LEARNING EQUATION SOLVING METHODS FROM EXAMPLES

Bernard Silver

Department of Artificial Intelligence
University of Edinburgh
Edinburgh EH8 9NW
U.K.

ABSTRACT

This paper describes LP*, a program that learns new techniques for solving equations by examining worked examples. Unlike most of the work in this field, e.g. (Neves, 1978), where the equations used have been very simple, LP uses complex equations, see below. LP can learn from one example, using concepts from the planning field.

In order to be able to successfully use a new technique, LP learns many different types of information. To learn new rewrite rules, LP compares consecutive lines in the worked example, finding differences between them. It also learns the strategic purpose of the steps, by considering the worked example as a type of plan for solving the equation. LP extracts the necessary information, and builds a plan which is stored for future use. LP executes the plan in a flexible way to solve new equations.

I. INTRODUCTION

This paper describes LP, a program which learns new techniques for solving symbolic equations by examining worked examples. These techniques can then be tested on some new problems. LP can learn from one trial by using concepts from planning.

The equations used by LP are symbolic, transcendental, non-differential equations, e.g.

$$\sin(x) + 2.\sin(2.x) + \sin(3.x) = 0.$$

LP is built around the equation solving program PRESS, (Bundy and Welham, 1981, Sterling et al, 1982).

Worked Examples

A typical instance of the type of worked example used by LP is shown below.

$$\sin(x) - 2.\sin(4.x) + \sin(7.x) = 0 \qquad \text{(i)}$$

$$2.\sin(4.x).\cos(3.x) - 2.\sin(4.x) = 0 \qquad \text{(ii)}$$

$$2.\sin(4.x).(\cos(3.x) - 1) = 0 \qquad \text{(iii)}$$

$$\sin(4.x) = 0 \quad \text{or} \quad \cos(3.x) - 1 = 0$$

$$\sin(4.x) = 0$$

$$x = 45.n1$$

$$\cos(3.x) - 1 = 0$$

$$x = 120.n2$$

The key step is the transformation from equation (i) to equation (ii).

II. METHODS AID OPERATORS

The basic operators of LP are called methods. Each method has an associated set of rewrite rules, and some control information which indicates when the method should be applied.

Equations are solved by applying methods, which in turn may apply rewrite rules. An example of a method is Collection, which applies to equations containing more than one occurrence of the unknown. Collection reduces the number of occurrences of the unknown. Collection is used in the example above to obtain line (iii) from line (ii), using the Collection rewrite rule

$$U.V + U.W \Rightarrow U.(V + W).$$

The control information includes preconditions, facts which must be true before the method can be applied and postconditions, which must be true after the application of the method. The postconditions are used both to ensure that the method has achieved the desired effects, and to allow planning.

Many learning programs work in domains where the basic operators are similar to those of STRIPS. The operators of STRIPS, (Fikes et al, 1972), have preconditions, an add list which contains the facts that the operator makes true, and the delete list which contains the facts that are no longer true after the application of the operator. If the preconditions are satisfied the operator can be applied.

*This work is supported by S.E.R.C. grant GR/C/20826, and a S.E.R.C. studentship to the author.

In contrast, LP methods do not have this desirable property. In general, a method is not certain to succeed, even if the preconditions are applicable. This is because the preconditions are too general, but we can not give stronger preconditions that do not involve actually applying the method to test if it is applicable! It seems that this might be a problem in many domains.

Similarly, the effects of a method are hard to classify. The postconditions are used specify what should be true after a method has been applied in the desired way, but there is no guarantee that the method will produce these effects.

III. WHAT LP WEEDS TO LEARN

In order to learn a "new technique", LP may need to learn at several levels. At the lowest level, it may need to learn new algebraic identities. These will be used as rewrite rules. We are not particularly concerned with this low level task. At the next level up, LP needs to learn new operators, i.e. methods. This will involve learning the control information for the method, and associating with it some rewrite rules. These rewrite rules may be new or old rules. The control information does not depend on which equation is being used.

However, LP needs also to learn the meta-control information, which controls the order in which methods are used. This information is recorded in a plan, called a schema. This records how the equation is solved, and can be used to solve new equations. The plans may be equation dependent. For example, the plan may record that the solution involved applying method M followed by method N. In the general case however, even if method M can be applied, it may not be the case that method N will then be applicable.

IV. LEASHING FROM EXAMPLES

Step Justification

The first task for LP is to discover how each line in the worked example is transformed into the next, we call this Step Justification. To do this, it examines consecutive pairs of lines, trying to find the method that transforms each line to the next.

Suppose that LP is working on the step from a line p to the next line, called q. LP first tries to see if an existing method can account for the step. To do this, LP computes the characteristic tuple, henceforth CT, for each of the lines p and q. *The* CT is a tuple of meta-level characteristics of the equation, consisting of facts such as the number of occurrences of the unknown, the type of function symbols occurring in the equations, e.g. trigonometric, whether the equation is a single equation or a disjunction etc. For example, the CT of

```
sin(x) - 2.sin(4.x) + sin(7.x) - 0
```

is

```
[occ(multiple,3) , trig, eq, functor(+ , 2)],
```

meaning that there are multiple occurrences of the unknown, 3 in all. The expression is trigonometric, it is an equation (rather than a disjunction), and the dominating functor on the Lhs is "♦" of arity 2.

LP then looks for a method that can transform the CT of p into the CT of q.* Use of the CT to constrain search is an example of the technique of meta-level inference which is used extensively in the PRESS and LP projects.

If LP finds a method, it attempts to use it to transform p into q (a method may not apply even if its CT and other control information indicate that it is suitable). If the method is successful, LP records that the step from p to q was performed by that method, and proceeds to the step from q to the next line. Otherwise, it tries to find another possible method. If no more possible methods can be found, LP conjectures a rewrite rule that would explain the step.

The rewrite rule is obtained by removing common terms from p and q and equating the remaining terms. For example, consider the lines (i) and (ii) in the above example. Both lines contain the additive term -2.sin(4.x) and the right hand side of both is 0. Deleting these and equating the remainder produces the conjecture

```
sin(x) + sin(7.x) = 2.sin(4.x).cos(3-x).
```

If the conjecture is correct the user is asked to provide the general rule. In this case, the user gives LP the rule

```
sin(A) + sin(B) ->
    2.sin((A + B)/2).cos((A - B)/2).          (iv)
```

LP avoids generalizing at this level by asking the user for the general rule. Given that LP is learning from one trial, there seems to be no easy way for the program to correctly generalize rules of this sophistication. Unlike Neves, (Neves, 1978), it is no longer sufficient just to replace numbers with variables.

Creating new methods

Once every step has been processed, LP examines its analysis to see if any new methods need to be created. New methods are created to explain the application of new rules. Suppose a new rule has been applied at line i, to produce the next line j. LP first finds the preconditions P of the

•This information is part of the control information of the methods.

method M applied at line j to give line k. It then finds which of these preconditions are satisfied at line i, call this set S. The remaining preconditions are not satisfied at line i, but satisfied at j, call this set U. If U is non-empty, LP assumes that the purpose of applying the rule is to satisfy U so that M can be applied. The set U is called the major effectB of the rule.

The above analysis is performed for each application of a new rule in the worked example. If no suitable method exists, see below, LP creates a new one. This method applies the rule and then method M. The preconditions of the method are S, plus any preconditions of the rule. The postconditions of the method are those of M.

In other cases, LP will find that it already has a method with the preconditions S, and the postconditions of method M. In this case, LP adds the rule to the set of rules that can be used by that method. In this way, methods gradually build up larger sets of associated rewrite rules.

If LP finds that all the preconditions of M were satisfied, i.e. U is empty, it looks for another explanation. One possibility is that the rule is used to manipulate the equation so that M can be applied, although no new preconditions are satisfied. This kind of behaviour occurs because the methods are not STRIPS type operators.

The schema is now created. This is a list consisting of all the methods used in the worked example. Each step is tagged with the conditions that it is used to satisfy, i.e. the major effects, plus any conditions that must also be maintained.

V. SOLVING NEW EQUATIONS

Schema are used to solve new equations. When LP is given an equation it first tries to find a schema that seems to be relevant. A schema is relevant if the equation that produced it has the same CT as the current equation. If one is found, LP tries to apply the steps listed in the schema. Suppose that the current line in the schema suggests the application of method M. LP uses the following procedure:

1. Try to apply method M. If this succeeds, continue with the next step in the schema.
2. Otherwise, try to find another method that has the same major effects as M, and apply it. If this succeeds, continue with the next line in the schema.
3. If the two steps above both fail, try to find a method that does not undo already satisfied conditions, and apply it. If this succeeds, go to 1.
4. If none of the above steps have succeeded, and if M has no major effects, omit the step entirely, and proceed with the next schema step.

If none of these attempts are successful, LP tries to solve the equation without the schema, in a similar way to PRESS.

The schema acts as a simple plan that can be executed in a flexible way.

VI. RESULTS AND CONCLUSIONS

LP has learned several new techniques from worked examples. One result is that LP has been able to solve difficult equations after examining much easier examples. For instance, after LP has been given worked examples for the equations such as

$$\cos(4.x) + \cos(6.x) = 0 \qquad (v)$$

it is able to solve

$$\sin(2.x) + \sin(3.x) + \sin(5»x) = 0. \qquad (vi)$$

The worked example for equation (v) contains 7 lines. The solution for equation (vi) contains 14 major steps, and includes a variant of equation (v) as a subproblem.

The techniques used by LP seem to be applicable to many domains. For example, symbolic integration is performed by transforming integrals using a sequence of operators, and this process seems similar to that of equation solving. It seems that LP could learn integration techniques from worked examples, using our planning style approach.

Using concepts from planning, the technique of learning from examples seems to work well in the algebra domain, even when the equations are much harder than those considered by earlier workers.

REFERENCES

Bundy, A. and Welham, B. Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulation. Artificial Intelligence, 1981, I^Of), pp189-212.

Fikes, R.E., Hart, P.E. and Nilsson, N.J. Learning and executing generalized robot plans. Artificial Intelligence, 1972, J>f 251-288.

Neves, D.M. A^ computer program that learns algebraic procedures by examining examples and working problems in _a textbook, pages 191 —1 95• Canadian Society for Computational studies of Intelligence, 1978.

Sterling, L., Bundy, A., Bjyrd, L., O'Keefe, R., and Silver, B. Solving Symbolic Equations with PRESS, pages 109-116. Springer Verlag, 1982.