DEPARTMENT OF ARTIFICIAL INTELLIGENCE

UNIVERSITY OF EDINBURGH

DAI Working Paper No. 127

Date: October 1982

Title: LADYBUG : A 'BUGGY-TYPE' PROGRAM FOR THE LINEAR ALGEBRA DOMAIN

Author: Kenneth Y Goldberg

Introduction

The goals of Artificial Intelligence can be divided into two main areas: to design machines to do tasks otherwise done by human beings, and to investigate into the nature of intelligent functions. Often, as in the case of computer assisted instruction (CAI), these areas overlap.

Developing a tool always involves studying the task. If the tool is to be used for welding or guiding a missile, then its development will, on the whole, remain in the concrete realm. But if the tool is to be used for educating human minds, then one must study the task of learning. How do we acquire knowledge? How does the mind develop? What is the nature of this development? What, in other words, is Thought? Needless to say, these avenues can lead one away from the welding shop, the battlefield, or even the classroom, into the realms of philosophical speculation.

The LADYBUG Project falls under the subject of computer assisted instruction, being concerned with the teaching of linear algebra. The goal was not to build a machine to teach algebra to students, but rather a machine to teach future teachers about the learning process. This involved writing a computer program to simulate the learning student -- not only the ideal student, but also the real student, who often learns only after making mistakes.

The project was based on the idea that students ~~by using~~ perform algebra with a systematic series (protocol) of procedures, even if their results are incorrect and appear to be random errors. If the protocols that lead to incorrect answers can be identified and reconstructed, then, for many cases, we will be able to reproduce the behavior the the student. In this report, all protocols which lead to <u>incorrect</u> answers will be referred to as "mal-rules", and the theory which holds that these are an accurate model of the learning mind as the "mal-rule theory".

This report will investigate some theories of Learning and Educational Psychology which provide the basis for the mal-rule theory. Two existing programs which address this theory, BUGGY and LMS, will then be described. Armed with this background, we will examine the LADYBUG program, its development and algorithms. If we assume that the mal-rule theory is valid, then we can confine the discussion to the best way to implement the mal-rule model given the constraints of the programming language. This 'engineering' work has in fact been the major focus of effort in the LADYBUG Project. But I think it would be a mistake not to make an attempt, in conclusion, to assess the validity of the task and raise some questions as to the wider implications of the mal-rule theory.

Learning Theories and the Roots of the Mal-Rule Theory

Jean Piaget was one of the first psychologists to probe beneath the surface of childhood learning. What he found can be divided into two areas for the purposes of this report: views on performance, that is the way in which humans perform tasks such as problem-solving, and views on learning, or how humans aquire the ability to perform. For Piaget these areas were inseparable; as Seymour Papert says in <u>Mindstorms</u>, Piaget felt that one could not separate the process from the structure of what is being learned [Papert].

In the area of performance, Piaget embraced the notion of the <u>schema</u>, first put forth by the neurologist, H. Head, which was defined as 'an active organization of past experiences' [Stones] which form an intellectual structure. Piaget felt that every cognitive act was controlled by a schema; much of his work was devoted to qualititively and quantitatively describing these schema.

Piaget sau schemata as being characterized by discrete substeps which are grouped together into a sequential and cohesive unit. For instance, when a baby wants to shake a rattle, he must first reach out his hand, curl his fingers around the handle, and then move his hand back ad forth. As the child grows these steps become more and more internalized until they resemble a single action. It should be possible to break down seemingly complex tasks into a series of pure procedures, which can be used like bricks to rebuild the task-solving process. Furthermore, Piaget thought fundamental the idea of <u>transfer</u>, whereby the bricks used to build one process could be combined with other bricks to build a new process for a previously unfamiliar task.

Here is where learning comes in. A child is faced with a task he has never seen before. Piaget found that the child's responses to this situation 'will reflect a melange of <u>organized</u> but inappropriate earlier structures and the halting and sporadic use of as yet incompletely organized neu Gradually, the new structure will crystallize into a 'tightly-knit, organized, and stable whole' [Flavell].

Thus we can view learning as the building of new structures using previously-existing bricks. An important corollary to this idea is that errors, or incorrect responses, are no longer considered random behavior unrelated to rationality, but are now seen as resulting from the same bricks that uill ultimately form the correct response. With this perspective on performance, it becomes tempting to analyze students' errors for the misplaced bricks.

The Russian psychologist L. S. Vigotsky did some experiments on perceptual thinking in the 1930s which involved just such an analysis. He constructed wooden blocks varying in size, shape, color, and height. He chose the variables such that the blocks could be grouped in different ways, for instance some blue blocks may be triangles, but not all the triangles were blue. He then asked the children to group the blocks. [Vigotsky's experiment was much more complex, involving nonsense syllables and language concepts; I'm only describing what is relevant to our discussion].

Careful study of several hundred subjects led Vigotsky to identify what he called the <u>pseudo-concept</u>. 'The pseudo-concept resembles the adult concept; for example the child might select all the triangles to form a group. He does not, however, make his selection on the basis of triangularity but upon the concrete visible likeness rather like the associative complex'[Stones, 157]. What he is saying is that the child's responses do seem to be governed by an intellectual structure; a structure, houever, which is inappropriate to the question asked.

In short, students' errors don't come from a chaotic pile of carelessly strewn bricks, but from a carefully constructed stack. Unfortunately, this stack may contain one or more wrong bricks or may be missing a feu crucial ones. It is the job of the conscientious teacher to study this stucture and help to repair it.

What does all this have to do with algebra and the mal-rule theory? Everything. What the mal-rule theory postulates is that most algebr errors do not result from random behavior but from highly organized schema which produce results consistent with their structure but not with the external standard. The <u>mal-rule</u> is like Vigotsky's pseudo-concept. It resembles the correct rule, but is somehow not quite perfect. What is important is that the mal-rule is still a rule, in that it will be consistently applied again and again unless externally corrected (external in that some feedback is required, even for self-correction).

To cement the idea I would like to make an analogy with how one learns to hit a golf ball. The golf pro demonstrates, and then the student makes an attempt. The golf pro watches the performance, comparing each phase, the backswing, contact, and follow-through, with his acquired knowledge of common errors (mal-rules). He may notice that the golfer is crooking his left elbow, or forgetting to shift his weight on the follow-through. He will patiently correct these as he identifies them, until the swing becomes closer and closer to the ideal.

To return to the LADYBUG Project, one characteristic of a good golf instructor is his familiarity with the common errors and ability to spot them. Accordingly, a good algebra teacher should be familiar with the common algebra errors. But identifying the appropriate error is often more difficult with algebra because the teacher only sees the end result on the test paper. The intermediate steps are not explicit. This is like asking the golf pro to tell you what's wrong with your swing just by watching where the ball lands!

But of course algebra errors are usually far less subtle than their counterparts in golf. With training, an algebra teacher should be able to recognize many common errors.

Training a teacher recognize mal-rules requires having some knowledge of what the particular mal-rules are. This should be done empirically, by analyzing real students. Some work has been done lately by Sleeman and by Alan Bundy in the area of inductively diagnosing mal-rules from the erroneous answers.

LADYBUG, however, was concerned with implementing some previously-identified mal-rules [see Appendix]. The forerunner in this kind of attempt was a program written in 1978 by J. S. Brown and Richard Burton, which they called BUGGY.

BUGGY

BUGGY gets its name from its authors' name for mal-rules, which they call 'bugs', in much the same way as computer programmers refer to faults in a program as 'bugs' [Brown, 1978]. BUGGY began as an attempt to build a deep-structure model of a student learning how to subtract two numbers. The authors studied data collected from 1300 Nicaraguan children and identified 60 possible bugs. They also recognized that many more bugs could result from combinations of these 60 'primitive' bugs acting together. With the help of a computer, they found that 40% of the students followed consistent mal-rules.

The next step was to build a model that would replicate many of the more common mal-rules. They introduced the term diagnostic model 'to mean a representation of a student's procedural knowledge or skill that depicts his internalization of a skill as a variant of a correct version of that skill' [got that?]. Basically, what they meant is that the mal-rules should be closely related to the correct rules. A minor change to the set of correct rules should be sufficient to produce mal-rule behavior. Why? Because this seems to be the most humanlike representation.

One of the main tasks of Artificial Intelligence is to learn about how human beings think. The study of this question falls under Cognitive Science. Much effort goes into structuring computer programs in a way which we hope might resemble the way a human mind is structured. Programs with this goal are judged as to whether or not they are 'psychologically valid'.

Many criteria can be applied, for instance the relative times of performance of hard and easy problems should compare with human responses. Or, as in the case with BUGGY and LADYBUG, mal-rule behavior should be replicated from a structure similar to the correct structure. That is, a change of one or two lines in the correct program should be sufficient to produce mal-rule errors.

Brown and Burton used a procedural network to represent the task of subtracting two numbers. A procedural network is a collection of independent operations linked to a control structure. Mal-rules can be replicated by variations to a procedure or to the control structure.

The format of the BUGGY 'game' was as follows. BUGGY randomly selects a mal-rule, not revealing it to the user. A sample problem with erroneous answer is shown to the user, who plays the role of a detective trying to discover the mal-rule. BUGGY then asks the user to offer some problems, which BUGGY duly solves using the mal-rule. When the user thinks he has discovered the pattern, he is asked to describe the bug in English (BUGGY simply stores this data; the authors have found that these descriptions often demonstrate that the user cannot adequately explain a mal-rule even when he understands how it works). Then BUGGY offers some problems and asks the user to solve them using the mal-rule. If the chosen mal-rule is verified, BUGGY provides a standard description and moves on to a new mal-rule. Otherwise, BUGGY tells the user to provide more diagnostic problems.

When tested with student teachers in the Boston Area, it was shown to significantly improve their ability to diagnose subtraction errors. More importantly, 'the realization that errors that appear 'random' are often the surface manifestations of a systematic underlying bug is a major conceptual breakthrough for many student teachers'[Brown, p.167].

D.H. Sleeman applied the mal-rule theory to simple algebra problems in 1981 at Leeds University. His Leeds Modelling System (LMS) is less concerned with teaching teachers than identifying the mal-rules and reducing the search space of combinations. He is presently working toward developing a test that could isolate and identify most common mal-rules with a minimum of test problems. I am indebted to his articles for providing five of the six mal-rules implemented in LADYBUG [see Appendix].

Choice of Domain

First of all, what is linear algebra? For those frightened by "higher mathematics": relax. Linear algebra is the most basic form of algebra. Sample problems look like this:

$$3*x - 5 = 4$$

$$4*x + 6 = 12 + 2*x$$

It's called "linear" because it can be used to describe straight lines on a Cartesian graph. Almost all schoolchildren are taught how to solve these simple equations when they are about 12 years old.

I chose the domain of linear algebra for several reasons. First, because it is mathematical, it can be described very precisely: certainly an advantage when dealing with a computer. Second, I restricted the domain to linear algebra so that answers could be expressed simply, without the need for square-roots or logarithms. Lastly, being aware that work had recently been done in the Edinburgh AI Department on a symbolic manipulation program, PRESS, I figured that I might be able to apply some of its results to LADYBUG.

Definition of Goals

After reading some literature and evaluating my capabilities, I decided to narrow my goals to three: 1) to write a psychologically valid program for correctly solving linear algebra problems, 2) to implement several mal-rules suggested by Sleeman (examples to follow), and 3) to interface the modeller such that it could be used by someone unfamiliar with computers.

DEVELOPMENT

Programming Considerations

In order to make a mal-rule theory modeller psychologically valid, one must examine the theory. It holds that students trying to learn algebra follow consistent rules which lead them to an answer, though the answer may be wrong. This assumes that algebra can be broken in Piagetian substeps (rules). Furthermore, it requires that these rules be capable of being independently added, removed, or modified. Thus what was needed was a computer program which broke the task into simple, easily modified, substeps.

Another aspect of algebra is that it usually involves performing one particular rule, such as moving an integer to the right-hand-side, repeatedly. Thus some element of recursivity was desireable in the programming language. Lastly, I needed a language that was available on the Edinburgh system.

This language, not surprisingly, was PROLOG, a recursive list processing language. A 'program' in PROLOG is a set of relationships between variables. These relationships can take the form of <u>rewrite rules</u>. A rewrite rule is an ordered pair of expressions, whereby one expression is used as a template for comparison and the other as a template for transformation. For example, the rule of arithmetic identity would appear as follows:

```
rewrite(X + 0 = Y , X = Y).
```

It means the same as:

```
X + 0 = Y  ->  X = Y
```

What this amounts to is a symbolic representation of algebra. It views the process from a meta-level where all constituents are considered as block-like <u>symbols</u> to be manipulated. The 'variable', x, is treated as a constant at the meta level. This symbolic representation allows us to express relationships between mathematical formulae. [Bundy, Artificial Mathematicians, p. 147].

It was found that all linear algebra rules could be expressed with rewrite rules. As a result, it was not necessary to consider other techniques (bags), which I wanted to avoid in favor of simplicity and clarity.

Choice of Algorithm

With thoughts toward implementation, I chose to implement the algorithm which I myself use in solving linear algebra equations. Dr.

Bundy had identified three major steps in the solving process: Attraction, Collection (consolidation), and Isolation.

Attraction: 2*x - 5 = 3 -> 2*x = 3 + 5

Consolidation: 2*x = 3 + 5 -> 2*x = 8

Isolation: 2*x = 8 -> x = 4

The order of the steps is important; I found that if each step were exhaustively applied, then all problems could be solved with three steps in the above order.

Implementation

From this point on, things may begin to get a bit technical; those unfamiliar with PROLOG may want to skip this section.

Each of Bundy's steps was to be implemented as a PROLOG predicate. To do this, all patterns which are applicable to a predicate must be made to match at least one clause of that predicate. For the ATTRACT predicate, I found that 18 unique patterns were possible in attracting integers to the left-hand-side (LHS) of the equation and variable terms to the RHS. In order to make the code as understandable as possible, I wrote 18 explicit clauses such as the ones below.

X + I = Y -> X = Y - I

I + X = Y -> X = Y - I

X1 - I + X2 = Y -> X1 + X2 = Y + I

As with all procedural networks, the program operates on several levels, each controlling the level directly below. In the case of CENTIPEDE, the SOLVE predicate was used to direct the process from ATTRACT to CONSOLIDATE to ISOLATE. Supervision of each predicate's clauses is handled by PROLOG, which recursively exhausts each predicate before returning to the control level.

Evaluation of Method 1

The resultant program, CENTIPEDE, required 18 ATTRACT clauses, 7 CONSOLIDATE clauses, and two ISOLATE clauses. The code was easy to read and the program admirably solved almost all linear algebra problems correctly. It wasn't until I considered implementing mal-rules that I realized the drawbacks to this method. Any simple alteration to the predicates, such as making ATTRACT 'forget' to change signs as it moved quantities acrosss the '=' sign, required that all 18 clauses be manually altered! This was certainly not the direction to go, even though it was recognized that highly specific mal-rules, such as 'forgetting' to change signs on integers sandwiched between variable terms on the LHS, could be conveniently implemented. I decided that such specific mal-rules would be rarely found, and that it would be inordinately difficult for the teacher-user to discover them.

Also, it was necessary to ask whether or not such a method was psychologically valid. Does the student learn 27 separate rules in order to perform this task, or does he learn three more general rules? The latter seemed more likely, so I began to consider ways of condensing the program into fewer, more general clauses.

CULMINATION

The LADYBUG Algorithm

Salvation came from advisors Bundy and Sterling, when they suggested a technique they had used in PRESS. This involved a MATCH predicate that would parse the required elements from the equation. MATCH could be shared by several predicates, thereby greatly reducing redundancy in the program. When implemented, MATCH reduced ATTRACT from 18 to only 4 clauses. The new ATTRACT clauses are of the form:

LHS = RHS and match(LHS, I, REST) -> REST = RHS - I

Furthermore, MATCH was more versatile than the 18 old predicates. LADYBUG was able to solve all sample problems.

Ceneralization of the CONSOLIDATE predicate

Upon studying various mal-rules and some outside literature [Sleeman] I decided to generalize the CONSOLIDATE predicate such that it would add all integers on a side, ignoring x's if they appear. If the problem were being solved properly, this would not lead to errors, as the LHS would be supplied with '*x' at a higher level.

The rationale behind this decision was threefold. One, using the same predicate for both LHS and RHS would reduce the number of predicates required; Second, this predicate would embody a simple rule: 'Now, add all the numbers on a side' which seemed appropriate to the environment; Third, and most importantly, this rule could be applied to an equation even if the attraction step were omitted. In fact, omission of the attraction step, I discovered, led to a very interesting mal-rule: Simply adding all numbers on each side and then 'forcing' the result to look like the standard by supplying an x to the LHS. This mal-rule was subsequently implemented by simply omitting ATTRACT from the sequence of operations.

Interfacing LADYBUG for Student Teachers

Last, an interface was built for LADYBUG which involved placing all mal-rules into one file, each as a distinct sequence of the existing predicates. This interface also included a tracing provision which could display all steps explicitly if asked. Two further additions involved sample problems and two predicates to 'switch' the tracer on/off and to choose a mal-rule.

CONCLUSION

What can be gained from this effort? The goal of serious research is to contribute in some way to humanity's body of knowledge. If we apply that goal to the LADYBUG Project, then she was less than a smash success. But if the goal was a more personal one, to provide a small-scale example of AI research, to learn what such research entails and how it might be implemented, then the LADYBUG soared.

A large part of the project involved research into Educational Psychology, a discipline I had never before encountered. I searched arduously for information on mal-rules from this neck of the woods but was unable to find anything explicit beyond Vigotsky's pseudo-concepts. It seems that most researchers tacitly assume the concept of mal-rules when they discuss behavior. I would still like to find some information dealing directly with the mal-rule concept.

As for the choice of project, I would not hesitate to choose it again. I am not aware that any field testing has been done on a BUGGY-type algebra program but am confident that something positive would result from such an experiment. The potential for future work on this kind of project is enormous.

In the course of my research I met with Dr. Margaret Donaldson, who strongly encouraged me to go to the source, the classroom, for direction. It has evidently been her experience that the scholar cloistered away in the ivy tower is unlikely to understand what goes on in the child's mind.

I think that this advice certainly applies to the LADYBUG Project. I built my model from introspection, and I now realize that the methods used by someone who has used algebra for many years are quite different from those of the neophyte. For instance, I now realize that the method taught to schoolchildren involves applying identical rules to both sides of an equation, always stressing the equality of both sides; and gradually isolating the variable. For instance, rather than offering the rule:

$$X + I = Y \quad \rightarrow \quad X = Y - I$$

the teacher will break it into the following steps:

$$X + I = Y \quad \rightarrow \quad X + I\ (-\ I) = Y\ (-\ I)$$
$$X + I\ (-I) = Y\ (-\ I) \rightarrow X + 0 = Y - I$$
$$X + 0 = Y - I \quad \rightarrow \quad X = Y - I$$

I think that the latter method will prove more meaningful and will be retained longer by the student. Maybe a future LADYBUG-type of program could be built to model this method of problem-solving.

I found that experienced teachers could offer a wealth of knowledge about students' protocols. One psychology student showed me a mal-rule I had never thought of:

$$3x = 36 \quad \rightarrow \quad x = 6$$

Another showed me

$$x - 7 = 12 \quad \rightarrow \quad x = 12/-7$$

Both of these have the added attraction of being discovered empirically from real children.

As for the program itself and the 'engineering' efforts of trying to build this model on the 2972 using PROLOG, I am for the most part pleased with the results. The rewrite rule have been successful for the mal-rules implemented, although I am beginning to think that the added flexibility of the 'bags' predicate may prove desireable. As Leon Sterling pointed out, the best approach may be to write a new string parser which would allow access to all elements of the equation and not be limited by the tree structure of PROLOG. For example, PROLOG is not able to parse '3X' as two separate atoms.

If I had more time, I would like to test LADYBUG out on some student teachers. The interface could stand some polishing, and it might be nice to build in some delay loops to mimic a pause as the 'child' works out the answer.

One area unexplored by LADYBUG was combinations of errors. This has been a major focus of effort by Brown, Burton and D.H. Sleeman, and deserves investigation. In fact, the existing program, LADYBUG, is well-suited to such models.

Post Script

The LADYBUG Project can, and should, be judged from more than one level. Beginning with the end, there is how well the program works. Back from there we can examine how well it is structured. Does the program elegantly make use of the programming language? The next step, does the program accurately represent the mal-rule theory? Could the program be of benefit to student teachers? Does the mal-rule theory accurately represent student behavior? Does the mal-rule theory accurately represent student thought? Do human beings think in terms of rules?

Although my work was confined to considering only the first few questions, it is the final few which are most important. I will not cloud this report with my own speculation on such questions as it is merely that -- speculation. But I would like to close this document by merely offering another question.

The idea that our minds can, in fact, operate in a manner alalogous to a machine is chilling. We find that many errors which seam to be random are caused not by a lack of obedience to the rules, but by slavish obedience to the wrong rules. Can it be that our minds are comprised of rules we are unable to disobey?

BIBLIOGRAPHY

Brown, J.S. and R.R. Burton, "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills" Cognitive Science 2.2 (1978) p. 155, 192.

Bundy, Alan, Artificial Mathematiicians [in press]

Cotton, J.W., Byrd, L., Bundy, A., "How can algebra steps be learned by students with only arithmetic skills" January, 1981 [in press].

Donaldson, Margaret, Children's Minds, (Glasgow:Collins and Sons, 1978).

Flavell, J.H., The Developmental Psychology of Jean Piaget, (Van Nostrand: Princeton, 1963).

Papert, Seymour, Mindstorms(Harvester Press: Brighton, 1980).

Pavell .R., Rothstein M., Fitch J., "Computer Algebra", Scientific American, December 1981.

Sleeman, D.H., "A Rule Directed Modelling System" Machine Learning (1982).

Sleeman, D.H., and M.J. Smith, "Modelling Student's Problem Solving" Artificial Intelligence, 1981, p.171-187.

Stevens, A. and A. Collins and S.E. Goldin, "Misconceptions is Student's Understanding", International Journal of Man-Machine Studies 11 (1979).

Stones, E., An Introduction to Educational Psychology, (Methuen and Company: London, 1966).