Causal Analysis and Inductive Learning

JOHN R. ANDERSON (ANDERSON@A.PSY.CMU.EDU)

Department of Psychology, Carnegie-Mellon University, Pittsburgh, PA 15213 U.S.A.

Abstract

PUPS learning mechanisms center around the representation of causal relationships between objects or events. These learning mechanisms involve (1) a set of principles for inducing causal relationships in absence of a domain theory; (2) a set of analogical mechanisms for extrapolating the causal analysis of one situation in order to predict or problem solve in another situation; and (3) compilation procedures for turning these extrapolations into general production rules. These learning mechanisms are illustrated with respect to a detailed example from the algebra tutor where they do a good job of reproducing the instructionless learning we observe of students.

1. Introduction

Inductive learning is concerned with the extraction of general principles from examples. It is one of the major goals of research in machine learning. The various methods that have been proposed have been successful in situations where there was a regularity in the domain that was congruent with the assumptions of the inductive method. There has been some uneasiness with the almost magic-like character by which a method just happens to capture a domain regularity. Presumably this dissatisfaction is part of what motivated the work in explanation-based learning (Mitchell, Keller, & Kedar-Cabelli, 1986; DeJong & Mooney, 1986). In explanation-based learning an axiomatization of the domain is provided which explains the regularities. This axiomatization is used to direct the derivation of general principles. However, now learning loses its inductive-like character. The products of the learning process are really just theorems given the axiomatization. The phrase 'knowledge compilation' (e.g., Anderson, 1983) has been used to describe this learning and to contrast it with inductive learning. Inductive learning in an explanation-based learning paradigm would amount to acquiring the axioms.

Humans manage inductive learning in absence of a prior axiomatization. On the other hand their learning is seldom of the syntactic variety that characterizes most machine learning approaches to inductive learning. Despite the lack of a prior axiomatization it is directed by an attempt to understand the problem domain. We have been working on a theory of learning called PUPS (Anderson & Thompson, 1986) which attempts to capture these properties of human learning and which we think promises to be much more general and robust than most machine learning methods. A key difference in this theory and my earlier theories of human learning (Anderson, 1983) is the process of causal induction by which PUPS infers that one thing in the environment caused another. These causal inferences provide something on the order of an axiomatization of domain from which PUPS can then analogically extrapolate to

provide predictions about new experiences. If successful these extrapolations will be compiled into rules for later use. The three major steps in the evolution of general rules from experience, then, are causal inference, analogical extrapolation, and compilation.

Twelve pages is much too little to give a careful development of this theory (especially as I sit here in Adelaide isolated from my machines trying to get it on my IBM convertible computer and printer). In lieu of that I will provide a brief discussion of each of these three aspects and will conclude with an extended illustration of their application to learning from our algebra tutor (Lewis, Milson, & Anderson, in press).

2. Causal Induction

People have an almost irresistable urge, when they encounter an event or object, to ascribe a cause to it where that cause is some other event or object. The majority of our adult causal ascriptions flow from theories we have already acquired, but people are quite capable of making these ascriptions in the absence of a domain theory. It is from these pre-theoretical inferences that domain theories eventually arise. Note I am making no claims in this paper about what causality is like in the real world or indeed if there is such a thing as causality. I am only asserting that people naturally perceive causality whether it is really there or not. The beauty of human causal perception is that it ignores the philosophical dilemmas about causality and so produces knowledge.

There are at least three well-documented factors inducing people to perceive one thing as causing another in the absence of an existing theory (Lewis, 1986; Shultz, 1982). Each of these can be fairly easily justified as a rational basis for making causal ascriptions:

1. Contiguity — People tend to perceive something as the cause the closer in time and space it is to the effect with the strong discontinuous provision that effects cannot precede their causes.

2. Similarity — People tend to perceive something as a cause the more similar it is to the effect. It is difficult to specify an all-encomposing metric for similarity, but for our purposes the important feature is that two things are more similar if they overlap in a number of components. For instance, suppose we observe two events involving an unknown computer system—the user points to an icon of an apple and he points to an icon of a dog. After both these events a third event happens—the icon of the apple disappears. We are more likely to think the cause of the third event is the pointing to the apple icon than the dog icon. This is because both cause and effect involve the apple icon.

3. Statistical — If a cause has been accompanied fairly regularly by an effect and the effect has seldom occurred in the absence of the cause, we are likely to perceive a causal relationship. Note perception of causality does not demand a perfect predictive relationship. There can always be extenuating circumstances.

It is something of an open question just how these three factors should be computed and combined to produce an attribution of a causal relationship. Psychological research can take the form of creating somewhat artificial situations to test for refined predictions of one scheme versus another. However, typically casual attributions are highly overdetermined. For instance, suppose we have no knowledge of computers and that we see (+ 2 3) typed into the computer and see 5 as a response. We would decide that the typing caused the 5 on the basis of contiguity, similarity (5 is the sum of 2 and 3), or statistical trials noting the regularity of the relationship.

The following illustrates the schema-like notation we have developed for representing knowledge in PUPS. We have used it to encode the (+ 2 3) example:

```
example: isa typing
      form: (list + two three)
      function: (cause event)
            precondition: (example context CommonLISP)
      context: CommonLISP

event: isa response
      form: (text five)
      function: (caused-by example)

three: isa integer
      form: (text 3)
      function: (third fact)

two: isa integer
      form: (text 2)
      function: (first fact)

five: isa integer
      form: (text 5)
      function: (fifth fact)

fact: isa addition-fact
      form: (sequence two plus three is five)
```

The form slot is used to record the physical form of the object or event while the function slot is used to encode causal and positional information. Thus, the function slots for example and event encode their causal relationship, while the function slots for two, three, and five encode their position in the addition fact $2 + 3 = 5$. There can be multiple functions associated with a PUPS knowledge structure. Preconditions can be attached to functions. Thus example has the precondition that it produces event in the context of CommonLISP. Context is just another slot associated with the example knowledge structure.

3. Knowledge Extrapolation

Knowledge extrapolation involves trying to extend a causal analysis to a new situation. Suppose for instance, one wanted to predict what would happen when (+ 3 5) was typed into CommonLISP. Analogical extrapolation allows one to map the past example onto the current example provided the categories (isa slots) of the objects are the same. In effect, we can extract from the example the following rule:

```
IF =structure: isa typing
        form: (list + =num1 =num2)
        context: CommonLISP
   =num1: isa integer
        function: (first =fact)
   =num2: isa integer
        function: (third =fact)
   =fact: isa addition-fact
        form: (sequence =num1 plus =num2 is =num3)
   =num3: isa integer
THEN =structure function: (cause =event)
       =event: isa response
            form: (text =num3)
```

The rule above is a production rule which predicts the function of =structure given its form. We can also create problem-solving productions in which the form necessary to achieve a function is specified:

```
IF goal: isa typing
        function: (cause =event)
        context: CommonLISP
   =event: isa response
        form: (text =num3)
   =num3: isa integer
        function: (fifth =fact)
   =fact: isa addition-fact
        form: (sequence =num1 plus =num2 is =num3)
   =num1: isa integer
   =num2: isa integer
THEN goal form: (list + =num1 =num2)
```

These analogical extrapolations depend on two basic assumptions. First, all members of a category behave identically with respect to their causal properties. Secondly, terms that bear the same functional relationships behave identically with respect to their causal properties. If I had room to unpack the evolution of categorical and functional structures in PUPS I could motivate the assertion that both of these extrapolations are reasonable inductive leaps although either could be in error in any particular case. We call these safe extrapolations.

In some problem-solving situations there may be no safe extrapolations of one's knowledge which will enable one to solve the problem. Then problem solvers may choose to push extrapolations beyond these safe bounds. This can occur in at least three ways. First, the problem-solver might try to apply the knowledge despite violated preconditions. For instance, the problem-solver might try (+ 2 3) in contexts other than CommonLISP. Second, the problem-solver might ignore category restrictions. Thus, despite the fact that (+ 2 3) is encoded as restricted to integers, he might

try it with reals. The third possibility is that, if the problem-solver has no example that can be extrapolated to solving the whole problem, he might try extrapolating an example that will solve part of the problem. An example we have observed is when we ask beginning LISP programmers to "Multiply X by the sum of Y and Z". Not knowing how to embed one operation within another they will type (+ Y Z) and then think about how they will achieve the requested multiplication.

The effect of such forced extrapolations is to try out the bounds of one's knowledge and discover new knowledge in the process. Thus, one might discover that the + operator is not restricted to integers.

Two components of this forced extrapolation throw problem-solvers into the large search spaces AI is typically associated with. First, students may subgoal satisfying preconditions such as finding a CommonLISP system. Second, there is the process of searching through different knowledge structures looking for one that will succeed. Managing search of such problem spaces is a major issue of control which shall also be ignored here.

4. Knowledge Compilation

The processes of causal inference and knowledge extrapolation produce general rules that can be used for prediction and problem solving. Those rules that prove successful get permanently recorded as production rules. In the domains of problem solving we have been studying (LISP programming, geometry theorem proving, algebraic symbol manipulation), there is a marked speedup after the first successful application of a piece of knowledge. The next application typically takes half as long. In our view this reflects the greater efficiency of rule recognition relative to knowledge extrapolation. Once a rule has been codified as a production it can accrue strength as it proves repeatedly successful. There is a gradual continued improvement in speed of rule application with practice.

This process of codifying successful rules we call knowledge compilation. It really does not change the behavioral potentials of the system, abstractly defined. For instance, an extracted rule can take precedence over a compiled rule if it offers a better fit to the problem situation. The only effect of knowledge compilation is to reduce the resource costs of manifesting knowledge. Of course, by changing the resource costs, one can change the effective behavior of a system even if not the abstract behavioral potentials. Paths of problem-solution which were too consuming of time or working memory now become feasible.

5. An Extended Example: The Algebra Tutor

For an example of these learning mechanisms actually producing learning in a significant knowledge acquisition situation, I would like to turn to an example from our algebra tutor. Learning with the algebra tutor is chosen rather than free-form learning because the algebra tutor provides a very well-defined situation and one for

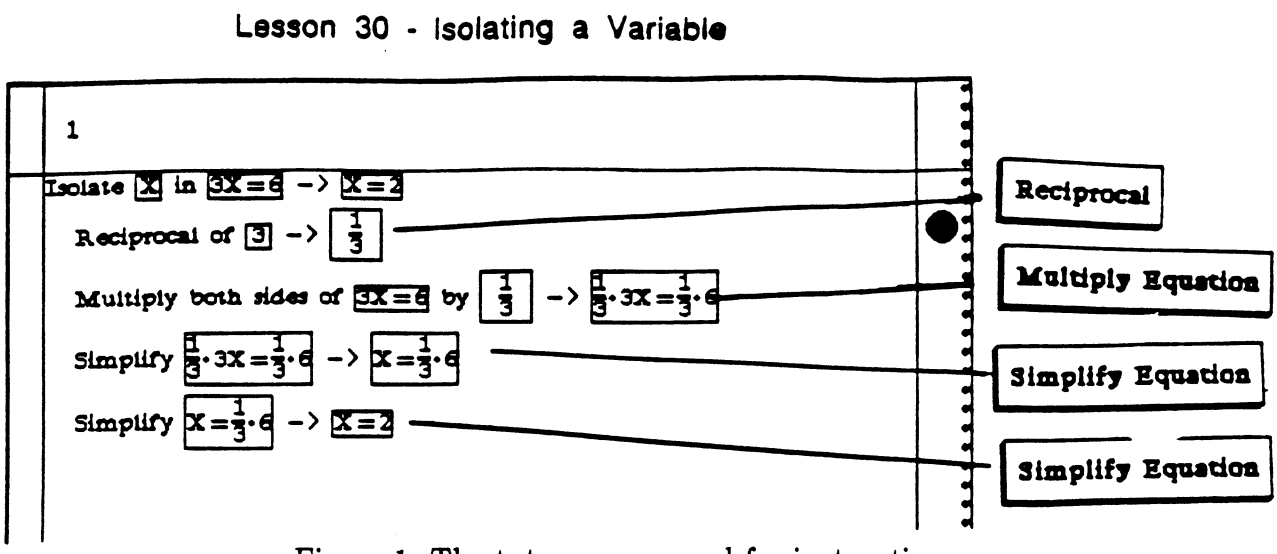

Figure 1. The tutor screen used for instruction

which we have ample empirical data. One of our observations of students behavior with the algebra tutor is that they tend to ignore instruction and work from examples. The same observation has been made of the learning of algebra in the classroom (Neves, 1978).

Figure 1 reproduces a screen image of an example used to tutor our students on the 30th unit of our algebra curriculum which concerns isolation of variables. The problem is initially presented to the student as "Isolation X is 3X=6 -> ???". If the student can write the resulting equation he can put that in the result box (the "???") and go onto the next problem. If not, however, the problem is broken down into a number of substeps which is what is illustrated in Figure 1. Each substep corresponds to an operation which the student has learned to do in a previous lesson.

Performing a substep amounts to selecting the operator (e.g., "reciprocal" from a menu, mousing in the arguments to the operator, and mousing in the result). After producing the result to the final substep the student can move that result to the result box for the original problem.

In our analysis of this example we will assume that the student knows the basic facts of arithmetic and has mastered the previous 29 lessons and so knows how to produce answers for each of the substeps in Figure 1. We will focus on how the student understands the relationships among the lines in Figure 1, extrapolates this knowledge to solving new problems, and codifies rules to represent the lessons of these extrapolations. In particular, we want to show how he extracts a rule that will directly produce an answer to the goal of isolating a variable. This rule will be in the same format as we are assuming for the rules acquired from previous lessons for doing the substeps.

Responding to length constraints (and good sense), I will restrict this discussion to the goal line, the first substep, and the last substep.

5.1 The Goal Line

The following is the PUPS encoding of the goal line in Figure 1:

top-goal: isa tutor-line
form: (line isolate term1 in exp1 -> exp2)

exp1: isa expression
form: (list term2 term3 = term4)

exp2: isa expression
form: (list term5 = term6)

term1: isa variable
form: (text X)

term2: isa number
form: (text 3)

term3: isa variable
form: (text X)

term4: isa number
form: (text 6)

term5: isa variable
form: (text X)

term6: isa number
form: (text 2)

Note that we have just given PUPS the forms of these lines. The causal relationships are to be inferred. I will postpone analysis of the second expression, exp2, until after the last substep because that is when it appears in interacting with the tutor. The one thing that PUPS does respond to in this goal line is the identity between the first X, term1, and the second X, term3, in exp1. This identity and the contiguity of the two leads to the inference that the first is the cause of the second. It is a bit tricky to judge whether this causal inference is correct or not. One might argue that the second X is the cause of the first because the second X is not isolated. Fortunately, this turns out to a causal inference that does not figure into later extrapolations.

5.2 The First Substep

The first substep is encoded in PUPS as:

line1: isa tutor-line
form: (line reciprocal of term6 is term7)

term6: isa number
form: (text 3)

term7: isa fraction
form: (stack term8 – term9)

term8: isa number
form: (text 1)

term9: isa number
form: (text 3)

PUPS uses the overlap between the goal line and the first line to introduce a causal link between the two. It also correctly identifies the cause of the first 3, term6, as the 3, term2, from the goal line. On the other hand, it uses the knowledge it already has of the reciprocal to predict the 1/3 in the result box.

When it comes to solving a later variable-isolation problem PUPS can use the causal hooks it has built to set up the first step on the way to solving the problem. The following is the rule that can be extrapolated from the causal analysis:

```
IF =problem: isa tutor-line
        form: (line isolate =term1 in =exp -> =box)
   =exp: isa expression
        form: (list =term2 =term3 = =term4)
   =term2: isa number
        form: (text =value)
THEN =problem function: (cause =nextline)
       =nextline: isa tutor-line
            form: (line reciprocal of =term6 is =term7)
       =term6: isa number
            form: (text =value)
```

where =term7 will be filled by another production. Similarly PUPS analyzes substep 2 (Multiply) as being caused by the goal line and substep 3 (first Simplify) as being caused by substep 2.

5.3 Fourth Substep

PUPS encoding of the fourth substep is:

```
line4: isa tutor-line
     form: (line simplify exp7 -> exp8)

exp7: isa expression
     form: (list term28 = term29 term30)

exp8: isa expression
     form: (list term31 = term32)

term28: isa variable
     form: (text X)

term29: isa fraction
     form: (stack term33 – term34)

term30: isa number
     form: (text 6)

term31: isa variable
     form: (text X)

term32: isa number
     form: (text 2)

term33: isa number
     form: (text 1)
```

```
term34: isa number
    form: (text 3)
```

PUPS makes two causal ascriptions: It relates this line to the previous and exp7 to the result of the previous line. From its knowledge of simplification it also predicts the result of this line. The rule which does so was acquired from previous lessons and has the form:

```
IF =line: isa tutor-line
       form: (line simplify =exp1 -> =exp2)
   =exp1: isa expression
       form: (list =term1 = =fraction =term2)
   =fraction: isa fraction
       form: (stack =term3 - =term4)
   =term1: isa variable
       form: (text =name)
   =term2: isa number
       form: (text =value2)
   =term3: isa number
       form: (text 1)
   =term4: isa number
       form: (text =value1)
   =fact: isa multiplication-fact
       form: (sequence =value3 times =value2 is =value1)
THEN =exp1 function: (cause =exp2)
      =exp2: isa expression
          form: (list =term5 = =term6)
      =term5: isa variable
          form: (text =name)
      =term6: isa number
          form: (text =value3)
```

5.4 Answer to Goal Line

The final thing that PUPS does is to process the filling in of the answer in the result slot of the goal line. It correctly attributes this as caused by the result of the fourth substep. PUPS has inserted causal chains leading from the original expression $3X = 6$ and the multiplication fact "2 times 3 is 6" to the result of the goal line, $X = 2$. Figure 2 illustrates these causal chains. By composing these causal chains PUPS infers that $3X = 6$ caused $X = 2$ and that the 2 in the result is caused by the multiplication fact. It can compile a rule from these composed causal links directly prescribing what should go into the result box of the goal line. The rule that can be compiled, in the same form as the rule for the fourth line, is given below:

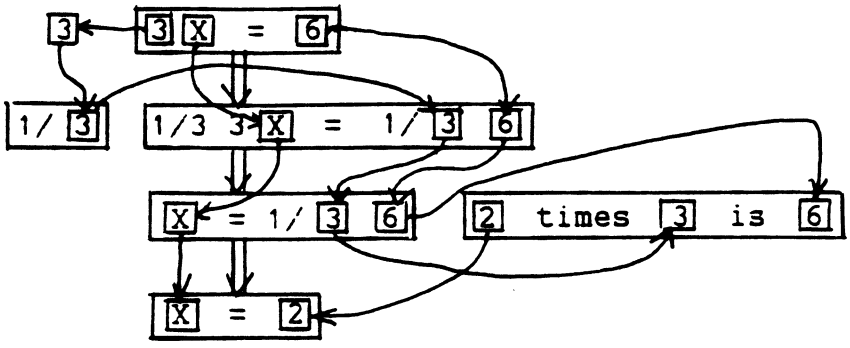

Figure 2. Representation of the critical causal linkages in PUPS analysis of Figure 1.

```
IF =line: isa tutor-line
        form: (line isolate =term1 in =exp1 -> =exp2)
   =exp1: isa expression
        form: (list =term2 =term3 = =term4)
   =term1: isa variable
        form: (text =name)
   =term2: isa number
        form: (text =value2)
   =term3: isa variable
        form: (text =name)
   =term4: isa number
        form: (text =value1)
   =fact: isa multiplication-fact
        form: (sequence =value3 times =value2 is =value1)
THEN =exp1 function: (cause =exp2)
       =exp2: isa expression
            form: (list =term5 = =term6)
       =term5: isa variable
            form: (text =name)
       =term6: isa number
            form: (text=value3)
```

5.5 Comments of the Algebra Tutor Example

The algebra domain seems to be particularly well-chosen for purposes of testing out the PUPS learning mechanisms. For instance, we can define routines to go from the tutor screen to the form encodings on which the PUPS causal routines operate. Thus we can automate the entire process of knowledge acquisition. Perhaps by the time of the conference, we will have had enough time to report a complete or nearly complete simulation of learning with the tutor in contrast to this hand simulation.

It needs to be said that we already know that PUPS learning with the algebra tutor is far from perfect. In particular, PUPS does not really learn the reasons for performing the steps in an algebraic manipulation. For instance, it does not represent that the reason for performing isolation is that the variable X is embedded in an expression on the left hand side of the equation. When it later uses isolate-variable as part of equation-solving it will view that operation as being caused by previous tutor lines—just as it analyzed the sequence of steps underlying isolation. Thus, PUPS makes what we call rote errors where it performs a substep which is usually part of a sequence of steps but which is not required in a current context. For instance, the variable might already be isolated. However, it needs to be added that students are very prone to these rote errors. Thus, the problem is not really one with PUPS but rather with the learning situation. These rote errors provide students and PUPS with the occasion for learning the critical features controlling these operations. In PUPS these features would be stored as preconditions not causes.

The fact that we can put PUPS behavior in correspondence to student behavior illustrates another advantage of working with the tutor. The PUPS analysis of these errors suggests that the tutor should be amended to teach students the features which control the setting of goals such as isolate. Students, like PUPS, tend to react to the salient features of the problem and infer that such features are the causes. We need to make salient the more abstract and relational features such as the fact that X is embedded with other terms on one side of the equation.

Acknowledgements

The development of PUPS has very much been a joint effort with Ross Thompson. The research on PUPS was supported in part by contract MDA 903-85-K-0343 from the Army Research Institute. The work on the algebra tutor is supported by grant 84-70337 from the National Science Foundation. Preparing this paper was facilitated by the gracious hospitality of Flinders University.

References

Anderson, J. R. (1983). *The architecture of cognition*. Harvard University Press, Cambridge, MA.

Anderson, J. R., & Thompson, R. (1986). Use of analogy in a production system architecture. Paper presented at the Illinois Workshop on Similarity and Analogy, Champaign-Urbana, IL.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning, 1*, 145–176.

Lewis, C. L. (1986). *Why and how to learn why: Analysis-based generalization of procedures* (Technical Report CS-CU-347-86). Boulder: Department of Computer Science, University of Colorado.

Lewis, M. W., Milson, R., & Anderson, J. R. (in press). Designing an intelligent authoring system for high school mathematics ICAI: The TEACHERS APPRENTICE Project. In G. Kearsley (Ed.), *Artificial intelligence and instruction: Applications and Methods*. Reading, MA: Addison-Wesley.

Mitchell, T. M., Keller, R. M., & Kedar-Kabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning, 1*, 47–80.

Neves, D. M. (1978). A computer program that learns algebraic procedures by examining examples and working problems in a textbook. *Proceedings of the Second Conference of the Canadian Society for Computational Studies of Artificial Intelligence* (pp. 191–195). Toronto, Ontario, Canada.

Shultz, T. R. (1982). Rules of causal attribution. *Monographs of the Society for Research in Child Development, 47* (1, Serial No. 194).