META-LEVEL INFERENCE IN ALGEBRA

Alan Bundy
Leon Sterling

D.A.I. RESEARCH PAPER NO. 164

Submitted to the Workshop on Logic Programming for Intelligent Systems, Los Angeles.

16th September 1981.

META-LEVEL INFERENCE IN ALGEBRA

by
Alan Bundy
Leon Sterling

Abstract

We describe two uses of meta-level inference: to control the search for a proof; and to derive new control information, and illustrate them in the domain of algebraic equation solving. The derivation of control information is the main focus of the paper. It involves the proving of theorems in the Meta-Theory of Algebra. These proofs are guided by meta-meta-level inference. We are developing a meta-meta-language to describe formulae, and proof plans, and have built a program, IMPRESS, which uses these plans to build a proof. IMPRESS will form part of a self improving algebra system.

Acknowledgements

This research was supported by SRC grant GR/B/29252.

1. Introduction

In [Bundy and Welham 81] we described the notion of meta-level inference, and used it as a technique for controlling search. We also speculated about how it might be used as a technique for deriving new control information. In this paper we describe work in progress to explore this second application of meta-level inference.

The domain used to illustrate these ideas is equation solving. In [Bundy and Welham 81] the PROLOG program, PRESS,* is described. PRESS successfully solves a wide range of equations of the difficulty of A-level mathematics papers. It uses meta-level inference to carefully guide the application of sets of rewrite rules. Without such guidance it would become hopelessly bogged down in a combinatorial explosion. PRESS clauses can be interpreted as axioms and theorems of the Meta-Theory of Algebra. Inference, in this mathematical theory, has the side effect of applying rewrite rules to algebraic expressions, and, hence, solving equations in the object-level domain of Algebra.

We are currently building a PROLOG program, IMPRESS,** which proves properties of PRESS procedures by deriving theorems in the Meta-Theory of Algebra. The same technique of meta-level inference is being used to guide the search. For IMPRESS the object-level entities are PRESS clauses, and the meta-level entities are methods for proving properties of them. The PROLOG clauses which implement these IMPRESS methods can be regarded as axioms and theorems of the Meta-Meta-Theory of Algebra.

IMPRESS will form part of a larger program designed to make PRESS into a learning program. Such a program might begin by being shown examples and non-examples of rules satisfying the syntactic criteria of a particular method.

---

*PRolog Equation Solving Ssytem.

**Inferring Meta-knowledge about PRESS.

It would also be shown examples and non-examples of when this method should be used, from which it would produce a PRESS clause to control the use of the method. We have experimented with a version of Winston's arch learning program for this task, [Winston 75, Young et al 77]. IMPRESS would then try to establish properties of this method, thereby demonstrating that the correct information had been learnt, and perhaps pointing to exceptions not covered by the examples. Finally the new method would be incorporated into PRESS.

The technique of meta-level inference is perhaps best illustrated on an example. As a running example, throughout this paper, we will use the equation-solving method of Isolation. A simplified axiomatization of Isolation will illustrate guided search when solving equations; while the correctness of Isolation, as a theorem in the Meta-Theory of Algebra, will illustrate the new control information that IMPRESS can derive.

2. A Method for Solving Equations

Isolation is a method for solving equations containing only a single occurrence of an unknown. That is, Isolation can solve

$$\log_{\sin x^2} \cos a = \tan a \text{ for } x$$

but not

$$e^{\sin x} + e^{\cos x} = 5 \text{ for } x$$

since the latter contains two occurrences of x. Isolation is a key method because many solutions work by reducing equations to a form in which there is only one occurrence of the unknown, and then applying Isolation.

The method consists of 'stripping off' the functions surrounding the single occurrence of x by applying the inverse function to both sides of the equation. This process is repeated until x is isolated on one side of the equation, e.g.

$$\log_{\sin x^2} \cos a = \tan a$$

$$\sin x^2 = (\cos a)^{1/\tan a}$$

$$x^2 = 180n + (-1)^n \arcsin (\cos a)^{1/\tan a}$$

$$x = \sqrt{180n + (-1)^n \arcsin (\cos a)^{1/\tan a}} \text{ or}$$
$$x = -\sqrt{180n + (-1)^n \arcsin (\cos a)^{1/\tan a}}$$

This stripping off is done by applying a system of rewrite rules to equation, in this case the rules:

$$\log_u v=w \rightarrow u=v^{1/w}$$

$$\sin u = v \rightarrow u = 180n + (-1)^n \arcsin(v)$$

$$u^2=v \rightarrow u = \sqrt{v} \text{ or } u = -\sqrt{v}$$

All rules used by Isolation have the same general form, namely

$$P \;\&\; F(U1,\ldots,Ui,\ldots,Un)=V \rightarrow Rhs$$

where: Rhs is usually of the form Ui=Fi(U1,...V,...,Un), but can also be a disjunction of such formulae; Fi is the ith inverse function of F; and P is an optional condition.

3. The Meta-Theory of Algebra

Isolation is implemented in PRESS by a series of clauses defining the ternary predicate, isolate, where

isolate(Posn,Lhs=Rhs,X=Ans)

means that expression X=Ans* is the result of isolating the subterm at position Posn in expression Lhs. The position of a subterm in an expression is a list of numbers which specifies the arguments it lies within (see figure 3-1).

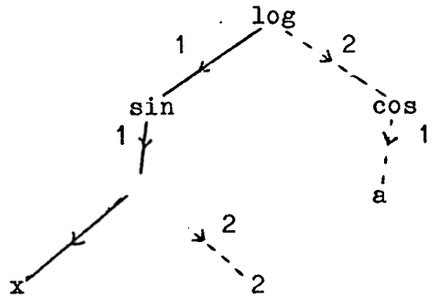

Figure 3-1: The Position of x in $\log_{\sin x^2} \cos a$

Interpreted declaratively, PRESS clauses can be regarded as axioms or theorems in the Meta-Theory of Algebra. For example, the clause which invokes the Isolation method can be interpreted as the theorem:

single-occ(X,Lhs=Rhs) & position(X,Lhs,Posn) &
isolate(Posn,Lhs=Rhs,X=Ans) -> solve(Lhs=Rhs,X,X=Ans) (ii)

where:

single-occ(X,Exp) means, X occurs precisely once in Exp;

position(X,Exp,Posn) means, X occurs at Posn in Exp;

- solve(Eqn,X,Soln) means, Soln solves Eqn for X.

As a theorem of the Meta-Theory of Algebra, (ii) asserts that when an equation contains only a single occurrence of X then Isolation is guaranteed to solve it. We will call this theorem, The Correctness of Isolation and will use it as an example of a program property that IMPRESS can prove.

Some typical axioms of the Meta-Theory of Algebra are the definitions of solve and isolate:

---

*Throughout the paper we use the PROLOG convention that words beginning with capital letters denote variables. In the Meta-Theory of Algebra, algebraic (object-level) variables are represented as (meta-level) constants. Thus v is a meta-level constant representing an object-level variable, whereas V is a meta-level variable ranging over object-level variables. Note that this avoids the use/mention ambiguity, because we cannot apply meta-level substitution to object-level variables.

```
solve(Eqn,X,X=Ans) <-> equiv(Eqn,X=Ans) & free-of(X,Ans)

isolate([],Soln,Soln)

isolate([Car|Cdr],Old,Soln)
        <-> ∃New [isolax(Car,Old,New) & isolate(Cdr,New,Soln)]
```

where:

equiv(Form1,Form2) means, Form1 and Form2 are equivalent algebraic formulae;

free-of(X,Exp) means, Exp contains no occurrences of X;

- isolax(N,Lhs,Rhs) means, Lhs->Rhs is an Isolation rule which isolates the Nth argument position of Lhs.

The definition of solve is derived form a PRESS clause, but would be available to IMPRESS. The definition of isolate is a cleaned up and simplified version of the PRESS clauses which define the Isolation method. The simplifications are that we have decided to ignore Isolation rules with disjunctive right hand sides and those with conditions, and to assume that the single occurrence of the unknown is always on the left hand side of the equation.

IMPRESS can prove (ii) from the above axioms and theorems among others. Currently this is the only theorem the program can prove, but another PRESS method, Collection, is being examined.

4. Using Meta-Level Inference to Control Search

We now show how the clauses of the last section are used, by PRESS, to solve equations. Suppose we wanted to solve (i) above for x. This would be presented to PRESS as the goal:

```
solve( log(sin(x^2),cos(a))=tan(a), x, Soln ) ->
```

This would be resolved against (ii) to produce the subgoals:

```
single-occ( x,  log(sin(x^2),cos(a))=tan(a) ) &
position( x,  log(sin(x^2),cos(a)), Posn ) &
isolate( Posn,  log(sin(x^2),cos(a))=tan(a), Soln ) ->
```

After the first two subgoals had been resolved away using the definitions of single-occ and position, and binding Posn to [1,1,1], the last subgoal would be called. This would resolve with the recursive definition of isolate to produce the subgoals:

```
isolax( 1,  log(sin(x^2),cos(a))=tan(a), New ) &
isolate( [ ,1], New, Soln ) ->
```

The first of these subgoals would resolve with only one of the pre-stored Isolation rules, namely

```
isolax( 1, log(U,V)=W, U=V^(1/W) )
```

binding New to sin(x^2)=cos(a)^(1/tan(a)). This latest meta-level resolution implicitly applies a rewrite rule to an algebraic expression, thus making a move in the object-level search space. The proving of the remaining isolate subgoal will cause two more such moves to be made, before the meta-level

inference terminates, having caused the solving of the equation as a side effect.

5. Using Meta-Level Inference to Derive Control Information

Following our experience with PRESS, we have adopted the technique of meta-level inference to guide the search of IMPRESS to a proof of (ii), but in this case we have a process of meta-meta-level inference guiding a meta-level search space. The Meta-Meta-Theory of Algebra deals with the representation of the Meta-Theory of Algebra as predicate calculus formulae.

In order to guide the proof the meta-meta-theory makes crucial distinctions between the parts of (ii). The main parts are the hypothesis, isolate(Posn,Lhs=Rhs,X=Ans), and the conclusion, solve(Lhs=Rhs,X,X=Ans). The remaining parts, single-occ(X,Lhs=Rhs) & position(X,Lhs,Posn), form the conditions. We expect these distinctions to be supplied by the Winston-type concept learning program as a consequence of the task of learning the conditions under which the hypothesis part could be used to derive the conclusion part.

To form a strategy to prove the theorem, IMPRESS looks at the hypothesis, isolate(Posn,Lhs=Rhs,X=Ans). This is a relation, recursively defined on the first argument, Posn: So IMPRESS decides to try a proof by induction on this argument. This reasoning parallels that of the Boyer/Moore theorem prover [Boyer & Moore 79], when proving properties of LISP functions.*

IMPRESS forms two subgoals: the basis and the step of the induction, by substituting [] and [n|posn], respectively, for Posn in negated and skolemized versions of (ii)

Basis:

```
single-occ(x,lhs=rhs)
position(x,lhs,[])
isolate([],lhs=rhs,x=ans)
solve(lhs=rhs,x,x=ans) ->
```

Step:

```
single-occ(x,lhs=rhs)
position(x,lhs,[n|posn])
isolate([n|posn],lhs=rhs,x=ans
solve(lhs=rhs,x,x=ans) ->
```

The basis subgoal is easily proved, since the condition position(x,lhs,[]) implies that lhs is x, and together with single-occ(x,lhs=rhs), that rhs is free of x. Hence lhs=rhs is already a solution for x.

To prove the step, (ii) with posn for Posn, is asserted as the induction hypothesis. That is, the new, temporary axioms are:

---

*In fact, in order to make comparisons, we have built and experimented with a toy version of the Boyer/Moore theorem prover, written in PROLOG. Both this, and our not so toy version of Winston's program, have proved easy and quick to implement in PROLOG: the implementations being both small and fast.

Induction Hypothesis:
```
single-occ(X,Lhs=Rhs) & position(X,Lhs,posn) &
        isolate(posn,Lhs=Rhs,X=Ans) -> solve(Lhs=Rhs,X,X=Ans)
```

Step Conditions:
```
single-occ(x,lhs=rhs)
position(x,lhs,[n|posn])
```

Step Hypothesis:
```
isolate([n|posn],lhs=rhs,x=ans)
```

Step Conclusion:
```
solve(lhs=rhs,x,x=ans) ->
```

In order that the induction hypothesis can be used, IMPRESS first unpacks the step conclusion into:

```
solve(Eqn,x,x=ans) & equiv(lhs=rhs,Eqn) ->
```

This requires the lemma:

```
solve(New,X,Soln) & equiv(Old,New) -> solve(Old,X,Soln)
```

Which IMPRESS proves using the definition of solve (twice) and the transitivity of equiv. The new 'solve' subgoal is then resolved with the induction hypothesis to get:

```
single-occ(x,Eqn) & position(x,Eqn,posn) &
        isolate(posn,Eqn,x=ans) & equiv(lhs=rhs,Eqn) ->          (iii)
```

Note that the step hypothesis, together with the recursive definition of isolate, can generate two consequences:

Performant: `isolax(n,lhs=rhs,new)`

Recursant: `isolate(posn,new,x=ans)`

The recursant can be used to resolve away the isolate subgoal of (iii), binding Eqn to new. IMPRESS expects to prove each of the remaining three subgoals from the performant, with the aid of the corresponding step condition in the case of the condition type subgoals. It conjectures the lemmas it would need to do this. These are:

```
isolax(N,Lhs=Rhs,New) & single-occ(X,Lhs=Rhs)
        -> single-occ(X,New)

isolax(N,Lhs=Rhs,New) & position(X,Lhs=Rhs,[N|Posn])
        -> position(X,New,Posn)

isolax(N,Lhs=Rhs,New) -> equiv(Lhs=Rhs,New)
```

In each case these conjectures are theorems of the Meta-Theory of Algebra. In order to allow the proof to continue we have supplied them as lemmas to IMPRESS. Thus IMPRESS completes the proof.

The above example illustrates the (meta-meta-level) language we have developed for describing parts of formulae, and how they can be used to guide a proof search. This parallels the (meta-level) language that we developed to describe algebraic expressions, and used to guide equation solving. Such a

language is the essence of the meta-level inference search-control technique. We have introduced the concepts of: condition, hypothesis and conclusion; basis and step; and performant and recursant. We have explained: how an induction schema and variable are chosen using the recursive definition of the theorem hypothesis; how the step goal is transformed to allow the induction hypothesis to be applied; and how, after the induction hypothesis has been applied, the remaining subgoals are proved by using either the performant or the recursant of the recursive definition of the theorem hypothesis.

Thus the language enables a proof plan to be built which then supervises the search for a proof. This proof plan is so explicit that it can specify the precise form of the lemma required to enact a particular part of the proof. Of course, all this has been induced from a single example. But it was not induced in an ad hoc way. General properties of recursive definitions, inductive proofs and mathematical theorems, were borne in mind, and this gives us a naive faith in its generality, borne out by continuing work. Our next step is to apply IMPRESS to a wide range of similar theorems, and hence to extend and modify it.

6. Conclusion

We have described two applications of meta-level inference: to control inference; and to derive new control information. We have illustrated these applications in the area of algebraic equation solving. The derivation of new control information is part of a wider goal of building a self improving algebraic manipulation program.

New control information can be derived by proving theorems in the Meta-Theory of Algebra. To control the search for such proofs we are developing a meta-meta-level language, and expressing proof plans in this language. A program, IMPRESS, has been built which runs on one example, and is in the process of being extended to further examples.

REFERENCES

[Boyer & Moore 79]
Boyer, R.S. and Moore, J.S.
ACM monograph series. : A Computational Logic.
Academic Press, 1979.

[Bundy and Welham 81]
Bundy, A. and Welham, B.
Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulation.
Artificial Intelligence 16(2), 1981.

[Winston 75]
Winston, P.
Learning structural descriptions from examples.
McGraw Hill, 1975, .

[Young et al 77]
Young, R.M., Plotkin, G.D. and Linz, R.F.
Analysis of an extended concept-learning task.
In Reddy, R., editor, IJCAI-77, pages p285. International Joint Conference on Artificial Intelligence, 1977.