Analysing Mathematical Proofs (or Reading between the Lines)

ANALYSING MATHEMATICAL PROOFS
(OR READING BETWEEN THE LINES)

Alan Bundy
Department of Artificial Intelligence
University of Edinburgh
Edinburgh, Scotland.

Abstract

He study equation solving and analyse the solutions of experienced mathematicians. We find that traditional theorem proving methods are inadequate to explain the directness of these solutions, and that the well known algorithms for polynomials etc. are inadequate to explain the wide variety of equations solved. Our analysis reveals a system of high-level descriptions, strategies and goals, which can be used to guide the search through an explosively large search space. A few of these strategies will be investigated in detail. It is hoped that this analysis will eventually be incorporated into a computer program that solves equations.

Introduction

The problem domain we address ourselves to in this paper is equation solving. In fact we are Interested in finding real valued solutions to equations involving rational, trigonometric, inverse trigonometric, exponential and logarithmic functions. In this paper we concentrate on finding general solutions to single equations in a single unknown.

A study of this domain is overdue because although the experienced human mathematician finds equation solving straightforward, no existing computer program can match his performance. Traditional theorem proving programs are inadequate to explain the directness (lack of search) of his solutions. The well known algorithms for polynomials (e.g. Gaussian elimination, Sturm sequences, Newton-Bairstow, etc.) are inadequate to explain his range (e.g. trigonometric, logarithmic equations, etc.). See, for instance, Martin and Fateman, 1971 p68 or Moses 1974. Neither can his solutions be regarded as part of a process of putting the equations into some normal form or of reducing their syntactic complexity. So the standard simplification packages provided by systems like REDUCE (Hearn 1967 3.3) or MACSYMA (Martin and Fateman 1971 p64) are inadequate to explain the success of his solutions.

In order to try to match the human mathematician's performance we analyse his solutions. This analysis reveals a system of high level descriptions, strategies and goals, which can be used to guide the search for a solution through an explosively large search space. These high level concepts appear to be useful not only in equation solving, but throughout elementary algebra. We believe, the methodology should prove useful throughout mathematics.

In addition elementary equation solving appears to be an especially appropriate domain to start with because:
i) Many useful high level concepts are revealed in the language used by mathematicians when they discuss equation solving, e.g., elimination of variables, change of variable, collecting terms, isolation, trigonometric equations, etc.

ii) Equation solving and pattern matching (unification)are very similar. Since all algebraic manipulation is made easier by having a sophisticated pattern matching package, we make life easier by studying equation solving first.

In this paper we analyse some solutions, tease out some high level strategies and discuss how these might be incorporated in a computer program. The strategies we discover are:
isolation, collection, attraction and the basic method.

In the extended version of this paper, Bundy 1975, we show how these strategies can be used to derive some historically interesting general solutions. We go on to analyse some solutions which cannot be explained by the above strategies and discover further strategies.

The Proposed Program

At the time of writing not all these ideas have been implemented in a computer program. I have no doubt that they all could be, and a student at Edinburgh, Bob Welham, has experimented with implementing some of them in the PROLOG language (see Warren 1974). However, it would be premature to invest a lot of effort in a program before the whole domain has been explored. New connections between strategies are constantly being discovered, and new problems are constantly emerging, so that such programming effort might be wasted. Our current ideas about the form of the program are reproduced below:

The program will be organised around a conventional theorem prover, run in proof checker mode. That is, the theorem prover will not be able to do anything unless instructed to. The strategies are the procedures which will do the instruction. The word "strategy" was adopted to describe each of these procedures because they provide the theorem prover with search strategic guidance information. However, for some of them descriptions like "tactic", "method" or "algorithm" seem more appropriate. In such cases we will sometimes use the more appropriate description as a synonym for "strategy", albeit with connotations.

These strategies will examine the current equation, using high-level predicates,to decide what to do. On the basis of this examination they may call other strategies or apply individual axioms. They will also decide whether those calls were successful, and what to do next. The strategies will need to exploit all the benefits of a high-level programming language, e.g., recursion, conditionals and search. It would be convenient if this language and the theorem prover could share

facilities such as search and database mechanisms. This may be possible using PROIOG, but care must be exercised in the mixing of Metalanguage and Object language concepts. For instance,we might decide to have a predicate ISVAR which is true if and only if its argument is a variable. Then ISVAR (U) would be true and ISVAR (3) false. However, we can derive ISVAR (3) from ISVAR (U) leading to a contradiction. Our solution to this problem is to regard all expressions in the object language (constants, variables, terms and formulae) as constants in the metalanguage, so that we cannot deduce ISVAR (3) from ISVAR (U). For a further discussion of these issues see, for instance, Kleene 1962 pp62-65 and p298.

The strategies will have available to them a store of laws of real number theory. We will call these laws axioms because the program will assume them true without proof. The axioms which we are planning to put in this store are listed in an appendix to Bundy 1975. These axioms are highly redundant and include many facts which would normally be regarded as theorems in a parsimonious logical system. We will reserve the words theorem and lemma for facts provel at run time. Strategies may pick which axiom / apply next, in a variety of ways, according to t. context.

For instance:

i) A strategy may have the axioms useful to it earmarked at compile time.

ii) A strategy may have only a general description of the axioms useful to it and have to retrieve suitable axioms at run time. (This does not prevent some prior indexing).

ill) A suitable axiom may not be available, and the strategy may have to appeal to a theorem proving program to prove a lemma to use instead.

No suitable axioms may be found, and the strategy will have to fail. On the other hand several suitable axioms may be found and the strategy will have to conduct a short search.

No manipulation of the equation will be allowed except by the theorem prover applying an axiom to it. This idea of separating the inference system from the search strategy is due to Kowalski 1970 pl81. It has a number of advantages (see Hayes 1974, section 3). For instance:

i) Provided the theorem prover is sound, the correctness of any solution is guaranteed. This is sometimes in doubt in a more ad-hoc program.

II) All the axioms can be equally accessible to all the strategies, so a new axiom need only be added once.

ill) The learning of new axioms and new strategies is made easier.

A Logarithmic Example

Let us consider an equation which most people will find fairly easy to solve.

Solve for X: $\log_e(x+1) + \log_e(x-1) = 3$

To the best of my knowledge, there are no known equation solving algorithms which could solve this equation (because it is logarithmic). In addition, it defines a search space which although modest by equation solving standards would present a considerable challenge to most Resolution (see Robinson 1965) theorem provers.

The solution we are going to analyse is given below:

1. $\log_e(x+1) + \log_e(x-1) = 3$
   .....(i)
2. $\log_e(x+1).(x-1) = 3$
   .....(ii)
3. $\log_e(x^2-1) = 3$
   .....(iii)
4. $x^2-1 = e^3$
   .....(iv)
5. $x^2 = e^3+1$
   .....(v)
6. $x = \sqrt{e^3+1}$ or $x = \sqrt{e^3+1}$

Analysis will lead us to discover the strategies of attraction, collection, and isolation, which we will link up into a strategy called the basic method.

Notation

Since in what follows we shall be reading between the lines of this solution, it will be necessary to be able to refer to the spaces between the lines. We have, therefore, labelled these spaces with Roman numerals, (on the right of the solution). These spaces will be referred to as steps, e.g., step (ill) is the one from line 3 to line 4.

In problem solving, the word "solution" has two different meanings. It can mean either

i) The written protocol of the solver, as in the sentence "Hand in your solutions by Friday".

II) The answer to the problem, as in the sentence "The solution of the equation is x=6".

For instance in the above solution the written protocol is the whole thing from lines 1 to 6, the answer is just line 6. Since we will need to distinguish carefully between these two concepts we use the word "solution** to mean the written protocol and we will use the word "answer" to mean the answer.

We will also need to define what syntactic forms are acceptable as answers. In equation solving, answers give values for the unknown in terms of expressions not involving the unknown. So close is this to the concept of a substitution that we have decided to make the connection between pattern matching and equation-solving more explicit and use the word "substitution" to mean

those formula which are syntactically acceptable as answers.

As explained in Bundy 1974, the x in the above solution is not a variable in the predicate logic sense. If it were we would be allowed to substitute arbitary terms for it during the course of the solution and we would get solutions such as :

1. $\log_e(x+1) + \log_e(x-1) = 3$

2. $\log_e(1+1) + \log_e(1-1) = 3$

3. $\log_e 2 + \log_e 0 = 3$

etc.

In fact x actually behaves like a constant when we are searching for a general solution and like a variable when we are searching for a particular solution. In this paper we are concerned only with general solutions so we regard x as a constant. Followinq Polya 1962 we distinguish it from any other constants in thee equation by calling it the unknown. We reserve the letters

x,y,z	for unknowns
a,b,c,d	for other constants
u,v,w	for (genuine) variables (in the axioms)
s,t	for terms
f,g,h	for function sympols
A,B,C	for formulae
and E	for equations

The Analysis

In this analysis, because we look only at the final solution, we are guilty of the sin of not displaying the search involved in discovering that solution. We will try to atone for this sin when we discuss how to implement the strategies we discover Unfortunately it will not be possible, until the program is run, to be certain about the size of the space searched before the solution is found. However, in the conclusion we argue that this space will be a small function of the length of the solution.

We look first at the end of the solution, lines 3 to 6. In line 3, for the first time the equation contains only a single occurrence of the unknown, x. From here on the solution is straightforward. The next three steps consist of stripping away the functions surrounding this single occurrence of x until it is left isolated on the left hand side of the equation. Each step consists of identifying the outermost (or dominant) function symbol on the left-hand-side, recovering the axiom which will remove it from the left-hand-side and insert it's inverse on the right-hand-side, and then applying this axiom. We will call the strategy which guides these three steps isolation, because it isolates the single occurrence of x on the left-hand-side. We will explain it in more detail in the isolation section. It can be regarded as the basis of nearly all work in equation solving as well as much of the rest of algebraic manipulation As with simplification, mathematicians often omit isolation steps from their written protocols, crowding as many as three or four steps into the transition from one line to another.

We look next at line 2:

2. $\log_e(x+1).(x-1) = 3$

This contains 2 occurrences of x, so isolation is not applicable. However, we can see step (ii) as preparing for isolation, by achieving a reduction in the number of occurrences of x from 2 to 1. This is done by applying the identity.

$$(u+v).(u-v) = u^2-v^2$$

This is an example of our second strategy, which we call collection, namely, when there is more than one occurrence of the unknown, try to find an axiom which will collect occurrences together.

Finally we look at line 1 and step (i)

1. $\log_e(x+1) + \log_e(x-1) = 3$

The two occurrences of x cannot be immediately collected, presumably because a suitable axiom was not stored. We can however prepare the occurrences for collection by moving them closer together, so that more identities will match the term containing them both. This is what happens in step (i). The notion of "closer** is defined in the next section. The strategies of moving occurrences closer together to increase their chances of collection, we call attraction.

When the above three strategies are combined we will call the resulting equation solving strategy, the basic method.

Our analysis is now as below.

1. $\log_e(x+1)+\log_e(x-1) = 3$
   (i) attraction
2. $\log_e(x+1).(x-1) = 3$
   (ii) collection
3. $\log_e(x^2-1) = 3$
   (iii)
4. $x^2-1 = e^3$
   (iv) isolation
5. $x^2=e^3+1$
   (v)
6. $x = \sqrt{e^3+1}$ or $x=-\sqrt{e^3+1}$

the basic method

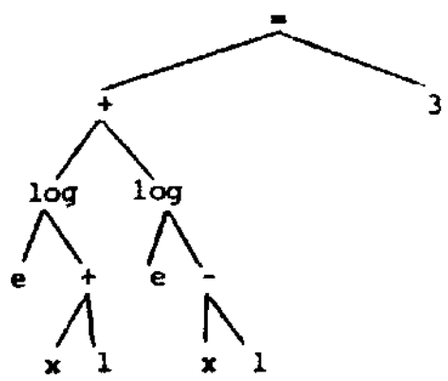
Definitions

In order to explain how we might implement these strategies, we will need to have some notation to discuss algebraic expressions, especially the notion of distance between occurrences of symbols.

We can represent any algebraic expression as a tree (we do not use a graph, since we want to be able to distinguish between occurrences):

$\log_e(x+1) + \log_e(x-1) = 3$

can be represented as



The distance between several occurrences of symbols in an expression is the size (number of links) of the smallest subtree in the tree which connects them all, e.g., the distance between the two occurrences of x in the example is 6 and the distance between the three integers is 8.

Any term or formula will be represented by a complete subtree. The symbol at the root of that subtree is called the dominant symbol of that term or formula:

log is the dominant symbol of $\log_e(x+1)$

\+ is the dominant symbol of x+1

An occurrence of an expression is said to contain another expression if the tree of the second is a subtree of the tree of the first:

log (x+1) contains x+1 and x.

An occurrence of a symbol is said to dominate an expression (ϕ say) if the symbol is the dominant

symbol of an expression which contains ϕ

log dominates x+1, e, and 1 in log (x+1)

Isolation

We now consider the four strategies in more detail in order to see how they might be implemented.

If we have an equation, E, containing a single occurrence of the unknown x then we can apply isolation. For the sake of argument suppose x occurs on the left-hand-side of E and that the dominant function symbol on the left-hand-side is the n-ary symbol f, i.e., E is of the form

$f(s_1 \ldots\ldots s_i(x) \ldots\ldots s_n) = t$

where s_i (say) is the only term containing x. The isolation process is to first isolate s_i, and then to call isolation recursively to complete the isolation of x. To isolate s_i we need an axiom of the form

$f(u_1 \ldots\ldots u_i \ldots\ldots u_n) = v \rightarrow A$

where A is some substitution for u_i in terms of the u_j, $j \neq i$ and v. A is often of the form

$u_i = f_i(u_1 \ldots\ldots v \ldots\ldots u_n)$ where f_i is an inverse of F.

But this is not always the case, as the axioms for sin and squared illustrate.

$u^2 = v \rightarrow (u = \sqrt{v} \ \vee \ u = -\sqrt{v})$

$\sin u = v \rightarrow (\exists n \in \text{integers})\ (u = n.\pi + (-1)^n.\arcsin v)$

We therefore define a substitution for x as follows

i) If S does not contain x, then x - s is a non-trivial substitution for x.

ii) If A and B are non-trivial substitutions for x then AvB is a non-trivial substitution for x

iii) If A(n) is a non-trivial substitution for x and S is a set of real numbers then $(\exists n \in S)$ (A(n)) is a non-trivial substitution for x.

iv) "True" and "False" are trivial substitutions for x.

v) A substitution for x is a trivial or non-trivial substitution for x.

With this definition of substitution there is an axiom of the required form associated with each of the functions of elementary algebra. We might speculate that this is not an accident, that mathematicians have deliberately defined new inverse functions, (e.g.,arcsin and square root) so that these axioms would exist. About these inverse functions little is known, and this causes problems when they occur in equations we are trying to solve. How these problems are tackled is discussed in Bundy 1975.

It should be a straightforward matter to write a procedure which can : decide whether or not there is a single occurrence of the unknown in an equation and on which side it occurs; identify the dominant symbol of that side; recover the appropriate axiom; apply it and then apply isolation recursively to the resulting equation or disjunction of equations. The only difficulty that might arise is if the axiom was stored in the wrong form:

e.g., the sin axiom as

$\sin u = \sin v \rightarrow (\exists n \in \text{integers})\ (u = n.\pi + (-1)^n.v)$

the right form would have to be recovered as a subgoal. We delay consideration of this problem.

Collection

Collection is the strategy which takes an equation and tries to reduce the number of occurrences of the unknown which it contains. For the sake of clarity we will limit our discussion initially to the case when two occurrences are reduced to one. This is the most common situation and, at the cost of some complication, our discussion can easily be extended to the more general case.

Having decided which two occurrences to collective

focus our attention on the smallest expression which contains them both. If the two occurrences are on the same side of the equation this will be a term, called the containing term, otherwise it will be the whole equation. We deal with the former case first. The containing term must now be replaced by another term containing one occurrence of the unknown, say x, so we must look for an axiom, say A, which will do this. We can easily build up a description which A must obey.

1) A must be an identity, i.e., an expression of the form:

   $s_1 = s_2$ or $B \rightarrow s_1 = s_2$, where B is called its precondition.

ii) One of the variables, say u, must occur either :
   (a) twice in s_1 and once in s_2
   or (b) twice in s_2 and once in s_1

   without loss of generality we will assume case (a),

iii) s_1 must match the containing term, with one of its variables, say u, being instantiated to x. Note that we are not allowing u to be matched to a proper term, say t(x), containing x. This situation will be handled by making a prior change of unknown of, say y, for t(x). We are also not allowing the situation where two variables, say u and v, are both matched to x. If A(u,v) is useful to collection by matching u and v to x then A(u,u) is also useful, and therefore non-trivial. We will assume that we will not need to do collection using A(u,v), because A(u,u) will also be present. To justify this assumption we need only make sure that all non-trivial axioms which can be gotten by identifying the variables of existing axioms are added to the axiom store.

If A obeys parts(i) and (iia) of the above description we will say that it is useful to collection, left to right, relative to u. If A obeys parts (i) and (iib) we will say that it is useful to collection, right to left, relative to u.

sin 2u - 2.sinu.cosu

is useful to collection, right to left, relative to u.

$(u+v).(u-v) = u^2-v^2$

is useful to collection, left to right, relative to u (and v)

$w.(u+v) = w.u + w.v$

is useful to collection, right to left, relative to w

$u+0 = u$

is useless to collection

To find a suitable axiom the collection strategy need only search among those axioms useful to it and find one which also obeys part iii) of the description.

The case when the two occurrences of x appear on opposite sides of the equation can be dealt with in a similar way. This time we are interested in axioms of the form:

(a) $B \rightarrow (C_1 \leftrightarrow C_2)$

or possibly

(b) $B \rightarrow (C_1 \rightarrow C_2)$

where C_1 (or C_2 in situation (a)) can be matched to the whole equation. We can build the appropriate definitions of "useful to collection" by replacing s_1 with C_1 and s_2 with C_2 in parts (ii) and (iii) of the previous descriptions.

We will want to extend these definitions to deal with any reduction in the number of occurrences of the unknown. Then we can include in collection the use of axioms like

u - u - o

to reduce 2 occurrences to none. The modification to the definitions is straightforward, (iia) becomes : u must occur more often in s than in s).

Even though strategies like collection guide the search for a solution, they do not eliminate search altogether. For instance, collection has to choose:

(a) which set of occurrences of the unknown to try to collect, (and therefore which subterm to try to replace)

(b) which matching, useful-to-collection axiom to apply.

Our initial experience is that collection narrows the search considerably. In fact we are lucky to find a single axiom, useful for collection, which matches one of the containing terms. In the event of a real choice we could use heuristics based on the fact that we would like to reduce the number of occurrences as much as possible.

Notice that we are demanding that collection do its job with one application of an axiom. This is because we want collection either to succeed or to fail quickly, so we can try something else. We could allow it to conduct a short search, but it would be difficult to know on what grounds to terminate an unsuccessful search. The problem with keeping such a tight rein on collection is that it might fail simply because the axiom which should have succeeded was not in the right form.

It would be a pity to fail to collect x in

(1) $\log_e(x+1).(x-1) = 3$

because the axiom were in the form

(2) $(u-v).(u+v) = u^2-v^2$

instead of

(3) $(u+v).(u-v) = u^2-v^2.$

Our solution to this problem is to have a pattern matcher which can recognise that axiom (2) matches (x+1).(x-1). This entails building-in axioms like the commutativity of multiplication (c.f. Plotkin 1972). One aspect of this building-in process is discussed in the section on pattern matching in Bundy 1975, but a complete discussion is delayed until a later paper. Such a pattern matcher will increase the number of axioms which will match the containing term. However, we still believe that collection will be lucky to find an axiom which both matches and is useful to it.

We will want the pattern matcher to spend a large, but bounded, amount of effort to try to make an axiom match an expression. If this is not to be computationally very time-consuming, collection had better only feed good candidate axioms to the pattern matcher. It can select good candidates by ;

(a) Only choosing those "useful to collection"

(b) Insisting that both the axiom and the expression it is to match obey the same high level description e.g., both are quadratic, trigonometric, logarithmic, etc.

Attraction

The attraction strategy is very similar to collection. As before we must choose the occurrences we are going to attract, then focus on their containing term. As before we will use a definition of axioms useful to attraction to recover a suitable axiom to apply. The main difference is in the type of axioms useful to attraction.

An axiom is useful to attraction, left to right, relative to some subset of its variables, u say, if for all ueu, u has the same number of occurrences on left and right, but the distance between all the occurrences is strictly smaller on the right than the left. We can make a similar definition for useful to attraction, right to left.

e.g., $\log_e u + \log_e v = \log_e u.v$

is useful to attraction, left to right, for{u,v}

$w.(u+v) = w.u + w.v$

is useful to attraction, right to left, for{u,v}

$w^{u.v} = (w^u)^v$

is useful to attraction, right to left, for{u,v}

The variables in the subset u will each be bound to terms containing the unknown, x.

The Basic Method

Isolation, collection and attraction can be combined into a basic method for solving equations.

If the equation contains a single occurrence of the unknown we isolate this occurrence. Otherwise we try collection on each combination of occurrences until either the number of occurrences is reduced to one, or no more collections are possible. In the first case we call isolation, in the second case we call attraction. We try attraction on one combination of occurrences. If this is successful we call collection, otherwise we try attraction on some other combination. If we run out of attractions to attempt, the method fails.

Conclusion

We have studied elementary equation solving. Existing equation-solving, computer programs, do not match either the range or the directness of the human mathematician. We have analysed such mathematician's solutions and discovered a system of high-level strategies. These strategies are not restricted in their range of application. They guide the search for a solution by

i) Providing a series of intermediate subgoals between the present equation and the final solution.

ii) Focusing on the subterm of the equation which is to be manipulated next.

iii) Selecting a small set of axioms which can be nade to apply to that subterm and which are relevant to achieving the next subgoal.

We claim that this guidance reduces the size of the space searched considerably and that sometimes there is no search at all.

We have discovered and investigated the strategies of isolation, collection, attraction, and the basic method. This is only a small part of the story. There are many other strategies at work in equation solving. There are general-purpose strategies like simplification, pattern-matching, change of unknown, and elimination, and there are special-purpose strategies for well understood types of equations like polynomial and trigonometric equations. We delay discussion of these to another paper.

Acknowledgements

We would like to thank the many people with whom we have interacted over this work for their encouragement, suggestions and inspiration. It is difficult to separate the contributions of each individual. Odd remarks and suggestions have often fermented together, resulting in a product unrecognisable to the original contributors. However, we single out for special mention : Soei Tien Tan, Gordon Plotkin, Ira Goldstein, Joel Moses, Luis Pereira, Bob Welham, Alan Robinson, Woody Bledsoe, Bob Kowalski and George Luger. We apologise to the many other people we have omitted.

References

1. Bundy, A. (1974) "A Treatise on Elementary Equation Solving". Internal Memo, Department of Artificial Intelligence, Edinburgh.

2. Bundy, A. (1975) "Analyzing Mathematical Proofs (or reading between the lines)". Department of Artificial Intelligence Research Memo No. 2, Edinburgh. (An extended version of the present paper).

3. Hayes, P.J. (1974) "Some Problems and Non-Problems in Representation Theory". Paper presented at the A.I.S.B. Summer Conference at Sussex.

4. Hearn, A.C. (1967) "REDUCE : A User Oriented Interactive System for Algebraic Simplification". In Interactive Systems for Experimental Applied Mathematics. Academie Press, New York, pp 79-90,

5. Kleene, *S.C.* (1962) "Introduction to Metamathematics". North Holland.

6. Kowalski, R. (1970) "Search Strategies in Theorem Proving". In Machine Intelligence 5, eds. B. Meltzer and D. Michie. Edinburgh University Press, pp 181-201.

7. Martin, W.A. and Fateman, R.J. (1971) The MACSYMA System. In Proceedings of the 2nd Symposium on Symbolic Manipulation, ed. S.R. Petrick,Los Angeles, pp 59-75.

e. Moses, J. (1974) Private Communication.

9. Plotkin, G. (1972) "Building in Equational Theories". In Machine Intelligence 7, eds. B. Meltzer and D. Michie. Edinburgh University Press, pp 73-90.

10. Polya, G. (1962) "Mathematical Discovery", Vol. I and II. John Wiley and Sons, Inc.

11. Robinson, J.A. (1965) "A Machine Oriented Logic Based on the Resolution Principle". In J.A.CM., Vol. 12, No. 1, pp 23-41.

12. Warren, D. (1974) "Epilog [400,400) - A Users Guide to the D.E.C. 10 PROLOG System**. Internal Memo. Department of Artificial Intelligence, Edinburgh.