybe' alternatives). The editor structured the book by inviting 26 experts to write in four areas: algebraic theory, logic design, ternary logic, physical components and applications.

The first, theoretical, part collates various points of view applying the ideas first put forward by E. L. Post in 1921 and is clearly fertile ground for rigorously pursuing the properties of concepts such as 'implication' and 'completeness' in multiple-valued systems. The last paper in this section provides probably the most exhaustive bibliography on the subject in existence (464 citations). The second part runs through all those topics familiar to 'binary' switching theorists: minimisation, sequential machine design and computer aided design—all, I feel, somewhat uninteresting due to a certain lack of fundamentality. A brief section on ternary logic is followed by one on electronics, oriented part on rather ingenious forms of implementation.

Possibly the most interesting paper for the computing scientist (as distinct from the computer engineer) is a short nine page description of multi-valued logic in programming applications (part V of the book). The sheer conciseness of this contribution in a rather thick, expensive volume belies a little the author's claim that '. . . there is a good chance that the impact of multiple-valued software considerations will be at least as important as hardware considerations on the use of fourth and fifth generation computing and data processing systems . . .' However, putting the emphasis on the 'will be' in the above statement, the book could be recommended as background to those who would like to be involved in making the editor's prediction come true.

I. ALEKSANDER (Uxbridge