EXPLANATION-BASED KNOWLEDGE ACQUISITION OF SCHEMAS IN PRACTICAL ELECTRONICS

A Machine Learning Approach

John H. Mayer

University of Michigan

Technical Communication Program
Technical Information Design and Analysis Laboratory
2360 Bonisteel Blvd.
Ann Arbor, MI 48109-2108

Technical Report No. 32 (TR-90/ONR32)

September 12, 1990

This research was supported by the Office of Naval Research, Cognitive Science Program, under Contract Number N00014-88-K-0133, Contract Authority Identification Number NR 442-f002.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for Public Release: distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR-90 / ONR-32	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION University of Michigan 6b. OFFICE SYMBOL (*If applicable*)	7a. NAME OF MONITORING ORGANIZATION Cognitive Science Office of Naval Research (Code 1142CS)
6c. ADDRESS (*City, State, and ZIP Code*) Technical Communication Program Ann Arbor, MI 48109-2108	7b. ADDRESS (*City, State, and ZIP Code*) 800 N. Quincy Street Arlington, VA 22217
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-88-K-0133
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 61153N; PROJECT NO. RR04206; TASK NO. RR04206-0f; WORK UNIT ACCESSION NO. 442f002

11. TITLE (*Include Security Classification*)
Explanation-Based Knowledge Acquisition of Schemas in Practical Electronics: A machine learning approach

12. PERSONAL AUTHOR(S)
John H. Mayer

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Technical Report	FROM 11-1-87 TO 9-12-90	September 12, 1990	103

16. SUPPLEMENTARY NOTATION
This report is the PhD dissertation of the author. For further information, contact project PI; David Kieras, at above address.

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUB-GROUP	Training, Learning, Machine Learning, Explanation-based Learning.
05	09		

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

This report describes an AI system that learns electronics concepts from the content of training materials similar to those used in military training in practical electronics. The system is given a series of circuits to learn about; each is described with a circuit diagram and a text expressed in propositional form that explains how the circuit acomplishes a specific function. The system uses a naturalistic domain theory to verify the claims made in the text, and then uses explanation-based learning techniques to generalize the explanation and construct a schema for the circuit that can be used to understand later circuits that include the schematic circuits as subcircuits. The system successfully understands later circuits in terms of the schematic ones, and shows savings in some measures of processing as well, and has implications for the design of technical instructional material. Certain important shortcomings of explanation-based learning and schema concepts become clear with this work, and are discussed in some detail.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	
22a. NAME OF RESPONSIBLE INDIVIDUAL Susan Chipman	22b. TELEPHONE (*Include Area Code*) (202) 696-4318 22c. OFFICE SYMBOL ONR 1142CS

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Abstract

This report describes an AI system that learns electronics concepts from the content of training materials similar to those used in military training in practical electronics. The system is given a series of circuits to learn about; each is described with a circuit diagram and a text expressed in propositional form that explains how the circuit accomplishes a specific function. The system uses a naturalistic domain theory to verify the claims made in the text, and then uses explanation-based learning techniques to generalize the explanation and construct a schema for the circuit that can be used to understand later circuits that include the schematic circuits as subcircuits. The system successfully understands later circuits in terms of the schematic ones, and shows savings in some measures of processing as well, and has implications for the design of technical instructional material. Certain important shortcomings of explanation-based learning and schema concepts become clear with this work, and are discussed in some detail.

Chapter 1

Introduction

One of the chief obstacles to the development of intelligent systems is the knowledge acquisition bottleneck. The most common approach to capturing knowledge has been knowledge engineering, in which a programmer codes the knowledge he elicits from a domain expert. Another possible source of knowledge is text. In this research I investigate the feasibility of automated knowledge acquisition from text in a typical technical domain, namely practical electronics.

Work in qualitative reasoning has made considerable progress toward developing robust domain theories to support common sense reasoning in the domains typically treated by technical materials, most notably electronics (DeKleer, 1984; Kuipers, 1984). Furthermore, technical prose often presents major concepts by means of explanations and work in machine learning has lead to development of a technique for acquiring a concept from an explanation, namely Explanation-Based Learning (EBL) (DeJong & Mooney, 1986; Mitchell et al. 1986). Taking advantage of these recent advances, I have built a system that learns circuit concepts from explanatory text in the domain of practical electronics.

My approach has been to treat each explanation, which I represent with a sequence of Prolog clauses, as an incomplete proof which describes the operation of a particular circuit. Machine comprehension of this explanation requires completing the proof by matching textual claims about circuit behavior against a simulation of the circuit supported by an electronics domain theory. The completed proof is generalized using EBL and yields two associated concepts, one describing the circuit's structure and the other its behavior. Later when the learned circuit is found embedded as part of a more complex device, use of the new concepts to predict its behavior through use of a single rule should reduce the problem solving effort required to analyze the new circuit.

Ironically EBL has not been applied to the processing of written explanations. Construction of the explanation is the computationally most expensive part of any EBL system. An explanatory text is not the same as the complete causal structure required by EBL, but it can be thought of as an outline of that structure. Filling in the outline is surely a simpler process than creating it. We would therefore expect that EBL systems based on explanatory text would have a decisive computational advantage over systems that are essentially left on their own during problem solving. Explanatory text also identifies goal concepts for the learner. Whenever a passage is devoted to explaining the workings of some new system, the learner can infer that the device is an important category which should be learned.

I have chosen practical electronics pedagogy as the domain for this project since electronics instruction is often organized around a well established set of high-level concepts corresponding to commonly encountered circuits. These are the concepts which are developed through explanation, typically a short passage describing how a circuit works. Most electronic equipment is at least partly a composition of these common subcircuits and can be analyzed in terms of them. Thus every standard radio has an oscillator, a detector, and stages of amplification, and an understanding of these components makes discussion of a complete radio much simpler. There is considerable agreement across electronics manuals and textbooks as to what the most important common substructures are. This extends to agreement on names and preferred ways of drawing diagrams of them. These common subcircuits are what I will call *conventional schemas* or schemas for short.

I hypothesize that it is the schemas that organize what can be usefully learned and transferred from one electronics reading to the next. To test this hypothesis and the suitability of EBL to acquiring schemas, I have implemented an automated reader/learner as a Prolog program and applied it to an integrated set of instructional readings that introduces seven simple electronic circuits. The program consists of two major parts, the comprehender and the learner. The comprehender module captures aspects of the reading task, producing complete explanations from incomplete textual ones which I have manually translated into Prolog clauses. The domain theory which supports the construction of proofs from explanations is largely determined by the attempt to model the common sense intuitions of the typical beginning student of electronics. The learner module is a domain-independent implementation of the EBL technique. It extracts new rules from the complete explanations and makes them available to the comprehender for use in processing later readings.

With this model, I have been able to conduct a machine experiment. The experiment has two conditions: the learning mechanism is enabled in one of them, but not in the other. In the learning condition, the system acquired rules describing the structure and behavior of seven circuits analyzed in a set of instructional readings. It then successfully applied these rules in the course of understanding readings about more complex circuits, suggesting the basic feasibility of automated reading in technical domains. The performance of the system in reading the passages shows considerable improvements in the learning condition, suggesting the basic feasibility of automated reading in technical domains.

In the course of the project, it has become clear how greatly the benefits of EBL depend on the particular computational assumptions of the implementation. The benefits of EBL are clearest when resources are measured according to the number of production rule cycles initiated during simulation of the circuit. This metric implies that any number of rules can be fired in constant time, but is reasonable for architectures which support parallel processing. On the other hand, in some instances use of EBL was detrimental in terms of CPU times, because the current implementation is neither parallel nor optimized for pattern matching.

The project has revealed some limitations of EBL. The most important concerns its usefulness in forming rules for more complex circuits. While the technique was successfully applied to an interesting set of D.C. devices, it was not possible to use EBL to capture the behavior of some A.C. circuits. In the case of the D.C. devices, it was always possible to derive the behavior of the complete circuit by composing the behavior of its subcircuits, since the latter remained highly local, even when they were combined in complex ways. By contrast, some A.C. circuits are quite sensitive to the way in which they are combined with other components; the behavior of such circuits often cannot be predicted locally. A single rule generated from analysis of such a circuit in isolation often cannot predict its behavior in combination with other circuits. Apparently, however, studying these circuits in isolation does allow people to reason more easily about composite devices, and so it is interesting to see the difficulties which prevent this sort of transfer from being captured by straightforward application of EBL.

Aside from the prospect of automated knowledge acquisition, I have also been interested in how this system might be used to ease the difficult task of designing effective instructional prose. To the extent this system can model how people learn from text, it can also be used to validate the design of a piece of instructional text, quantifying the benefit of the concepts used to organize the material and identifying flaws in their development through the text. The current system is especially suited to tracking the coherence of the explanations and subsequent use of principles derived from those explanations. If the machine cannot derive a useful principle from a given explanation, it may be that the typical human learner will also have difficulties.

Chapter by Chapter Overview

In Chapter 2, I discuss EBL. In Chapter 3, I describe the explanatory texts on which the project is based and characterize their structure. In Chapter 4, I describe the implementation of the reader/learner, including a domain theory for practical electronics. In Chapter 5, I analyze the performance of the reader/learner in a machine experiment based on the explanatory texts, while in Chapter 6, I discuss several difficulties encountered while developing the system. Finally, in Chapter 7, I present my conclusions from this research.

Chapter 2

Review of Explanation-Based Learning

In this work, I have adopted EBL as a model for learning from textual explanations. This approach to machine learning is quite recent and is being actively investigated by a number of researchers. My own approach is largely based on the formalization given by Tom Mitchell, as discussed in Section 2.1, while also incorporating various extensions suggested by Gerald DeJong, which I present in Section 2.2. EBL uses a problem solver and a powerful domain theory to acquire a new rule from an analysis of a single *training example*. Essentially, the EBL method works as follows: the problem solver and the domain theory are used to produce a proof tree for some proposition typically called the *goal concept*. The data used in the proof, i.e. the leaves of the proof tree, are a subset of the training example. A mechanism outside of EBL identifies both goal concept and the training example. Since the proof tree was constructed through search, it represents potentially valuable problem solving experience. To capture this experience a new rule is derived from the proof tree and made available for future problem solving. When this new rule fires successfully, it will recreate some or all of the proof without incurring the full cost of the original problem solving.

The new rule is redundant with respect to some set of the old rules. Thus, in principle, the problem solving capabilities of the system do not increase after learning, though the new rule may result in faster processing and therefore have the same effect as addition of new control knowledge.

2.1 Mitchell's Approach to EBL

These concepts can be made more concrete by examining an example discussed by Mitchell. The example is based on Winston's ANALOGY program (1983), which learned to visually recognize a class of objects given only a high-level functional description of the class and representative members of it. Winston's program learned to recognize various kinds of drinking cups in this way.

Mitchell's formalization identified four inputs to the EBL procedure. The first input, the goal concept, is a high-level functional description of the concept to be learned, i.e. the goal concept describes what it is that a cup ought to do in order for it to be a cup. As a rough approximation a cup ought to be an open vessel which is stable and liftable. Because it is given mostly in functional terms (stable, liftable) and avoids getting bogged down in describing structures that might implement these functions, this definition is fairly easy to formulate. It succinctly specifies the entire set of cups. On the other hand, the goal concept is no great help in actually spotting a cup just because it is so abstract and far removed from the sort of features that the vision subsystem might detect. A camera and relatively standard vision algorithms may be able to identify spheres, toruses, and cylinders in a given relation to each other, but spotting "liftable" as required by a high-level functional description may involve expensive computation. The second input is the training example. This consists of an actual cup presented to the vision subsystem which produces a description of it in terms of its lowest-level structural features. The third input is the domain theory. The program can use the domain theory to prove that the sample cup is everything it ought to be, namely an open vessel that is liftable and stable. The sample cup is shown to be an instance of the goal concept. The theory includes rules that allow the deduction of abstract functional properties such as liftable from a combination of structural properties and less abstract functional ones.

The fourth input is the operationality criterion. Many definitions of the goal concept are possible, but they meet the operationality criterion only if they are computationally useful in performing some task. In the example we have been developing, the task is to efficiently recognize cups. The example goal concept is not operational because using it for recognition would not be efficient. The problem solving required to demonstrate that the observable structure of an instance satisfies the functional requirements of the goal concept is too expensive. One plausible operationality criterion for the cup recognition system is that the definition should consist exclusively of terms used in describing the training example and other "easily evaluated predicates". Such a definition is guaranteed to classify cups based on data produced by the vision subsystem and a minimum of computation.

In a sense, at the start of processing, an EBL program already has two very different starting points for learning about cups - the goal concept and the training example. Since each of these is inadequate for the task of recognition, a third conceptualization is derived from the interaction of the first two. The goal concept is a description of the whole class of relevant objects, but in terms which are not easily manipulated by the recognizer. The training example is given exclusively in such terms and therefore

is operational for the task of recognition, but describes only one member of a possibly infinite class. The EBL output will be a third definition which is operational and applies to all instances of the goal concept that share the essential features of the training example, or in other words the acquired concept is an operational generalization of the training example.

The EBL method for achieving this generalization depends on first proving that the training example is an instance of the goal concept, i.e. that the sample cup is an open vessel which is liftable and stable. The principles used in the proof are drawn from the domain theory and the axioms are various structural features of the training example. Note again that in principle the system is able to classify any given candidate cup in this way before it has done any learning; this is in fact a requirement for applicability of the method. The difficulty is that generating such a proof is too expensive prior to learning.

The first benefit from having constructed a proof is that features are immediately partitioned into relevant and irrelevant ones - clearly a crucial step toward generalization. Those structural features which turn up in the proof as axioms are relevant since they apparently contribute to meeting functional requirements. Those features that don't turn up can be dismissed as unimportant variation between members of the class. Whenever we delete a feature from the set which defines the training example, we generalize since the remaining features describe an ever larger superset of the training instance.

The full set of features defining the cup training instance is shown in Figure 2.1. The representation is a semantic net in which the arcs are labeled with the relations they represent. The proof of "cupness" is shown in Figure 2.2. Note the tree structure. The formula of each internal node has been unified with the consequent of some rule from the domain theory. (In the case of the root, the goal concept has been used.) The formulas of the children of that node are the antecedent clauses. Thus, as can be seen from the leftmost child of the root and its children, one of the domain rules is:

```
open-vessel(X)  <-

        part-of(X,Y), isa(Y,concavity), is(Y,upward-pointing)
```

(N.B. Following the Prolog convention, variables begin with a capital)
The leaves in this tree (i.e. the axioms of our proof) are features appearing in the training example. Only a portion of the features supplied by the example appear in the proof and are therefore candidates to appear in the generalized version of the example. Features describing color and ownership have not been used.

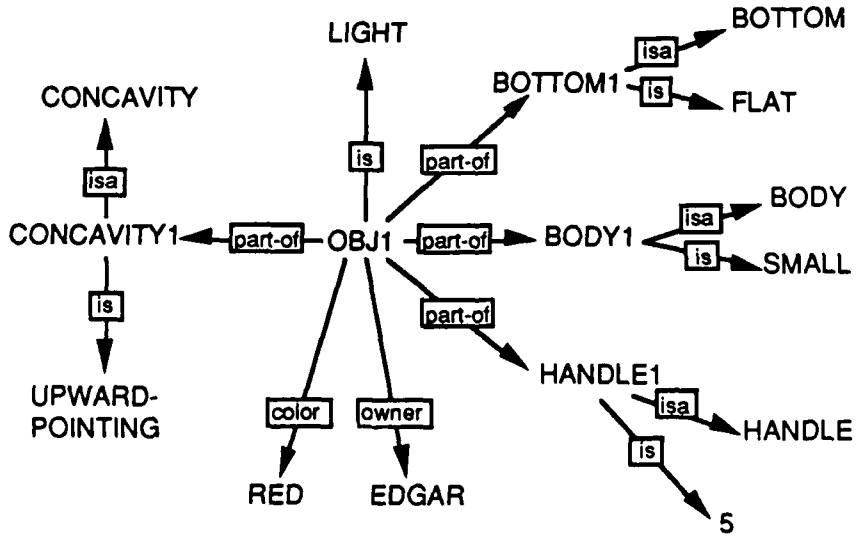


Figure 2.1. Features of the cup training instance. (Adapted from Mitchell, 1986)

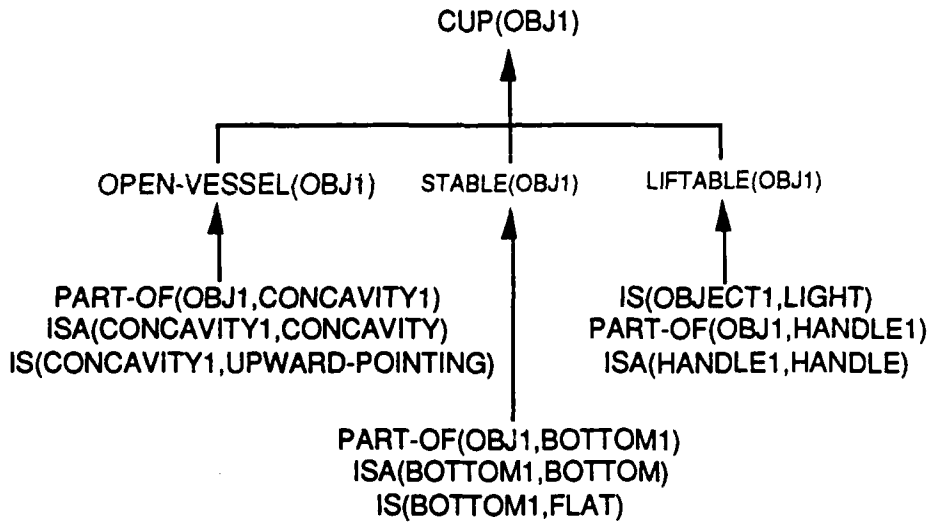


Figure 2.2. The proof of "cupness." (Adapted from Mitchell, 1986)

At this point we are ready to make a first attempt at extracting a new rule for cup recognition from our proof. Taking the proof axioms as antecedents we have the rule of Figure 2.3. The most obvious objection to this rule is that it has no variables and so applies only to the training example. We will therefore replace identifiers with variables to obtain the rule of Figure 2.4.

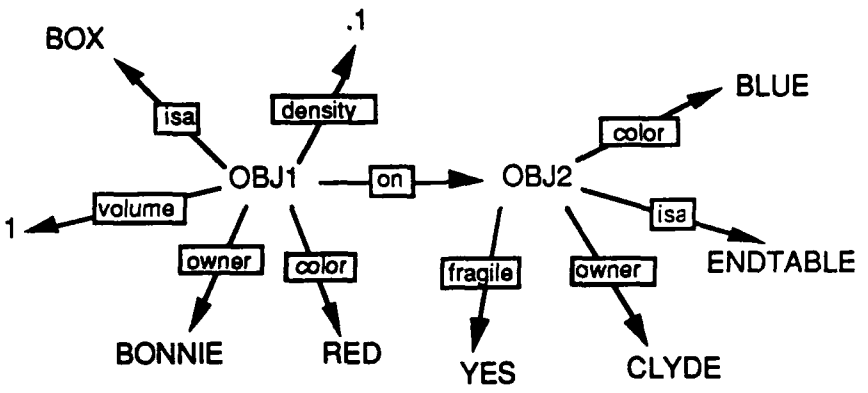
```
cup(obj1) <-
    part-of(obj1,concavity1),
       isa(concavity1,concavity),
       is(concavity1,upward-pointing),
    part-of(obj1,bottom1),
       isa(bottom1,bottom),
       is(bottom1,flat),
    is(obj1,light),
    part-of(obj1,handle1),
       isa(handle1,handle).
```

Figure 2.3. First attempt at a new cup definition.

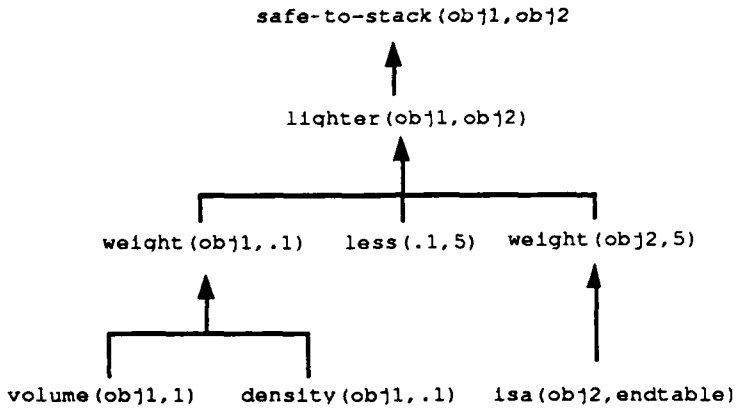
```
cup(O) <-
    part-of(O,C),
       isa(C,concavity),
       is(C,upward-pointing),
    part-of(O,B),
       isa(B,bottom),
       is(B,flat),
    is(O,light),
    part-of(O,H),
       isa(H,handle).
```

Figure 2.4. Second attempt at a new cup definition.

This is a valid and reasonable generalization. In the following example, however, replacement of constants by variables yields a very unsatisfactory generalization. This example, based on learning when one object can be safely stacked on another, is also from Mitchell and is summarized in Figure 2.5. The goal concept is `safe-to-stack`and the figure shows both a training instance (a) and the proof that the objects stacked in the training instance are set one on the other in a stable fashion (b). Figure 2.6 shows the rule derived from constant replacement.



(a) The safe-to-stack training instance



(b) The proof of `safe-to-stack(obj1,obj2)`

Figure 2.5. OBJ1, a box, can safely be stacked on OBJ2, an endtable. (Adapted from Mitchell, 1986)

Note that in Figure 2.6, the clause `less (.1,5)` has been omitted, since although it is an axiom in the proof constructed for the example, it can be statically evaluated and deleted as an optimization. This rule makes clear the inadequacy of our first attempt at a generalization algorithm. The rule is a valid one, but misses a fairly obvious opportunity for even greater generality. It is not crucial that the volume of X be 1 and its density .1; any combination of volume and density with a product less than five will do. Thus we would prefer the rule of Figure 2.7.

```
safe-to-stack(X,Y) <-
     volume(X,1),
     density(X,.1),
     isa(Y,endtable).
```

Figure 2.6. First attempt at rule for safe-to-stack.

```
safe-to-stack(X,Y) <-
     volume(X,V),
     density(X,D),
     less(V*D,5),
     isa(Y,endtable).
```

Figure 2.7. Second attempt at rule for safe-to-stack.

Various techniques for deriving such generalized rules from proof trees have been developed. Generally they are variants on goal regression. Goal regression is a very powerful technique for finding the necessary and sufficient preconditions which will allow some rule R to be used to derive a formula F. This notion is then extended to find a set of necessary and sufficient preconditions which will allow a whole set of rules (i.e. a domain theory) to be used to derive F. Taking the formula

```
safe-to-stack(A,B)
```

as F and the definition

```
safe-to-stack(X,Y) <- not(fragile(Y)) or lighter(X,Y)
```

as R, the regression of F through R would simply be

```
not(fragile(B)) or lighter(A,B).
```

This expression can in turn be regressed, one predicate at a time, through definitions of fragile and lighter and so on until we finally derive an expression which is operational, i.e. given in terms of easily computed predicates. If there are multiple definitions of fragile or lighter, then all of them will have to be explored.

Actually, full-scale goal regression is simply too expensive and unwieldy a procedure to be of much use in reformulating a goal concept. Notice, for instance, how the single regression step shown leads to a doubling in the number of terms in the expression because the `safe-to-stack` concept was a disjunction. This will clearly lead to a dangerous exponential explosion in the length of the expression and ultimately in the size of the rule. By exploring every disjunct of every rule, goal regression produces a rule for recognizing *all* the members of the class defined by the goal concept - i.e. every conceivable cup. Generally this is very wasteful; only a fraction of all conceivable cups will be encountered in practice. By contrast, EBL derives less comprehensive rules designed to recognize typical instances by generalizing one or more training examples.

It is for this reason that EBL modifies standard goal regression liberally. As before we start with a single formula F to be regressed. This time, however, we do not work backwards through the whole set of domain rules; instead we use the proof tree built during analysis of the training instance. Beginning at the root, F is regressed back through the particular rule which was used there (the goal concept, as before). Then whatever preconditions are derived are regressed in their turn through rules used by children of the root. In this manner, the leafs are eventually reached. Instead of exploring every disjunct of every rule, we consider only the rules that are actually used in the proof tree, and only disjuncts that were actually satisfied there. Let us trace this procedure for the `safe-to-stack` example.

The initial formula,

```
safe-to-stack(A,B),
```

is unified with the portion of the definition that was satisfied in the proof:

```
safe-to-stack(X,Y)   -> lighter(X,Y).
```

Since the disjunct, `not(fragile(Y))`, was not satisfied, it has been omitted. The substitutions generated by unification are `X -> A` and `Y -> B`, so the new current expression is the single term `lighter(A,B)`. In the next step we regress it through the rule

```
lighter(X,Y) <-

  weight(X,WX), less(WX,WY), weight(Y,WY)
```

to get a new current expression:

```
weight(A,WX), less(WX,WY), weight(B,WY).
```

In the example the first weight term was proved by using the rule

```
weight(X,VX*DX)   <- volume(X,VX), density(X,DX),
```

so the first conjunct is regressed through it to yield

```
volume(A,VX), density(A,DX), less(VX*DX,WY), weight(B,WY).
```

Notice how the substitution of VX*DX for WX in the weight term has been applied to the less term as well. The second weight term is regressed through the default reasoning rule:

```
weight(Y,5)  <- isa(Y,endtable)
```

This gives the desired final regressed expression

```
volume(A,VX), density(A,DX), less(VX*DX,5), isa(B,endtable)
```

where substituting 5 for WY while unifying the weight terms led to the term, `less(VX*DX,5)`.

2.2 DeJong's EBL Method

In his own work with an EBL-based story comprehender called Genesis, DeJong has identified several ways in which Mitchell's approach to generalizing explanations, while useful, does not go far enough (DeJong and Mooney, 1986). Some of these missed opportunities are more easily demonstrated in a larger example such as the following one cited by DeJong. Analysis of the narrative in Figure 2.8 results in the explanation in Figure 2.9. The narrative is meant to convey the notion of kidnapping to Genesis. The system understands the passage by instantiating schemas to account for the various events. Thus the program recognizes that part of what is going on early in the story consists of a capture scenario which enables the bargaining schema at the end of the narrative. Schemas are thought of as compiled plans. The mission of the system is to create new schemas to capture novel plans. In order to judge what parts of the story are related to the new technique for earning money, the program isolates that part of the narrative representation that has "John gets the money" for a root. Due to the causal links that have been inferred by the program this structure includes not only the bargaining that immediately precedes the ransom payment, but also the capturing that makes the bargaining possible.

> Fred is the father of Mary and is a millionaire. John approached Mary. She was wearing blue jeans. John pointed a gun at her and told her he wanted her to get into his car. He drove her to his hotel and locked her in his room. John called Fred and told him John was holding Mary captive. John told Fred if Fred gave him $250,000 at Trenos then John would release Mary. Fred gave him the money and John released Mary.

Figure 2.8. DeJong's kidnapping Story

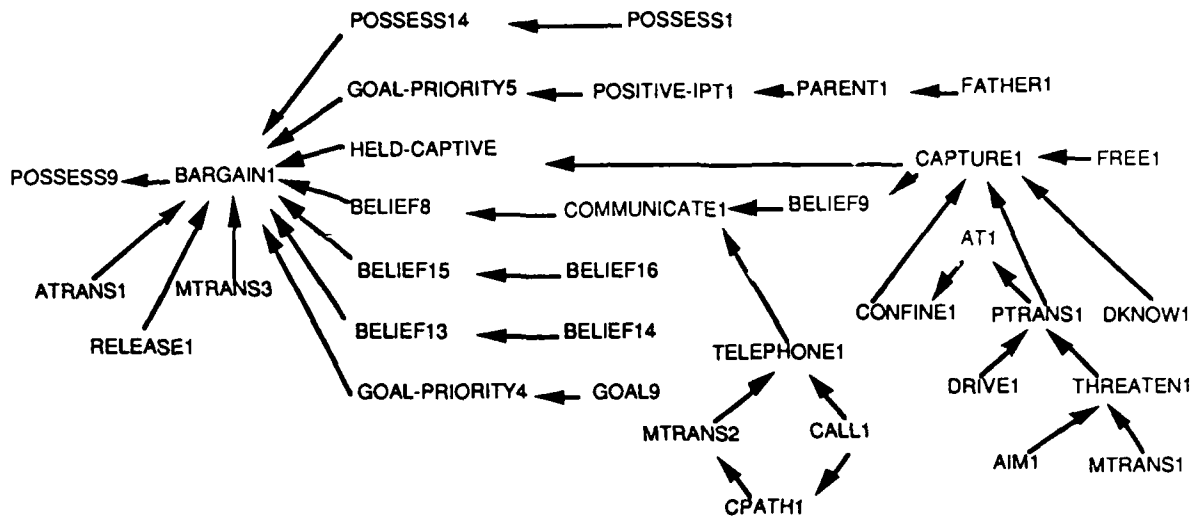


Figure 2.9. Full explanation structure.

Up to this point differences between Mitchell's approach and DeJong's are mostly cosmetic. Granted, the first uses theorem-proving with inference rules as its model for understanding an example, while the second prefers schemas, but as noted by DeJong, these differences are nothing fundamental. Both give rise to an explanation structure, i.e. a tree of propositions in which the children of each node are the evidence for having asserted the node's proposition.

The differences in the approaches are apparent in what is done to the trees. As discussed above, Mitchell proceeds immediately to a goal-regression procedure which essentially gathers up the leaves of the tree and generalizes their arguments as much as possible, converting some to variables, rewriting others as expressions in these variables, and leaving still others as literals. The end result is a set of terms which, if they should match any future instance, would allow us to make exactly the same argument for the root proposition as was used in analysis of the example.

Unfortunately, there will many instances which are very similar to the example, and for which large portions of the same reasoning are applicable, but which vary from it in some more or less minor details. Therefore a rule formed by Mitchell's approach will not fire in the majority of the cases where the example ought to be relevant.

As an example consider that generalizing the kidnapping explanation structure in Figure 2.9 will make the kidnapper's use of the telephone an integral part of "generalized" kidnapping; therefore in processing a future story in which the kidnapper sends a letter, the system finds its kidnapping concept too narrow and has to try deciphering the meaning of the capturing and bargaining without the benefit of the useful precedent it has already come across.

This problem of undergeneralization arises because the explanation structure which is generalized to yield a rule is too deep; it reaches down so far that its leaves are likely to be details of the example which while causally relevant to proving that the instance is an example of the goal concept, will not always turn up, even in quite similar examples.

One way to keep overly specific features out of the generalization is to prune certain portions of the tree, which has been done in Figure 2.10. One of DeJong's suggestions is to eliminate those nodes which support inferences to more abstract actions. Thus in the case of our kidnapping example, the kidnappers must communicate their demands before bargaining can occur. The fact that they do this by telephone is not especially relevant. The only role of telephoning is to support inference of the more general action - communicating. By deleting the node TELEPHONE1, we are altering the set of leaves which will appear as antecedents in the generalized rule. Unduly specific nodes are eliminated in favor of more abstract ones which they support.

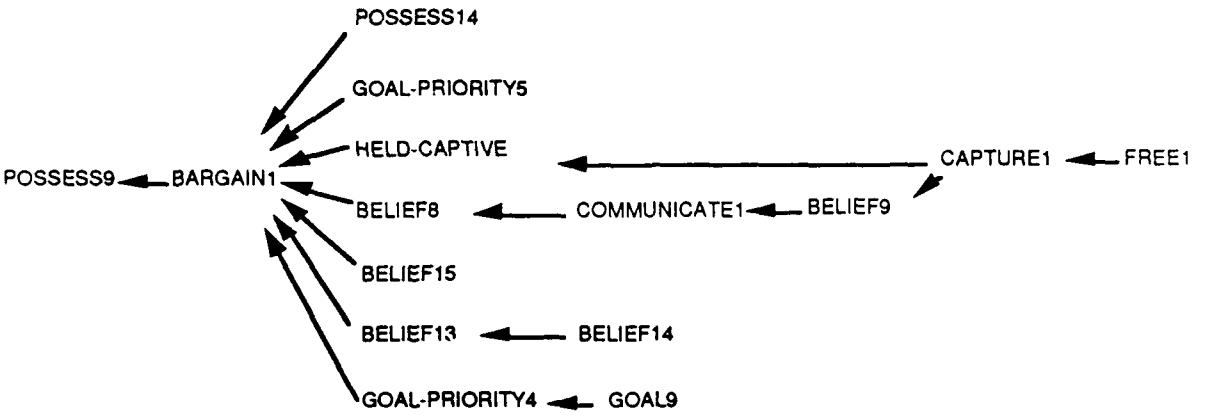


Figure 2.10. Explanation structure after detail deletion.

Another method used by DeJong to obtain a more general rule by deleting portions of the explanation structure is to omit propositions that are merely details in realizing some schema. The program which reads about kidnapping has a schema for capturing. Also the only role played by the component actions for that schema is to allow the system to infer the capture event. Therefore the nodes beneath CAPTURE1 have also been removed in Figure 2.10. This is reasonable as there is no point in remembering exactly how this kidnapper captured his victim. The kidnapper will be in just as good a position if he captures his victim with a lasso. By only retaining the top-level description of the capture act, we have once again generalized the example in a very useful way. Overall these modifications to Mitchell's generalization procedure are extremely valuable as they clearly lead to more widely applicable definitions.

2.3 Producing Explanations for EBL

All EBL projects share the broad strategy outlined above. One of the greatest sources of variation among EBL-based learners is the extent to which they either learn independently or need the help of an instructor. Input can benefit a system in at least three ways. First, the system can be left to identify goal concepts on its own, as Genesis does, or the instructor can specify them for it. Second, the instructor can choose training instances of appropriate difficulty. Since EBL requires problem solving, depending on the gap between the system's current domain knowledge and any given training example, analyzing the training example may be beyond the system's capabilities (in practice if not in principle). The instructor can prevent this by directing the system toward training instances that match its current aptitude. Third, the instructor can assist the program in building this structure. The following discussion surveys a number of EBL-based systems and evaluates them along these lines.

Genesis (Mooney and DeJong)

To return to Genesis, the approach taken by DeJong and Mooney (1985) is to have the learner read casually, i.e. the learner is given stories or newspaper articles not specifically intended to be instructive. A narrative comprehension subsystem builds explanations for whatever it can understand with its current schemas, paying special attention to the planning behavior of the various characters. Because no attempt is made to match the system's current level of competence with the difficulty of the readings, it is not always possible for the program to generate the causal explanation required for generalization. The current story may simply be too hard for Genesis, given its current schema knowledge. If the causal structure of the narrative cannot be inferred, then the program learns nothing. Aside from wasting time trying to understand things which are too difficult, Genesis must also spend a good deal of time on stories about planning for which it already has schemas, as no instructor is available to steer it toward more interesting material. On the other hand Genesis is a very independent pupil and can eventually learn about kidnapping without benefit either of a textbook or tutor. In addition, Genesis makes deliberate decisions about goal concepts. It looks for novel plans in which the planner obtains an important goal (i.e. money, love, career advancement). The plan is novel if the system fails to recognize it as an instantiation of some current schema.

Learning New Principles (Winston)

The system presented in Winston's "Learning New Principles from Precedents and Exercises" (1982) is almost as independent. The program is given a story and must "understand" it by identifying the underlying causal structure of the narrative. The program is then given an exercise which consists of an incomplete narrative and a question relating to its completion. The first story is meant to serve as a precedent for the exercise. As an example suppose the first story is about a law suit and its resolution. Then the exercise would be a similar law suit together with a question about its likely outcome. The student is meant to find an analogy between the precedent and the exercise by matching the causal structure of the two narratives. He then uses this match to reason about a likely decision in the exercise based on what the court ruled in the previous case. By juxtaposing precedent and exercise, the instructor is drawing attention to an overlapping piece of reasoning which the student is meant to identify and abstract. Thus the instructor never has to explicitly state the goal concept.

Lex (Mitchell)

The most learner-independent way of generating instances is to let the learner make up its own as in Mitchell's LEX (1983). LEX is initially given a set of operators for solving integral equations. This initial description of the operators includes the conditions under which they may legally be applied. For each operator, LEX infers a corresponding goal concept, namely the conditions for applying the operator effectively, i.e. so that use of the operator leads to a solution. Whereas goal concepts are implicitly specified for LEX by the operators it is given, the examples required to operationalize the goal concepts are devised by the program. Because the evolving goal concept is explicitly represented as an incompletely learned heuristic, the problem generator can identify the heuristic which has been least well learned and propose a problem whose solution is likely to lead to a refinement of that heuristic.

Learning Apprentices

Another approach to bringing instructive examples to the learner's attention is to let the learner observe an expert. The learner is given the goal concept, and while it can observe everything the expert actually does, it does not have any access to the expert's planning. This must be reconstructed by the learner. Once the expert's solution has been understood, a new plan that generalizes the observed problem-solving can be compiled. Segre and DeJong (1985) have used this approach to build a program which learns to operate a robot arm. One advantage an apprentice has over Genesis is that the expert presumably selects problems that are novel without being too difficult for the apprentice to puzzle out.

2.4 Unanswered Questions for EBL

A learning technique such as EBL raises a number of questions. The current research will not address all of them. We are mostly interested in determining whether EBL can capture the circuit concepts illustrated by the instructional readings described in the next chapter.

Utility of New Rules

One of the most difficult questions facing EBL is documenting the performance of rules generated by the learning technique. This involves identifying some set of problem-solving trials and determining whether the benefits conferred by the new rules outweigh the costs. This calculation clearly depends on what trials are used. For example, testing the utility of a kidnapping schema using several readings about kidnapping can only go so far toward establishing the schema's utility. This sort of sample, artificially rich in opportunities to use the newly acquired schema, will tend to overestimate its utility. Likewise the cost of carrying the new rules while processing unrelated material will be underestimated.

The present research tests the utility of new schemas in the context of a coherent set of readings sequenced according to definite pedagogical goals. Each new concept is developed in anticipation of being used soon afterward. We therefore expect an unusually rich set of opportunities for using the new rules. The readings are a reasonable benchmark for determining their utility, since they reflect the structure of realistic cognitive task, namely understanding instructional materials, but clearly it is not the only possible benchmark and others might even be preferable.

Applicability of EBL

Most importantly, the continuing research in EBL attempts to discover the range of problems to which the technique can be usefully applied. It is possible to apply EBL to a wide variety of rule-based system. But not every computation implemented in such a system will benefit from EBL. As an example, imagine the usual procedure for insertion into a binary search tree implemented as a set of productions. Running without a learning technique, the procedure is already optimal. If we now add an EBL module to this system, then no matter what we identify as goal concepts or training instances, we will only create redundant rules which at best will be ignored, and more likely will increase the cost of rule matching.

The only practical way of investigating this issue is to test the EBL technique in a wide range of domains. The domains used in EBL tend to fall into two categories. First are the domains which lend themselves to well-understood formal representations such as mathematics (Mitchell, 1983; O'Rorke, 1984) or design of digital electronic circuits (Ellman, 1985; Mahadevan, 1985; Mitchell et al. 1983). Then there are the programs that attack toy narratives such as Genesis or Winston's programs. Most technical domains are richer than mathematics or digital electronics, but also have domain theories, such as qualitative physics, that are more formal than those explored in toy narrative domains. Thus, this research extends EBL into a technical domain, practical electronics, that offers an important middle ground between the two, a domain that is both richer than mathematics and more formal than reasoning about every-day occurrences; since learning in this domain is important both for machine knowledge acquisition and human instruction, this extension is not only scientifically relevant, but potentially practically useful as well.

Chapter 3

The Structure of Explanatory Prose

Learning schemas from text requires a model of text comprehension. Instructional text comes in a wide variety of forms, including advice, rules, and explanations. Some research has already been done on automatic acquisition of knowledge from advice (Mostow, 1983) and from rules and examples (Moon and Lytinen, 1989). This project focuses on textual explanations. The model used here is largely based on the characterization of explanatory text as an incomplete proof. In this chapter, I will discuss the structure of explanatory prose and the adequacy of modeling it in this way. I will also consider aspects of the textual explanations that lend themselves to use of EBL as a learning model. Finally I will describe the representation of the texts used in the machine experiment.

3.1 Technical Explanation as Incomplete Proof

The texts used in this research can be described as "how-it-works" explanations. A "how-it-works" explanation tries to make clear the relation between the function of a device and its structure. Consider, for example, the passage of Figure 3.1. This paragraph is taken from a manual for amateur radio operators (American Radio Relay League, 1961) and explains the operation of a voltage regulator circuit. There are other types of explanatory text than the "how-it-works" studied here. However this type of argument is widespread and important enough to deserve the central role it will play here.

"How-it-works" explanations are about a device. We can think of a device as a collection of components (likewise devices) each having a characteristic behavior. Once properly interconnected, this set of parts exhibits a composite behavior which makes the device interesting. The text is essentially a list of assertions about internal behaviors, each one causing the next, either as an immediate consequence or via some implicit intermediate steps. Through a combination of picture and the text, the passage explains how the structure of the device supports its characteristic behavior.

> When the load connected across the output terminals decreases, the output voltage tends to decrease. This takes the voltage on the control grid of the 6AU6 less positive, causing the tube to draw less current through the 2-megohm plate resistor. As a consequence the grid voltage on the 807 series regulator becomes more positive and the voltage drop across the 807 decreases, compensating for the reduction in output voltage.

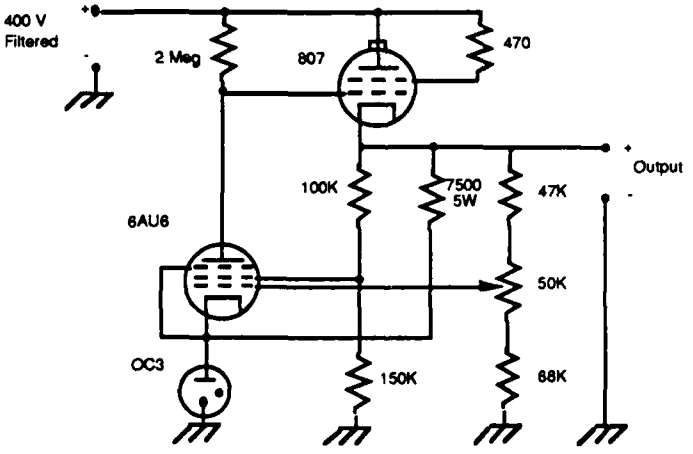


Figure 3.1. Radio Amateur's Guide voltage regulator text and figure.

This explanation can be thought of as a proof including premises, axioms, and conclusion. The premises are propositions describing the structure of the device and an initial perturbation hypothesized at the beginning of the passage. The axioms are drawn from the novice electronics student's domain theory and the conclusion is typically the final behavioral proposition appearing in the passage, which can often be characterized as the principal interesting behavior of the device.

The proof is sketched in the explanation rather than stated fully because the latter would be too tedious both for the author and the reader. What is included in the text are some of the major intermediate steps of the proof, and hints about their logical relations indicated by rhetorical expressions such as "thus", "because", "if", and "causing". Typically the author will not cite the principles of electronics that he has used while tracing out the behavior of the circuit. Nor will he indicate how the structure contributes to this behavior. Nevertheless, the reader understands the natural language explanation most deeply by filling in these details. It is possible for a reader to be more or less critical while reading such an explanation and put correspondingly more or less effort into verifying the account given by the author. The program developed here does not have the luxury of glossing over any part an explanation, however. Instead it constructs a complete proof of the device's behavior since this will be essential to acquiring a generalized concept of the circuit's structure and behavior.

3.2 Comprehension of a Technical Explanation

If the text can be analyzed as an incomplete proof, then comprehension can be cast as completing the proof. This involves searching for some combination of domain principles and prior behavioral assertions that will allow each behavioral assertion in the passage to be verified. The simplest possible structure for the explanation results when each statement can be deduced from the immediately preceding one only. This yields a proof tree for the last claim which is basically linear and includes all the other claims made in the passage. We could describe this as a perfect linear explanation. When this happens, it is especially easy to identify the last proposition as being the goal concept for EBL, since all the other behavior observed is included in subproofs of the proof of the last proposition. Not all of the readings analyzed here were perfectly linear. In some cases, the structure was more genuinely tree-like, starting from a single root at the initial perturbation and then leading to two or more leaves, each describing an output behavior or circuit property interesting in its own right. To form a single goal concept which would capture all of this potentially important information, it was necessary to form a conjunction of the last two or three propositions in the reading.

3.3 Generalizability of a Technical Explanation

Technical prose, such as the set of readings analyzed here, often illustrates a general design principle by explaining its implementation in a typical system. The details of that particular system are of secondary importance and are usually not remembered. On the other hand, the reader is expected to retain its basic structure and operation. By a sort of generalization, the textual account of a single system is converted to an understanding of a whole class of devices. This is the case with the voltage regulator passage. The discussion is by no means a complete theory of voltage regulation, but humans can read this passage and acquire a general notion of the structure and operation of a voltage regulator which they can use in reasoning about any similar device.

Once the technical passage is reduced to a proof or explanation structure, it becomes quite simple to model the rest of the processing as an EBL problem. The goal concept is the high-level functional behavior of the device being analyzed, i.e. the major output signal of the device which is generally described in the last proposition of the reading. In the case of our voltage regulator, the goal concept is the compensation in the output voltage noted in the last line. The training example is the particular voltage regulator which is analyzed in the text and depicted in the accompanying diagram. The domain theory is practical analogue electronics, as described in the next chapter, and the operationality criterion requires simply that any new schema we create should make use of previously acquired schemas whenever possible.

3.4 Overview of the Texts Analyzed

The readings used to test the reader/learner were taken from materials developed for use in a human learning experiment (Kieras & Mayer, 1989), which were based on the type of presentation in the Radio Amateur's Handbook (American Radio Relay League, 1961) and military electronics training material (VanValkenburgh, Nooger & Neville; 1955). They are presented in Appendix 1 along with the propositions that were used to represent their content. The suite covers seven circuits and has been segmented into 13 explanations. Figure 3.2 shows the transfers which were anticipated. For example, it was expected that the triode amplifier schema acquired in readings AMP1 and AMP2 would enhance understanding of the reading CAMP. The cathode bias amplifier schema was expected to transfer from the CAMP reading to the voltage regulator readings, VR1 and VR2,

and the voltage regulator schema was presumed to enhance understanding of the stabilized voltage regulator circuit described in readings SVR1 and SVR2.

There is not a one-to-one correspondence between circuits (e.g Basic Voltage Regulator) and explanations (e.g. VR1 and VR2) because the reading accompanying each circuit generally traces more than one behavioral scenario for that circuit. Often there are two explanations which are complements of each other, i.e. the causal principles used are the same but every increasing quantity has become a decreasing one and vice versa (see SER1 and SER2). This means that there is an opportunity for transfer between explanations of a single circuit. This possibility was recognized by the author of the readings and some of the explanations are therefore incomplete in the original materials. These incomplete explanations begin by hypothesizing an initial perturbation opposite in phase from the immediately preceding explanation. They then trace the behavior for one or two steps before ending the explanation with a phrase such as "... and the opposite effects occur." These incomplete explanations have not been used in the machine experiment. In other cases (see SVR1 and SVR2), there is more than one interesting initial perturbation, each requiring an explanation.

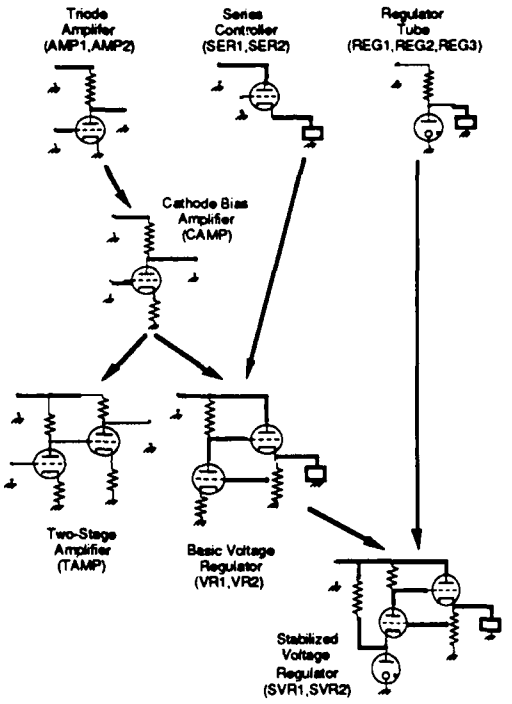


Figure 3.2. Expected transfers between schematic circuits.

3.5 Representation of Explanatory Text

The translation of text to propositions is done by hand. The translation is not complete; it is limited to capturing only the content which can be reduced to simple statements about voltage, current, and resistance. More specifically, the target language is given by the context-free grammar shown in Figure 3.3. A fair amount of the text simply cannot be translated into this target language and has been left out. A good example of this is the second sentence of the reading CAMP which describes a fairly complex relation between the grid and input voltages.

Because the grammar is very limited and the intuitive match of its sentences to the text is often close, most of the manual translation can be done on a syntactic basis, without using knowledge of electronics. I would like, however, to briefly discuss the most important exception to this general rule. As mentioned in the discussion of the domain theory, the common sense notion of voltage regulation or stable or constant voltage depends on proving that one change leads to a countervailing one. The word "constant" is used in this sense in such phrases as "kept constant" or "remains constant" in SVR2. On the other hand the text

sometimes refers to voltages staying the same and means just that - they don't change either way. This is to be seen in VR2 in phrases such as "stays constant," "will remain constant," "is the same," and, "will stay decreased." I have disambiguated these phrases by translating the former examples as a predicate with "constant" and the latter ones with a pair of propositions of the form `not(change(decrease,X))) and not(change(increase,X))`.

```
<statement> ->
        <event>
        constant(<quantity>)
greater(<quantity>,<quantity>)
        higherv(<vl>,<vl>)
produces(<event>,<event>)
        not(<statement>)
        and([<statement>*])
        large(<event>)
        larger(<event>,<event>)
<event> ->
        change(<trend>,<quantity>)
<trend> ->
        increase
        decrease
<quantity> ->
        vage(<vl>,<vl>)
        resistance(<vl>,<vl>,[...])
        current(<vl>,<vl>,[...])
<trend> ->
        increase
        decrease
```

Figure 3.3. Grammar for reading propositions.

Chapter 4

The Reader/Learner Model

4.1 System Architecture

The basic architecture of the system is shown in Figure 4.1. The system is presented with a series of readings. Each reading consists of a circuit diagram for some electronic device, the *Figure*, and a short passage explaining its behavior, the *Text*. The operation of the system with respect to a single reading is as follows. The figure is manually converted to a set of propositions describing the circuit structure. These propositions are passed to a schema instantiation routine which uses a library of schema definitions to parse the circuit. Propositions describing the parsed circuit are one of the inputs to the simulator. The text is manually converted to a set of propositions which are successively verified by the theorem prover, generally by simulating the parsed circuit using a library of simulation rules. After it has processed all the text, the system extracts an explanation of how the device works and generalizes it. The generalized explanation has both structural and behavioral aspects. The structure references are collected together to form a new schema definition which is added to the schema library, and the behavioral assertions are used to form a new rule which is added to the simulation library.

4.2 Representation of Circuit Diagrams

The circuit diagrams are manually translated to a set of propositions representing component types and their interconnections. See Appendix 2 for a listing of this propositional representation for each circuit used in the learning experiment.

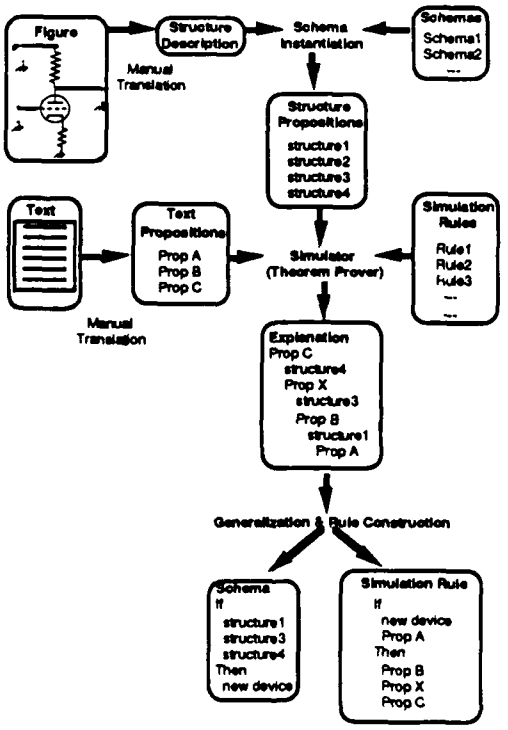


Figure 4.1. System processing.

Electronic circuits are usually drawn according to graphic conventions. Thus a drawing can be criticized as being typical, unclear, or misleading. However, no attempt has been made to capture these conventions or to represent the graphical layout of the circuit drawing.

4.3 Circuit Parsing

The set of propositions derived from the circuit diagram is the input to the schema instantiation routines. These routines make use of a growing library of schema definitions to parse the circuit, labeling each recognizable subcircuit. There is an important preliminary step in analyzing the circuit: before any schema labeling is done, we establish the flow of current through the various components.

Current Analysis

Figure 4.2 shows how current flow in the voltage regulator is analyzed. Proceeding in a way that stays close to a layman's understanding, this analysis begins by identifying potential current paths or *circuits* by tracing out all closed paths through the device. Some components, such as resistors, allow current flow in either direction, while others offer a current path in one direction only. For example, electron flow through vacuum tubes T1 and T2 is only from cathode to plate. The circuits through them are therefore directed. In all, there are five circuits for the voltage regulator: three of them pass through the voltage source and the other two pass through the load and potentiometer, R3, one clockwise and the other counterclockwise. The next step is to identify electromotive forces (EMF's). EMF's are what actually cause current to flow, an obvious example being a voltage source, but also including, in the case of an A.C. circuit, a charged capacitor or an inductor through which a changing current is flowing. The only EMF in the voltage regulator is the voltage source. Next the EMF's found along each circuit are qualitatively summed. If the sum can be shown to exceed zero, as for the three circuits which include the voltage source, we further describe the circuit as being a *biased circuit*. If a component is part of exactly one biased circuit, then we can infer the direction of current through that component. Such components are represented as *biased components*. If a component is part of more than one biased circuit, current flow can still be inferred, provided each of the biased circuits implies the same flow. Thus, current flow through the voltage source can be determined, since the three biased circuits agree in assigning the same direction to it.

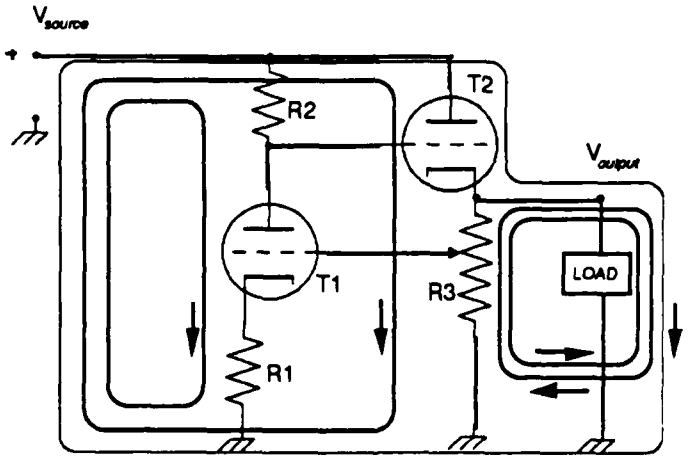


Figure 4.2. Circuit analysis for the voltage regulator showing current paths.

One might object that a student of electronics rarely undertakes such a deliberate analysis of the current flow, but he has quick access to all the conclusions of such an analysis through use of graphical conventions. Visual inspection allows him to make plausible guesses as to where current is going. Generally a component with wire drawn above and below carries current from the top to the bottom. By reducing the figure to topology only, we lose the ability to do this sort of reasoning. The foregoing analysis compensates for that loss.

Schema Instantiation

Schemas are represented in a straightforward way: they specify a set of simpler schemas (or primitive components) and the connections between the various ports of these simpler schemas or components. Initially, however, this library contains just one schema, the voltage divider, all the others being acquired from the readings. The instantiation is essentially an exhaustive

bottom-up search for the part configurations given in the schema library. The schemas are ordered to suit this bottom-up approach and are instantiated serially. The most elementary structures are located first. Then in each successive round, the system will search for higher-level structures defined in terms of simpler ones already found. New schemas are added at the end of this sequence, where they represent a new highest level in the hierarchy.

The system is initialized with definitions for two types of voltage divider, the most primitive and common schema. The purpose of a voltage divider is to tap some high voltage and derive a lower voltage for application elsewhere in the circuit. One variety of voltage divider contained in the schema library is formed by two biased components in series. The other consists of a potentiometer where the movable tap of the potentiometer is used to supply the lower voltage. Much of the reasoning used throughout the readings can be construed as reasoning about voltage dividers.

All the other schemas are learned, and correspond to the circuits shown in Figure 3.2:

1. The triode amplifier
2. The series tube controller
3. The regulator tube circuit
4. The cathode bias amplifier
5. The basic voltage regulator

Most of these are instantiated in the circuit shown in Figure 4.3, which shows a partial schematization of the basic voltage regulator including a voltage divider, a triode amplifier, a cathode bias amplifier, and a series tube controller. Figure 4.4 shows the schema learned for the basic voltage regulator. Notice that it includes all the component schemas learned previously, except for the triode amplifier which is subsumed by the cathode bias amplifier.

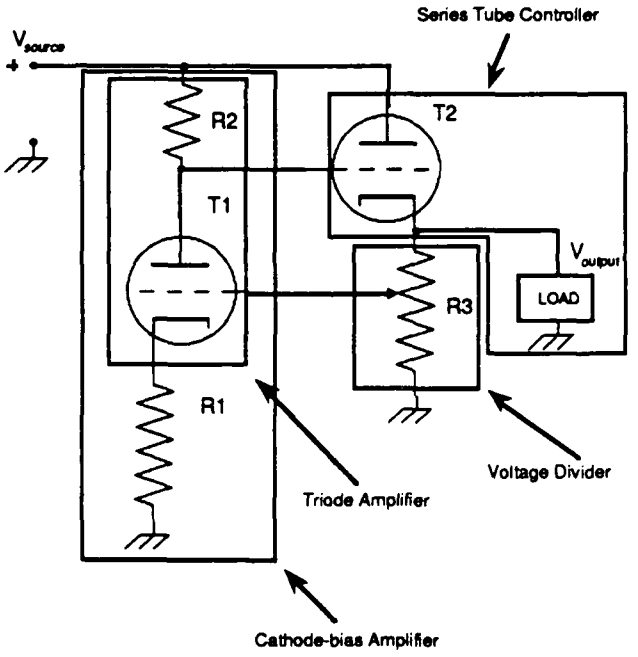


Figure 4.3. Schematization of voltage regulator with component schemas outlined.

```
spot_schema(voltage_regulator) :-
     not(constant(vage(V0,V1))),
     bcomp(_,series_tube_controller,
          [V3,V4,V0,V5,V6,V1,V7]),
     bcomp(_,cathode_bias_amplifier,
          [V9,V10,V11,V0,V5,V12,V13,V14]),
     bcomp(_,cathode_bias_amplifier,
          [V19,V11,V1,V20,V10,V21,V22,V23]),
     comp(_,V26,[V20,V1]),
     resistive(V26),
     bcomp(V28,voltage_divider,[V0,V1,V29,V11]),
     bcomp(V30,load,[V0,V1]),
     gensym(voltage_regulator,VR),
     add(bcomp(VR,voltage_regulator,
          [V9,V0,V1,V30,V19,V11,V20,V10]).
```

Figure 4.4. Structure definition for the voltage regulator.

4.4 Simulator/Theorem Prover

In accordance with the discussion of Chapter 3, the comprehension phase of the program has been modeled as theorem proving applied to a simulation. The domain theory includes both forward and backward chaining rules. The forward-chaining rules (or operators) represent the causal behavior of individual components: each operator has some event among its preconditions which causes one or more events to be inferred. Whenever an operator fires successfully, it makes a claim about some new behavior in the device, which is likely to be interesting. Thus a systematic application of the operators is used to push simulation forward in time whenever the reading runs ahead of whatever predictions have already been made.

```
Add the first reading proposition
For each remaining proposition
       Try to prove the proposition via backward chaining only
       If success,
             return to remaining proposition loop
       Else
          Do forever (resume simulation)
                   For each operator (a cycle of simulation)
                         Instantiate the operator completely
                         Add any new clauses generated by it
                         Try to prove the proposition
                         If success,
                               return to remaining proposition loop
                         Else remove the clauses added and save
                               them until end of the operator loop
                   End (for each operator)
                   Add all the saved clauses
                   Try to prove the proposition
                   If success,
                         return to remaining proposition loop
             End (do loop)
End (for each remaining proposition)
```

Figure 4.5. Reader algorithm for the reader program

Not every inference about an electronic device involves cause and effect relationships. I call other inferences which have no causal interpretation acausal reasoning and represent them with backward-chaining rules. As an example, consider transitive reasoning. If one voltage change is greater than a second which is greater than a third, then the first is greater than the third. Because this reasoning has no causal or temporal aspect, it is less likely to yield a conclusion interesting in its own right, which is why this reasoning is represented by backward-chaining rules, initiated only when needed by the forward-chaining rules or to verify a claim taken from the circuit explanation.

The top level of the automated reader is shown as an algorithm in Figure 4.5. The system always treats the first proposition of a circuit explanation as an assumption. All the remaining propositions are taken as successive claims to be verified. The prover begins this process by determining whether a given proposition can be deduced by backward-chaining from the current state of the simulation. If this succeeds, we continue to the next proposition.

If the current proposition cannot be proved in this way, the simulation is resumed via breadth-first search using the operators. The operators are fired serially according to a fixed order. The order is initially irrelevant; that is, their order is not determined by any domain-dependent consideration. But acquired rules are added at the head of the operator evaluation queue to ensure preferential use of acquired schemas. Firing each operator for every possible set of variable bindings constitutes one full cycle of breadth-first simulation. Although many propositions are verified in one cycle, sometimes several cycles are required.

One of the main goals of the architecture is to terminate search as soon as possible. In particular, we test for provability of the current proposition not just after completing a cycle, but after each successful operator so that a cycle can be interrupted if the most recent operator application has created new clauses leading to a proof. This means use of new rules may lead not just to a reduction in the number of cycles, but also to quicker cycle termination. Since many of the queries are solved in one cycle even without learning, these intra-cycle savings are important. The system is also designed so that within a cycle, operators do not communicate with each other, i.e. clauses added by the fourth operator to fire during a particular cycle are not available for matching preconditions of the fifth operator during that same cycle.

4.5 The Domain Theory

In this section I will present the domain theory used by the reader/learner to create proofs that correspond to textual explanations. Common-sense or qualitative domain theories for electronics have been developed by a number of researchers, such as Stallman and Sussman (1977) or DeKleer (1984), for various purposes. The main constraint I have worked with is the need to model the domain using only knowledge that a student of practical electronics might have. Since I am interested in the pedagogical implications of the automated reader system, I wanted to avoid basing the domain theory on knowledge not available to the intended audience of the readings. The anticipated audience knows basic electricity concepts such as Ohm's law and partial models for all the basic components, but has not studied the full mathematical theory of electronics, such as network theory.

This largely rules out the simple adoption of very sophisticated theories such as DeKleer's. DeKleer's theory includes not only component models which are within the grasp of the typical student and have close functional counterparts in the theory developed here, but also causal heuristics that are much more difficult to justify to the typical reader of a simple electronics text. Of course there are good reasons for the existence of mathematical electronics theory, namely the inadequacies of common-sense understanding of electronics. In this work I use a simple domain theory, not because the theory is the best available, but because it represents the understanding of a particular kind of student.

As mentioned above, the domain theory includes two types of knowledge. The first is knowledge about behavior of individual components which is represented by operators. I discuss the operators next. The second type of knowledge supports acausal reasoning and is implemented by backward-firing rules. They are described after the operators.

The Domain Theory: Operators

An operator captures a small piece of causal reasoning about some component, either primitive, or the instantiation of a schema. The operators have one or more preconditions and on meeting those preconditions, they add one or more conclusions to the simulation. The preconditions include some structural specifications as well as at least one behavioral assertion which is the cause of the resultant behavior(s) described in the conclusion. As an example, consider the operator for the regulator tube (Figure 4.6). The operator is presented in three ways: in English, in an abbreviated qualitative physics notation, and as a Prolog rule. The rule has one structural precondition which refers to the regulator tube, and another requiring a change in voltage across the

tube. That change will result in an opposite change in resistance across the tube. Appendix 3 contains a complete summary of the operators, including an explanation of the formulas occurring in the Prolog rules.

R-REGTUBE The Regulator Tube Rule

A change in current through a regulator tube causes an opposite change in its resistance.

If +(v),
then -(r)

```
bcomp(L,regulator_tube,[PLATE,CATHODE]),
change(C,voltage(PLATE,CATHODE)),
  opp_change(C,OC),
      ->      change(OC,resistance(PLATE,CATHODE)).
```

Figure 4.6 The regulator tube operator.

The Domain Theory: Backward-Chaining Rules

Backward-chaining rules are meant to represent acausal reasoning, i.e. reasoning about the logical, as opposed to the behavioral, consequences of changes observed in the circuit. This section presents the system's acausal reasoning. As an example of such reasoning, consider the following rule written in pseudo-Prolog for inferring a voltage change between two points, P1 and PN (shown as goal on line 1). First we establish a path of N components laid out between P1 and PN (line 2). The first component lies between points P1 and P2; the second, between points P2 and P3; and so on. If we can establish that there is a change C along at least one of the N components (lines 3 and 4) and also that there is no countervailing change along any of the components (line 5 and 6), then we may infer a voltage change between P1 and PN:

```
1     change(C,voltage(P1,PN))  :-
2           there exist pairs [[P1,P2],[P2,P3],...,[PN-1,PN]] such that
3                 for some pair [Pj,Pk]:
4                       change(C,voltage(Pj,Pk))
5                 and for each pair [P1,Pm]:
6                       not((opp_change(C,OC),change(OC,voltage(P1,Pm))))
```

One of the concepts frequently used by the readings is a simple comparison of voltage between two points, i.e. one point in the circuit is at a higher voltage than another point. This is represented with the predicate

```
higherv(Point1,Point2).
```

Higherv is computed during circuit analysis for all pairs of points that are adjacent along some current path and separated by a resistive component. The relation is further defined for two arbitrary points by a transitive definition:

```
higherv(x,y)  :-
      there exist pairs [[P1,P2],[P2,P3],..[PN-1,PN]] such that
      for some pair [Pj,Pk]:
            higherv(Pj,Pk)
            and for each pair [P1,Pm]:
            not(higherv(Pm,P1))
```

Some of the readings, particularly those describing amplification effects, make claims about the relative magnitude of two changes. This sort of comparison is supported by two rules, the first being a straightforward application of transitive reasoning to three quantities represented as voltage(A,G), voltage(B,G), and voltage(C,G).

```
    greater(change(CH,voltage(C,G)),change(CH,voltage(A,G))) :-
greater(change(CH,voltage(C,G)),change(CH,voltage(B,G))),
greater(change(CH,voltage(B,G)),change(CH,voltage(A,G))).
```

The second rule allows further reasoning about relative magnitudes by tacitly assuming that causes and effects are in some sense of the same magnitude. The notation, `produce(change(C1,Q1),change(C2,Q2))` indicates that change C1 produces change C2 either directly or through a series of intermediate changes. Suppose it is given that one change (C2) is greater than another (C1) and also produces a third (C3). Then we can conclude that C3 is greater than C1.

```
greater(change(C3,Q3),change(C1,Q1)) :-
        greater(change(C2,Q2),change(C1,Q1)),
        produces(change(C2,Q2),change(C3,Q3)).
```

This is the formalization of some very approximate reasoning which is not always correct. It is, however, just the sort of reasoning which underlies the explanations given in the reading suite. As an example, one of the most important conclusions drawn by the reading AMP2, which describes the triode amplifier, is that the change in output voltage is greater than the change in input voltage. The circuit and a diagram summarizing its behavior are shown in Figure 4.7. The simulation begins with a change in voltage at the input. Now the domain theory predicts a resulting change in the tube's resistance and also implies that this change will be greater than the change in input voltage. Furthermore the changing resistance results in changing output voltage, but since the relative magnitude of these last two effects is unknown, what can we say about the relative magnitude of the input and output voltage changes? This is where the rule outlined above is useful. It allows us to tacitly assume that the last two effects are comparable in magnitude, because they are causally related. We can therefore conclude that the output voltage change is greater than the input voltage change.

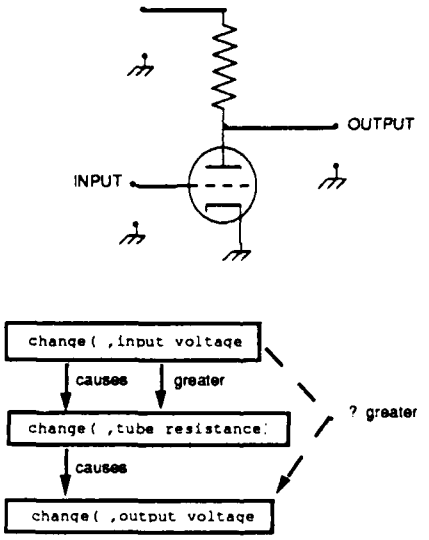


Figure 4.7. Reasoning about amplification effects in the triode amplifier.

A set of three backward-chaining rules was developed to capture a wide variety of inferences concerning voltage across a vacuum tube and its bias element (usually a resistor). Figure 4.8 illustrates these three rules. During simulation it often becomes

clear that the grid voltage of a vacuum tube has changed, as indicated on the left of each diagram. However to make any inference about the behavior of the tube, it is necessary to establish some change in the voltage between grid and cathode (shown dashed). The three rules described below enumerate the cases in which the latter change can be inferred from the former.

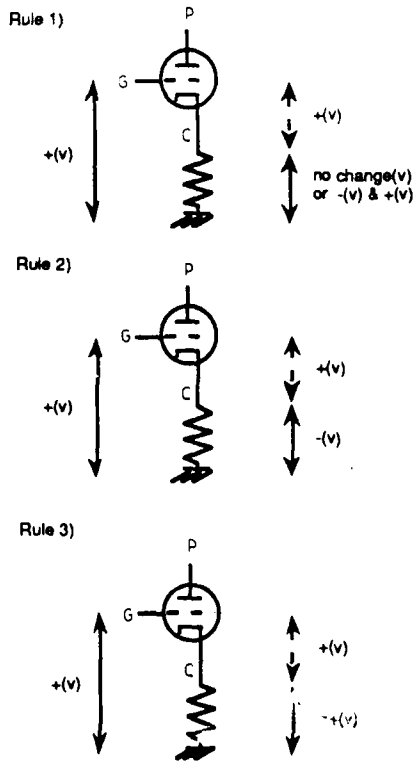


Figure 4.8. Inferring changes in grid voltage.

Cathode Bias Rule 1. If the grid/ground voltage changes one way and the cathode/ground voltage is constant, then the grid/cathode voltage changes in the same way.

```
change(C,voltage(GRID,CATHODE))   :-
        change(C,voltage(GRID,GROUND)),
        constant(voltage(CATHODE,GROUND)).
```

Constant voltage between two points can be inferred in two ways. Either there should be no change noted so far, or there should be compensating changes (both a decrease and an increase).

Cathode Bias Rule 2. If the grid/ground voltage changes one way and the cathode/ground voltage changes in the opposite way, then the grid/cathode voltage changes as the grid/ground voltage did.

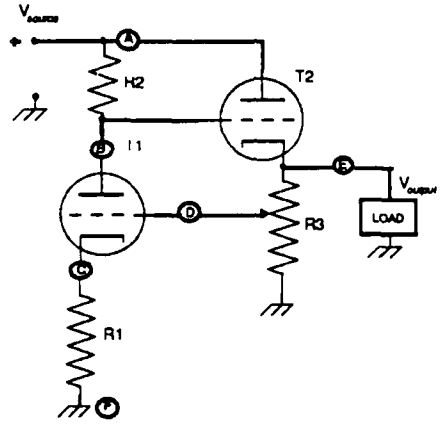
```
change(C,voltage(GRID,CATHODE))   :-
        change(C,voltage(GRID,GROUND)),
        opp_change(C,OC),
        change(OC,voltage(CATHODE,GROUND)).
```

Cathode Bias Rule 3. If grid/ground voltage changes one way and there is no similar change in cathode/ground voltage, then the grid/cathode voltage changes that way too.

```
change(C,voltage(GRID,CATHODE))   :-
        change(C,voltage(GRID,GROUND)),
        not(change(C,voltage(CATHODE,GROUND))).
```

Sample Circuit Simulation

As an example of the use of the operators, the following analysis summarizes the operation of the voltage regulator as described in reading VR1. The propositions from the reading and a diagram of the circuit are shown in Figure 4.9. Table 4.1 shows the simulation in the form of a proof. The proof statements in the table are assertions that some voltage (v), resistance (r), or current (i) is either increasing (+), decreasing (-) or constant (c). Each voltage or resistance is specified relative to two points (A-F) in the circuit. The rationale for each statement lists the domain rules and previous statements supporting each step. Finally those statements from the proof which match propositions from the VR1 reading are indicated.



Reading Propositions

P1 If more current begins to flow through the load,
P2 the output voltage begins to decrease.
P3 This makes the voltage on the grid of T1 more negative relative to the cathode
P4 causing less current to flow through T1.
P5 The grid of T2 then becomes less negative,
P6 decreasing the resistance of T2,
P7 and thus keeping the output voltage the same.

Figure 4.9. The voltage regulator circuit and reading VR1.

The initial perturbation of the system is an increase in current through the load (Step 1). This increase is described by P1, the first proposition of the reading, and syntactically tagged as an assumption by the conjunction "if". It results in a decrease in output voltage (Step 2), as noted in P2. The potentiometer is a voltage divider, so the output voltage decrease is accompanied by a voltage decrease at the potentiometer midpoint, from which we infer that the grid voltage of tube T1 has decreased, not only with respect to ground, but also with respect to its cathode, as stated in P3 (Step 4). Due to the changing grid voltage, resistance across T1 increases (Step 5), resulting in a decrease in current across the tube, as stated in P4 (Step 6). The increasing resistance of the tube also increases the voltage across it (Step 7). Therefore the grid voltage on tube T2 is increasing (Step 8) and this matches P5. The changing grid voltage leads to a decrease in resistance across the tube as noted by P6 (Step 10), as well as a decrease in voltage across it (Step 11). Because the tube and load are in parallel with a voltage source, voltage across the series is fixed, and the decrease in voltage across T2 must be compensated for by an increase in voltage at the output (Step 12). The combination of an initial decrease in output voltage and a resulting increase is recognized as a feedback pattern that results in keeping the output voltage constant (Step 13) and this matches the final reading proposition, P7.

Table 4.1.
Simulation of the voltage regulator.

STATMENT	RATIONALE	READING PROP.
1. +(i E,F)	given	P1
2. -(v E,F)	R-LOAD, #1	P2
3. -(v D,F)	R-VDIV, #2	
4. -(v D,C)	#3	P3
5. +(r B,C)	R-TUBE, #4	
6. -(i B,C)	R-CHOKE, #5	P4
7. +(v B,C)	R-RV, #5	
8. +(v B,F)	#7	P5
9. +(v B,E)	#8	
10. -(r A,E)	R-TUBE, #9	P6
11. -(v A,E)	R-RV, #10	
12. +(v E,F)	R-COMPV, #11	
13. c(v A,F)	#2, #12	P7

4.6 Generalization of the Explanation

As each text proposition is verified during comprehension, the simulation rules used to deduce the proposition are saved together with the variable bindings that allowed the rule to fire. This allows us to reconstruct a proof for each of the assertions made by the text. As noted in Chapter 2, most readings develop a perfect linear explanation, i.e. the proof for each proposition includes the proof of the immediately preceding one.

For purposes of illustrating the rule formation process, Figure 4.10 (a) shows the proof derived from reading REG1. Starting from the bottom of the tree, the rules used in the proof tree are R-LOAD, R-REGTUBE, and R-RV. The top level of the tree uses a backward-firing rule, namely the definition of a constant quantity as one that is being regulated as evidenced by its increasing and decreasing. My own version of goal regression converts the proof tree into a rule tree (Figure 4.10 b); for each internal node of the proof tree, an uninstantiated copy of the rule used to derive the node from its children is created in the rule tree. Thus the actual rules used during the proof and their relation to each other is preserved while premises describing the current circuit are left behind. The rule tree is then traversed so that all unifications implied by the original proof are applied to the rule copies.

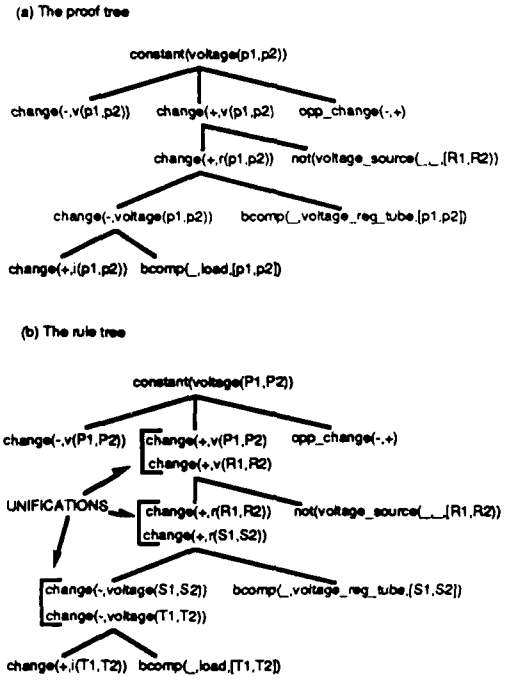


Figure 4.10. Rule formation. The proof tree (a) is used to construct a tree of uninstantiated rules (b) which are merged by unification as shown. Following the Prolog convention, underscores are anonymous variables.

During this procedure, any two leaves in the rule copy structure that were in fact matched by the same clause in the training example are also unified together. This restricts the generality of the rule and reduces the number of instantiations. Without this modification, a rule tree referring to a single resistor three times will lead to a new rule that can be matched to three different resistors. The additional unifications instead lead to a rule placing three conditions on one resistor. Heuristically this is justified by the observation that when a given structure serves more than one function in generating circuit behavior, it is usually no coincidence. Rather it reflects a deliberate economy in design. The same strategy, rationalized in a different way, has been adopted in SOAR (Laird, Newell & Rosenbloom 1987).

Once the explanation has been generalized in the form of the rule tree, it is broken down into a new schema and a new simulation rule (see Figure 4.1). Basically all the statements appearing in the proof that relate to circuit structure are taken as the basis for a structure definition to be added to the schema library, whereas the statements relating to the behavior of the circuit are included in a new simulation operator. The antecedent clauses of this new simulation rule include any behavioral assumptions that the argument rests on (initial perturbations) and reference to some instantiation of the corresponding circuit structure. The consequent clauses include all the behaviors resulting from the initial perturbation. Thus, in a future circuit, if the schema is instantiated, and the initial event is present, all of the implications of that event identified by the generalization process will be immediately added.

An interesting problem occurring at this point is separating out those parts of the explanation which relate to the new circuit being learned and those which describe its structural context in the explanatory example. As an example, consider the regulator tube circuit, shown in Figure 3.2. The circuit itself consists of only the resistor and the regulator tube, but the explanatory passage that accompanies it begins by hypothesizing a change in current through the load. The load therefore figures in the explanation and if reference to it were not pruned, the load would also be mentioned in the structural definition derived from the explanation. This specialization is entirely unnecessary and would prevent recognition of the regulator tube subcircuit found in the stabilized voltage regulator, also shown in Figure 3.2, because the load is no longer in parallel with the regulator tube. There are some textbooks which indicate the extent of a new schema graphically by drawing a line around the relevant subcircuit. The present system is given equivalent information. The description of each circuit from which the system acquires a schema includes identification of the components that make up the schema. This can be seen in Appendix 2, where the clauses used to

represent those circuits include the formula new_device_component. Those parts of the explanation that refer to structure outside the new device are pruned by the system prior to generalization.

4.7 Multi-State Simulations

The system as outlined above can support reasoning about D.C. circuits such as those included in the test readings and lends itself to application of EBL. However other circuits are more difficult to simulate because they require a more complete model of time. I have been able to simulate a number of interesting A.C. circuits using a state-based approach to representing time.

The operation of an electronic circuit across time can be represented as a sequence of qualitative states. The states partition the circuit's behavior into distinct time intervals, each state being characterized by a set of propositions which implies what the next state is. This sort of reasoning is largely domain independent and forms the kernel of the various envisionment algorithms developed by researchers in qualitative reasoning (e.g. Kuipers, 1984).

In Figure 4.11 the actual quantitative model of a discharging capacitor is shown graphically and then segmented into qualitative states according to whether or not the capacitor is charged. Each state is represented by a set of propositions describing the circuit and each proposition is taken to be true throughout the interval associated with the state. The actual length of a state's interval is not generally known or important, but it is useful to distinguish two basic types of interval: those which persist during some period of time, and those which represent only an instant.

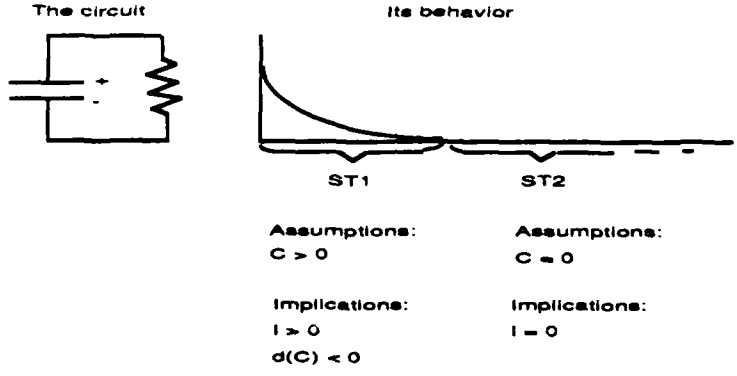


Figure 4.11. Qualitative analysis of discharging capacitor.

The propositions associated with a state may be divided into two groups: first, the assumptions that can be thought of as defining it; and second, the implications of those assumptions. These implications are derived from the structure of the circuit and the sort of electronics domain theory outlined above in Section 4.5. Thus, a typical simulation as shown in Figure 4.11 would begin with an initial state, such as ST1, defined only in terms of the assumption $C > 0$, where C is the capacitor charge. ST1 would then be elaborated by adding all the implications of that assumption, in this case, the existence of a current, $I > 0$, that diminishes the charge of the capacitor, $d(C) < 0$. Once the state description has been completed this way, it will usually include among the assumptions some inequalities such as $C > 0$ as well as some inferred changing quantities such as $d(C) < 0$ that threaten to invalidate those assumptions.

Computing a State Transition

It is often possible to use common-sense reasoning to unambiguously predict a next state for the circuit. Since there are no other changes possible in ST1, $C > 0$ and $d(C) < 0$ can be used to infer that, in the next time interval, $C = 0$, i.e. the circuit enters ST2 where the capacitor is uncharged and there is no current. But generally, determination of a next state requires identifying all the inequalities that could possibly be affected by change in the current state. The simplest of these are constraints that reference a changing value.

There is sometimes ambiguity with respect to the future of even a single inequality. This happens when more than one of the values referenced by it is changing. Table 4.2 shows for a hypothetical circuit how changes in two quantities, x and y, might

determine possible next states. The current state includes the assumption x > y. If x is decreasing and y in increasing, then the next state will be characterized by x = y. If both x and y are increasing, then the next state may or may not be revised to x = y. In actual simulations there will often be more than one revisable assumption, in which case it may not be clear which assumptions will have to be updated in the next state. In the worst case, all possible combinations of these assumptions will have to be considered. Sometimes, however, it is possible to reason about the relative rates of the various changes and exclude some of the possibilities. Thus in Table 4.2, if both x and y are increasing, but it is also known that y is increasing faster than x, we can eliminate x > y as a possibility.

Table 4.2.
Possible projection of x > y into a next state.

	d(y) = 0	d(y) > 0	d(y) < 0
d(x) = 0	x > y	x = y	x > y
d(x) < 0	x = y	x = y	x = y or x > y
d(x) > 0	x > y	x = y or x > y	x = y

Examples of Multi-State Simulation

I have used this state-based representation of time to simulate several circuits, the most complex being the D.C. power supply shown in Figure 4.12. Before presenting a full analysis of that circuit, it may be helpful to look at the first component of the power supply, the A.C. voltage source. Figure 4.13 shows how it is modeled as a primitive component which generates a sine wave at a fixed frequency. In many cases an adequate qualitative abstraction of this wave is a sawtooth curve which can be described by a succession of four states. Each transition is due to the interaction of some changing quantity (the A.C. voltage indicated as V) and a boundary describing the current state.

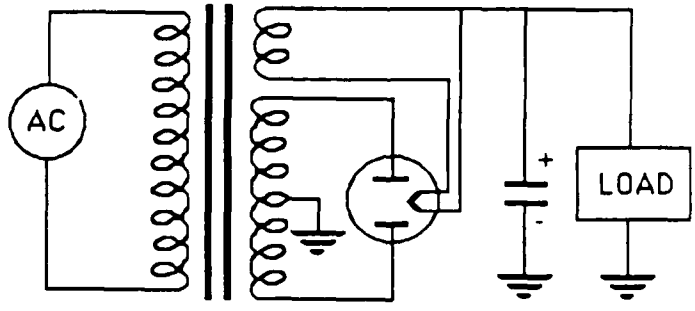


Figure 4.12. D.C. power supply.

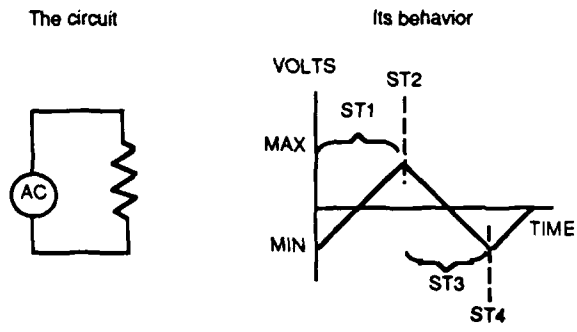
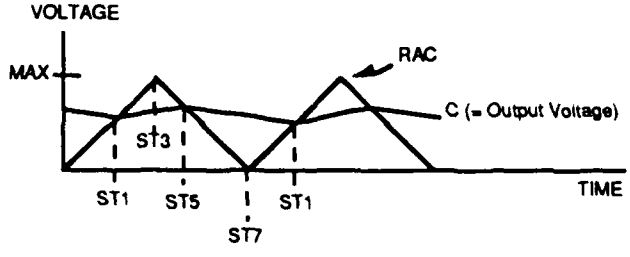


Figure 4.13. Simulation of the A.C. voltage source.

Figure 4.14 shows the simulation of the D.C. power supply. The states are fairly complex. RAC is the EMF due to the rectified A.C. source (the sawtooth portion of the figure). It is bounded between MAX and 0. C is the charge of the filter capacitor. Reading from left to right, the state at any moment is determined by the relation of RAC to MAX, and to 0, the change in RAC, and its magnitude with respect to C. The last column shows the change in C, a particularly important implication of the comparison between RAC and C. When the EMF due to the A.C. source exceeds the capacitor charge, current flows into the capacitor and increases the charge. When the capacitor charge is greater, current flows out of the capacitor.



ST1	MAX > RAC	RAC > 0	d(RAC) > 0	RAC = C	d(C) = 0
ST2	"	"	"	RAC > C	d(C) > 0
ST3	MAX = RAC	"	d(RAC) < 0	"	"
ST4	MAX > RAC	"	"	"	"
ST5	"	"	"	RAC = C	d(C) = 0
ST6	"	"	"	RAC < C	d(C) < 0
ST7	"	RAC = 0	d(RAC) > 0	"	"
ST1	"	RAC > 0	"	"	"

Figure 4.14. Simulation of the D.C. power supply.

All the state transitions are caused by the interaction of the two changing values, d(RAC) and d(C), with the inequalities relating RAC and C to each other and the various bounds. In ST1, for example, since RAC is increasing and C is unchanging, RAC = C can be true for only an instant before it must be updated to RAC > C. There are other boundaries in ST1 that could be jeopardized by the increasing RAC - for example MAX > AC. This is not observed in the transition to ST2, however, since the equality AC = C is invalidated instantaneously whereas an inequality can only be invalidated after some interval of time.

In ST2, there are two propositions that could be invalidated on the way to a next state and therefore the simulation is ambiguous. RAC is moving toward MAX while, at the same time, the comparison RAC > C could conceivably be violated since

both RAC and C are increasing. Which will happen first is not clear. In Figure 4.15, the two alternative transitions for ST3 are shown. In (a) the RAC voltage peaks first; in (b) the capacitor charge catches up with RAC before RAC peaks. Of these two possibilities, the first is actually observed. The simple qualitative domain theory used here cannot provide a rationale for eliminating the second, so a search among the states may be required during simulation. I have not implemented this search and instead selected states manually. The text explaining the circuit's behavior may be sufficient to guide the simulation through a simple breadth-first search of this state space, but it should also be noted that pictures such as Figure 4.16 often accompany texts explaining the circuit's behavior and offer an obvious chance to reduce this search drastically.

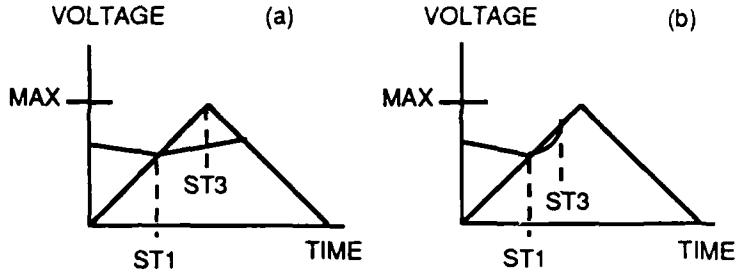


Figure 4.15. Ambiguity in the simulation.

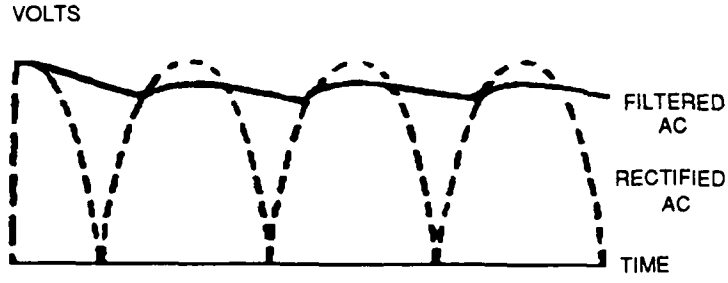


Figure 4.16. Textbook illustration of D.C. power supply output.

It may seem that behavior of any complexity requires a description spanning several states, but this is not the case. All the D.C. circuits simulated in this project were simulated in a single state. To see how this is possible, consider an example of causal behavior typical of these D.C. circuits. Figure 4.17 shows the effect of a decrease in the grid voltage of a triode. This is an ongoing change with no particular duration and no particular limit. It causes an ongoing increase in the tube's resistance. The temporal relation between the two changes is that they are approximately simultaneous, extending over the same interval. We can therefore record them both in a single state. Intuitively, the causal relation between the two changes suggests that the start of the first change must precede the start of the second. However the causal delay is so small compared to the scale of other changes in the circuit, that it can be ignored. Behavior that arises from chaining together a number of these approximately simultaneous cause and effect relationships can also be described in a single state, which is why it is possible to trace out fairly elaborate chains of cause and effect, as seen in the most complex readings, without having to reason explicitly about time in terms of multiple states and transitions between them.

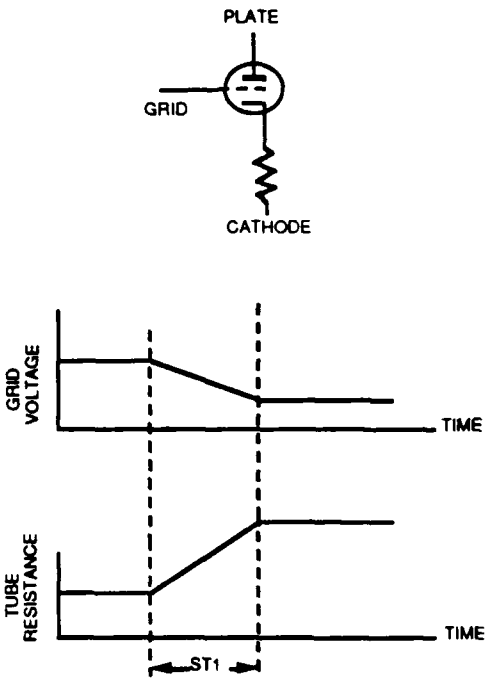


Figure 4.17. Causal behavior within a single state.

Chapter 5

The Machine Experiment and Results

5.1 Experimental Design

I have used the system developed here to acquire schemas from a series of instructional readings which introduce the seven increasingly complex electronic circuits described in Chapter 3 (see Appendix 1). The readings are designed as an integrated set, the latter circuits incorporating the earlier ones. The earlier readings are mostly an opportunity to acquire schemas while the later ones are mostly a chance to use them. One of the important questions addressed by the simulation is whether the schemas are demonstrably useful to the reader program. I assume that knowledge of conventional schemas allows people to learn more effectively by identifying structures whose behavior is likely to be consistent over some range of applications. This consistency means that time spent building a representation of the structure's behavior in one circuit can result in some advantage when that representation is used to reason about the behavior of the structure in another circuit. The question then is whether the behavior associated with the structures is consistent enough to confer any benefit on a reading comprehension program running in conjunction with EBL.

To test this assumption, I have conducted an experiment in which the readings are processed in two different conditions, learning and non-learning. A simplified version of this experiment is shown in Figure 5.1. In Condition 1, the non-learning condition, a typical complex circuit is processed using only the initial domain theory and this requires a certain amount of resources indicated here simply as a reading time of 100 seconds. In Condition 2, the learning condition, the EBL module is active. Since an earlier reading has generated a new and relevant schema which is subsequently used in processing the more complex circuit, the resources required decrease. Figure 5.1 shows the acquisition of only one schema and the possibility of a single transfer between two readings. But the full set of passages leads to the acquisition of five schemas and thirteen transfers among the readings as shown in Figure 5.2. Each node represents a reading. The base of each arrow identifies a reading that supported acquisition of a schema and the head of the arrow shows a passage where that rule was used.

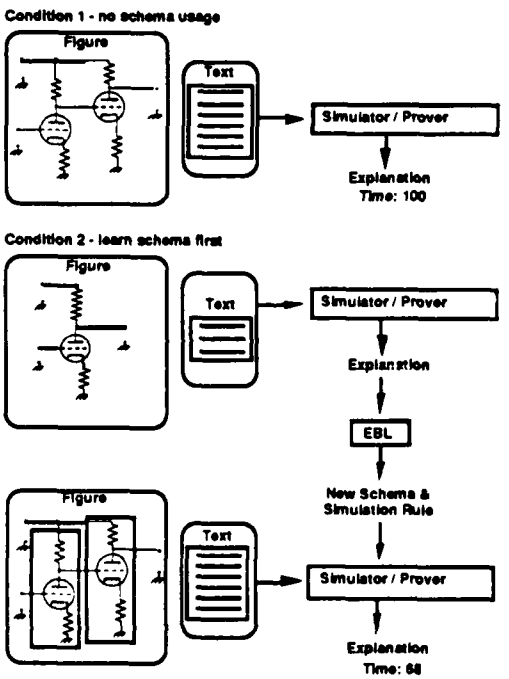


Figure 5.1. The Learning Experiment for Two Readings. When a schema hasbeen learned from the single tube circuit, it can be instantiated in the two tube circuit, reducing reading time for the latter.

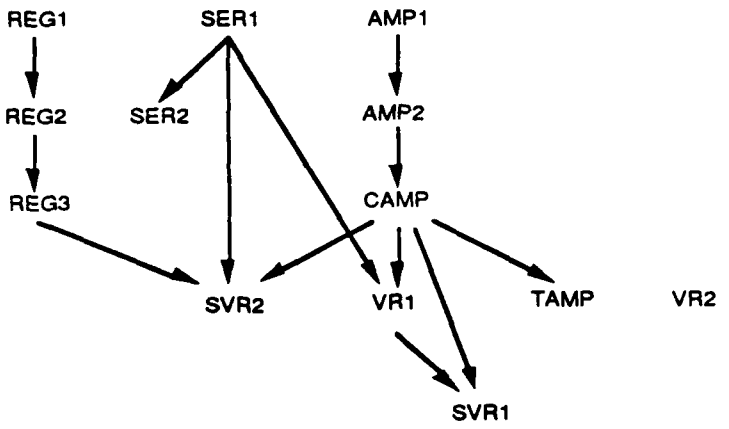


Figure 5.2. Actual transfers observed among explanations.

Another of the questions the experiment is meant to answer is: what learning techniques are needed by the system to acquire the schemas? Inductive similarity-based techniques and EBL are two major alternatives suited to very different learning problems. Given the research interest in combining the two (Lebowitz, 1986; Pazzani,1988), it appears that some problems require use of both approaches. In this work it seemed that EBL alone would probably suffice.

One of the hallmarks of EBL is the expectation that a concept will be acquired after just one trial. This expectation seems applicable to the practical electronics domain where each schema circuit is typically presented and described once. Another indication that schema acquisition is an EBL-suitable problem is the availability of an extensive, explicit domain theory that explains how the circuit's behavior arises from its structure. Since the only value of an electronic circuit is its behavior, the constraints on its structure implied by that behavior ought to be a complete definition of the concept. Therefore I have assumed that EBL is the only learning technique required. If the acquired schemas are overly general or overly specific, that will serve to disconfirm this assumption.

5.2 Overview of Results

The results of the experiment are fairly complex, so I will first summarize them in this section before reviewing them in detail in the remainder of this chapter. The results of the reading simulation experiment suggest that the schemas are of considerable benefit. The behavior of the schematized subcircuits was consistent across readings, leading to many cases of transfer and a large decrease in the number of simulation cycles initiated, the most interesting of the performance metrics. Cumulative cycles for the entire learning condition fell 38% compared to the non-learning condition, suggesting that the acquired concepts may be able to confer some benefit to a reader program.

Furthermore, the acquired schemas were generally consistent with expert intuitions. Their quality validates the assumption that acquisition of high level concepts in the electronics domain requires little or no induction. The chief exceptions to this finding will be discussed below.

The performance of the system as revealed by run times is somewhat ambiguous. There were a number of very clear reductions in CPU times, but there were also a few readings which ran slower in the learning condition, resulting in a 5% increase in cumulative reading time. There are a number of factors accounting for this. There was some over-generalization in the learned structure definitions which, while it did not lead to any erroneous reasoning about those structures, diminished the computational efficiency of the system. Furthermore, the present system is far from optimal in computational support for pattern-matching and pays a higher price than necessary for the acquisition of new rules.

The remaining sections discuss first, the validity of the acquired schemas (5.3); second, their utility according to a relatively implementation-independent metric, namely simulation cycles (5.4); and third, their utility as measured by actual run times for reading comprehension (5.5).

5.3 Validity of the Acquired Schemas

Before beginning a reading, the system does a complete parse of the circuit. This parse includes instantiation of the newly acquired schemas. It is therefore a fairly simple matter to check the validity of the acquired schemas, by comparing the circuit parse with the intuitions of an electronics domain expert. The question of their validity then amounts to whether the learned schemas were found where they were expected and not elsewhere. By and large the acquired schemas were instantiated in a way that is consistent with human intuitions, but the discrepancies are revealing and I will discuss them at some length, and show how additional learning mechanisms and heuristics can address these problems.

Multiple Schema Instantiations

There was some tendency to instantiate schemas multiply. This can be seen especially in the schematization of the SVR circuit (see Figure 5.3). There we find two instantiations of the cathode bias amplifier, each based on tube T2. They can be distinguished according to the bias element they identify. The purpose of the bias element as stated in the training reading is to raise the voltage of the cathode above the ground level. This constraint can be satisfied either by the potentiometer R3 or by the load. Actually neither of these is a very good example of a cathode bias amplifier, so it is difficult to prefer one or the other of these analyses: they are both counter-intuitive.

The schematization also identifies two regulator tube circuits: one that pairs the regulator tube with the resistor R1 and another that pairs it with the tube T1. Again the difference is not significant as far as our ability to correctly analyze the behavior of the voltage regulator, but a human expert would prefer pairing R1 with the regulator tube. This touches on general knowledge about design, i.e. every part of the structure has some function. T1 is recognized as part of a triode amplifier, but R1 has no function aside from its association with the regulator tube. Also the corresponding component in the regulator circuit training reading is a resistor. The practical consequence of these multiple instantiations is that reasoning about all of them takes extra time without yielding any benefit.

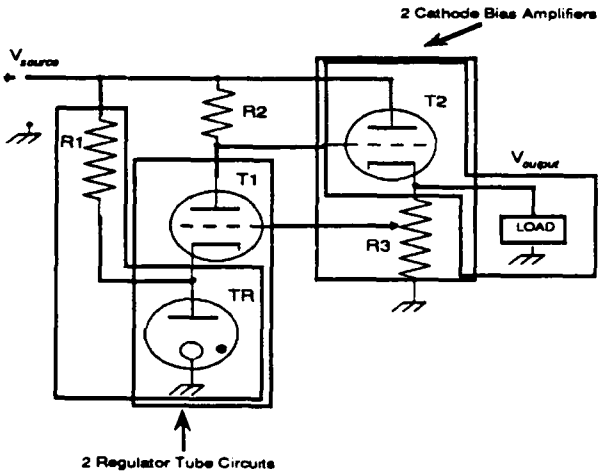


Figure 5.3. Schematization of stabilized voltage regulator showing multiple instantiations of two component schemas.

Overgeneralization

The system suffers from a tendency to overgeneralize. Figure 5.4 shows the range of this effect in the case of the schema acquired for the cathode bias amplifier. Circuit (a) is the training circuit including a vacuum tube, plate resistor above and bias resistor below. Beneath are two circuits which the acquired schema successfully parsed. Subcircuit (b) varies from a typical amplifier by having a voltage regulator tube in place of the bias resistor. Classifying it as an amplifier may be described as creative, but justifiable. Subcircuit (c) has a potentiometer in place of the bias resistor and no plate resistor. To label it an amplifier is a violation of expert intuition.

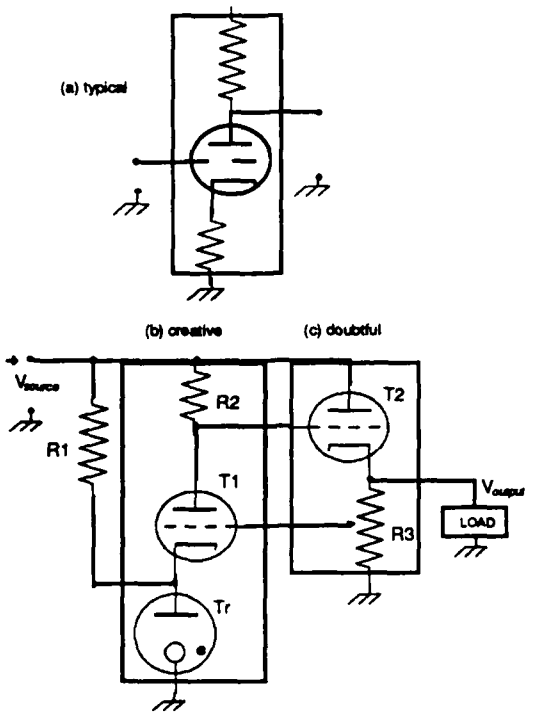


Figure 5.4. Overgeneralization of acquired schemas (a) in the training circuit for the cathode bias amplifier. (b) and (c) are instantiations of the schema.

The CAMP schema inherits its overgenerality from a component schema, the triode amplifier. Figure 5.5 shows the original training diagram for that circuit (a). The only behavior claimed for this circuit is that when grid voltage on the triode changes in one sense, voltage across the tube will change in the opposite direction. EBL analysis shows that the only constraint this behavior imposes on the structure of the circuit is that the triode must be in series with a resistive element. The resistive element is not necessarily above or below the triode. The other circuit of Figure 5.5 (b) shows a structure which meets this constraint and exhibits the same behavior as the original amplifier. If the resistor of this "inverted" amplifier is replaced by a potentiometer, the result is the doubtful, overgeneralized cathode bias amplifier, Figure 5.4 (c). This is actually a specialized amplifier circuit, the "cathode follower," which provides no amplification of voltage.

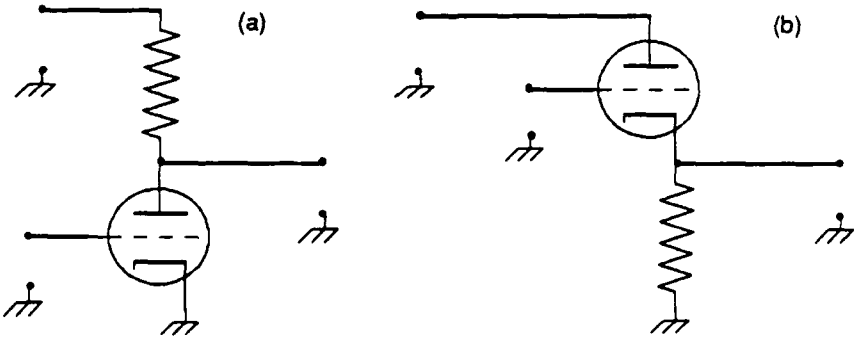


Figure 5.5. (a) Training circuit for triode amplifier; (b) overgeneral instantiation.

From this example, I conclude that knowledge of how electronic circuits work is not always sufficient to constrain structural definitions for common circuit categories. Similarity-based heuristics principles are also needed. With similarity-based heuristics, the system could further constrain the acquired amplifier concept to include the fact that the resistor is above the triode.

It would also be possible to limit overgeneralization by incorporating knowledge of additional design principles into the learning algorithm in a simple heuristic. As mentioned in Section 4.6, the EBL algorithm has already been modified so references to the same structure found at different points in the explanation are merged into a single clause in the resulting rule. This specialization reflects a design heuristic, namely that if one component satisfies several constraints, then that economy is probably deliberate and will also characterize other versions of the device.

A second modification along similar lines would require that references to different structures in an explanation not be allowed to match a single structure when applied as a rule. This can be justified by supposing that if a single component were really capable of playing roles originally filled by several, then the training circuit would have taken advantage of the economy as well.

This heuristic might have prevented overgeneralization of the CAMP schema since the training circuit for CAMP, Figure 5.4 (a), shows two resistors: the one above the triode is the plate resistor and the one below it is the cathode bias resistor. If the explanation produced for this circuit references both of them as well as the triode, then the proposed heuristic would generate a rule requiring three separate components. This rule would not apply to the doubtful amplifier shown in Figure 5.4 (c). Unfortunately, not all explanations of CAMP include references to both resistors, which points to another general design principle which would benefit the system: every component has some purpose. If it is necessary to choose between an explanation which references every part of the device, and one that references only some, the system ought to adopt the former.

5.4 Utility of the Acquired Schemas: Cycles

In this section I discuss the gains due to reasoning with the acquired circuit schemas. As explained in Section 4.4, circuit simulation is structured as a series of cycles during which each forward-firing rule is matched to the database in all possible ways. The learning effect of the EBL module can be seen in a reduction of the number of cycles required to process many of the readings. Since EBL introduces additional rules that do not preclude firing the old ones, in the worst case, i.e. if the new rules are of no use, the reading will require just as many cycles in the learning condition as in the non-learning condition. Thus the number of cycles gives a best-case estimate of the performance of a hypothetical system in which the cost of adding additional rules to the system is negligible. As an approximation, we may be willing to assume this is the case for people and that an efficient pattern-matching architecture would make it true for computer programs as well. With that caveat, we can proceed with a detailed analysis of the cycle data.

The cycle results on the whole are quite good. They confirm most of the predictions we would have made based on intuitive analysis of the reading suite. A tabulation of the cycle savings is included in Table 5.1. Table 5.1 shows the number of cycles in both the non-learning and learning conditions as well as the percentage improvement (i.e. decrease). This is shown graphically on a reading by reading basis in Figure 5.6 and on a cumulative basis in Figure 5.7.

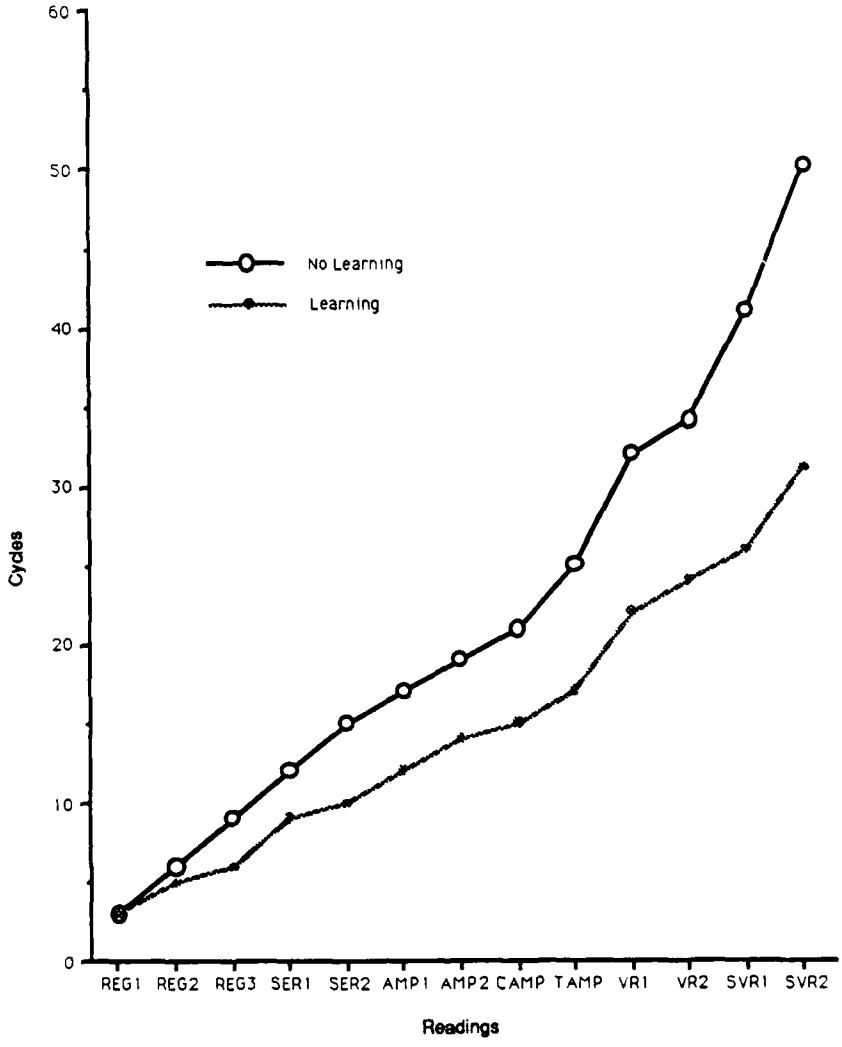


Figure 5.6 Effect of learning on cycles.

To explain how the number of cycles is determined, Figures 5.8 through 5.11 show a trace of which operators are applied during the readings. As an example of how to interpret these diagrams, consider the reading, REG1 in Figure 5.8. In both conditions, the system produces an explanation using the three-operator sequence, R-LOAD, R-REGTUBE, R-RV. In the learning condition some of that explanation is generalized to form the new operator, NR-REG, as indicated by the bracket. The first operator is not used in the new rule because it refers to the load which is not a part of the device being learned. In reading REG2, the non-learning condition applies operators R-VDIV, R-REGTUBE, and R-RV, but in the learning condition, the last two operators in this sequences are replaced by NR-REG, reducing the cycle count by one. The schema acquired in REG2, NR-REGB, has an even more important effect on reading REG3 which requires only one cycle, since readings REG2 and REG3 are complementary (see Section 3.4). SER1 and SER2 are also complementary readings, so the single new rule NR-SER suffices to explain all of SER2.

Table 5.1.
Simulation cycles required in each condition.

	Cycles		
Reading	No-Learning	Learning	Improvement
REG1	3	3	0%
REG2	3	2	33%
REG3	3	1	67%
SER1	3	3	0%
SER2	3	1	67%
AMP1	2	2	0%
AMP2	2	2	0%
CAMP	2	1	50%
TAMP	4	2	50%
VR1	7	5	29%
VR2	2	2	0%
SVR1	7	2	71%
SVR2	9	5	44%
TOTAL	50	31	38%

In contrast, no saving is observed in AMP2 (Figure 5.9), even though the rule acquired in AMP1 does transfer. The problem is that AMP2 describes aspects of amplifier behavior that AMP1 does not touch on. Thus even though the system can incorporate NR-AMP into the final explanation produced for AMP2, it has to supplement use of NR-AMP with a retreat to reasoning on the basis of first principles, including again firing the operator R-TUBE despite the fact that some of the logic supported by R-TUBE has already been captured in NR-AMP. The AMP1 reading only reasoned about the vacuum tube's signal inverting function, while the AMP2 reading depends on the tube's amplification property.

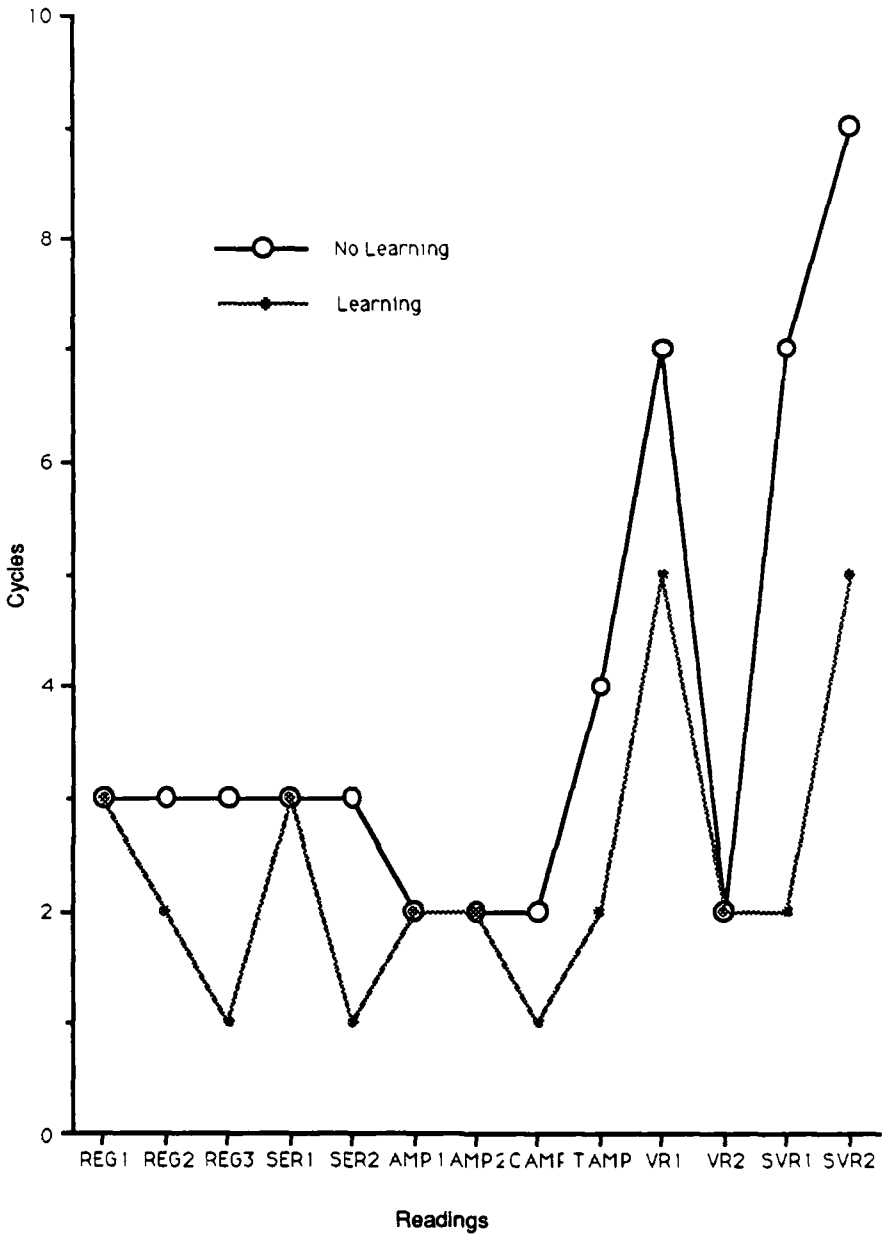


Figure 5.7. Effect of learning on cumulative cycles.

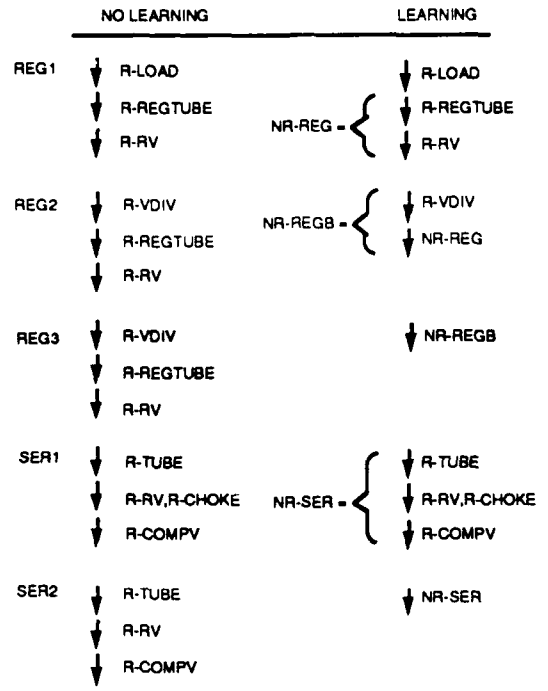


Figure 5.8. Operator trace for the regulator tube circuit and the series tube controller.

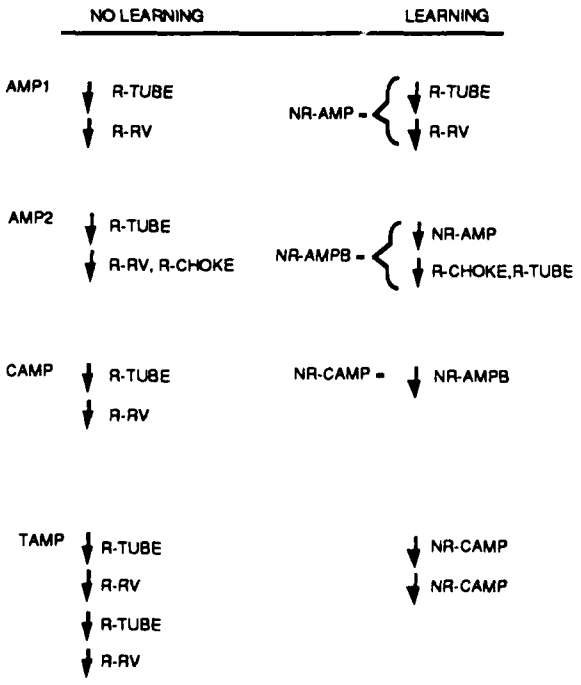


Figure 5.9. Operator trace for the triode amplifier, cathode bias amplifier, and two state amplifier.

The reading CAMP largely duplicates the behavior asserted in AMP2, while also pointing out the peculiar property of a cathode bias amplifier, i.e. its cathode bias. This bias is proved with acausal reasoning; therefore the operators used to support reasoning in AMP2, as summarized by the new operator NR-AMPB, are sufficient to process CAMP. As TAMP is a two stage amplifier, it is not surprising that reasoning there has been reduced to two application of NR-CAMP, versus the four operator cycles in the non-learning condition.

In VR1 (Figure 5.10), the sequence R-TUBE, R-RV appears twice in the non-learning condition. In the learning condition, each pair is captured by NR-CAMP. In the case of the second pair, this is not optimal since NR-SER would in fact reproduce the entire three-operator sequence R-TUBE, R-RV, R-COMPV if it had been fired before NR-CAMP, thereby eliminating one cycle and simplifying the final explanation by using only NR-SER where a hybrid of NR-CAMP and NR-SER is currently required. (As mentioned in Section 5.3, applying NR-CAMP to this part of the circuit is the result of overgeneralization.) No transfers to VR2 were expected or found.

SVR1 (Figure 5.11) is the same reading as VR1 applied to a circuit with a small but important difference, the regulator tube Tr. The entire VR1 explanation has been learned and when applied to SVR1 reduces the simulation to firing just two rules. One might expect that since SVR1 is the same text as VR1, the rule learned from VR1 should be the only one required to process the latter reading, and the appearance of NR-CAMP may seem surprising. Actually, the explanation produced for VR1 includes all but one of the propositions of that reading. It leaves out a reference to changing current which occurs halfway through VR1. The system was able to prove all subsequent propositions of VR1 without reasoning further about that changing current, therefore the change does not turn up in the explanation structure or in the rule NR-VR. When the changing current is mentioned in SVR1, the system must retreat briefly to reasoning on the basis of first principles to make up for the omission. With respect to the current system, the changing current could be described as irrelevant or a distraction. But this is not to say that the reference to the changing current is superfluous in an absolute sense. There are two explanations consistent with the VR1 reading, and one of them includes the changing current, but the system happened to find and generalize the other.

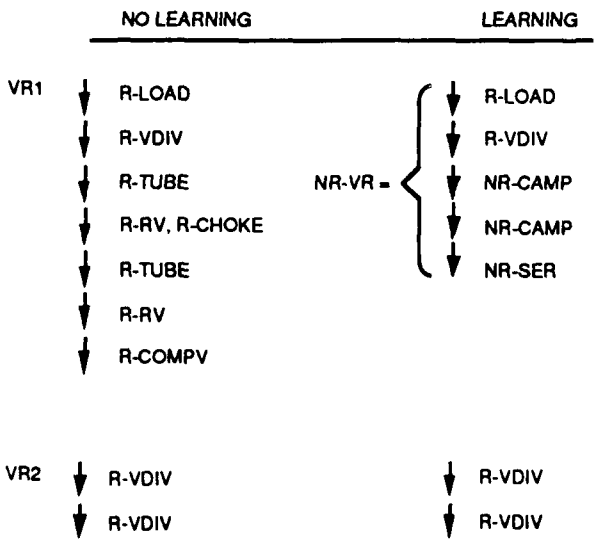


Figure 5.10. Operator trace for the voltage regulator.

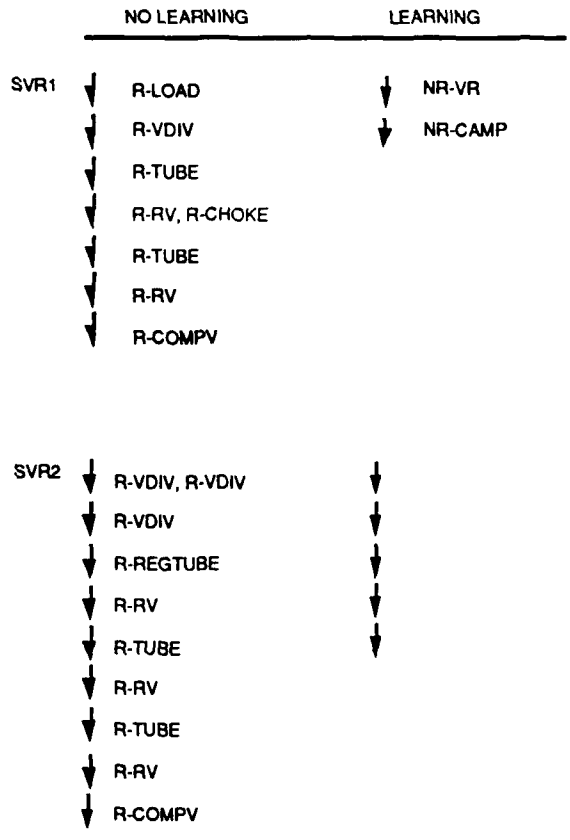


Figure 5.11. Operator trace for the stabilized voltage regulator.

SVR2 shows three cycle-saving transfer effects: NR-REGB has replaced the sequence R-VDIV, R-REGTUBE, R-RV; the following NR-CAMP replaces the sequence R-TUBE, R-RV; and the pair of new rules NR-CAMP/NR-SER has again preempted the sequence R-TUBE, R-RV, R-COMPV.

5.5 Utility of the Acquired Schemas: Reading Times

Since the circuit parse is an exhaustive search for schema instantiations, adding new schema definitions must require more parsing time. The effect is increasingly pronounced with each learned schema, and SVR1 requires 35% longer to schematize in the learning condition than in the non-learning condition. The time in CPU seconds required to parse the circuits in each condition of the experiment is shown in Table 5.2. The general deterioration reflects the fact that no attempt is made to optimize the learned structure definitions, but more importantly, the formulation of the schema instantiation process as an all-solutions search undercuts all possible gains from EBL during circuit analysis.

Table 5.2
Circuit analysis times (CPU seconds) in each condition.

	Circuit Analysis Times		
Reading	No-Learning	Learning	Improvement
REG1	50	50	0%
REG2	51	51	0%
REG3	50	53	-6%
SER1	27	36	-33%
SER2	28	30	-7%
AMP1	31	30	3%
AMP2	28	30	-7%
CAMP	36	39	-8%
TAMP	55	65	-18%
VR1	92	109	-18%
VR2	90	111	-23%
SVR1	121	163	-35%
SVR2	137	162	-18%
TOTAL	796	929	-17%

In the rest of this section, I discuss the results of the experiment in terms of the time elapsed during the text analysis phase. As with circuit analysis, addition of new rules increases the potential cost of pattern matching. But since reading is not modeled as an exhaustive search, we might hope that the computational utility of the acquired schema rules would lead to a net reduction in reading times. Table 5.3 presents the reading times in CPU seconds and they are graphed in Figures 5.12 and 5.13. In about half the readings, strong net reductions were observed. However in two important readings the system performed so much slower in the learning condition, that cumulatively, reading time increased by 5%. The problem is especially severe in VR1 and VR2, because all the learned schemas have been successfully instantiated, often leading to futile evaluation of the rule preconditions for each schema instantiation on each cycle.

Table 5.3.
Reading times (CPU seconds) in each condition.

	Reading Times		
Reading	No-Learning	Learning	Improvement
REG1	30	30	0%
REG2	28	15	46%
REG3	38	9	76%
SER1	32	35	-9%
SER2	28	11	61%
AMP1	13	16	-23%
AMP2	25	29	-16%
CAMP	34	25	26%
TAMP	71	40	44%
VR1	127	228	-80%
VR2	53	134	-153%
SVR1	135	78	42%
SVR2	1168	1226	-5%
TOTAL	1782	1876	-5%

Because the current system, like most Prolog implementations, does not include an efficient algorithm such as a rete matcher (Forgy, 1982) to minimize the cost of the additional rules, the overall picture revealed by the readings times is overly pessimistic. Looking at individual readings, it is clear that there are really two major effects on reading times. One is that successful use of a new rule can drastically reduce reading time by preempting the firing of less relevant rules. This can be seen in SVR1 (42%), REG3 (76%), and SER2 (61%). The other effect is that the new rules themselves are often irrelevant to achieving the current goal. In such cases the cost of matching them outweighs the benefits of the first effect. This happened in readings VR1 (-80%) and VR2 (-153%).

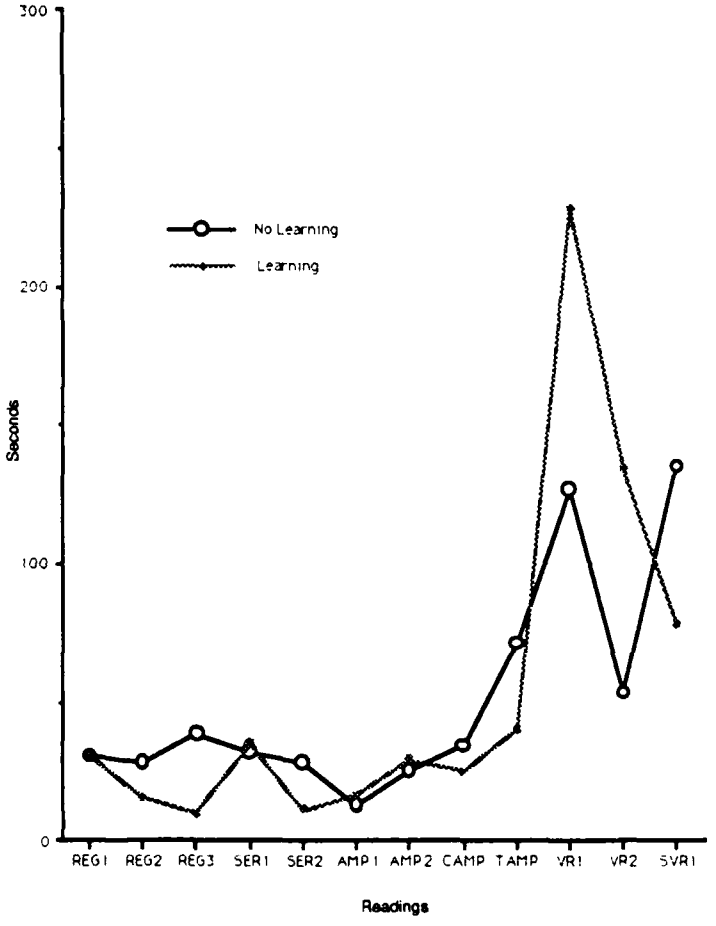


Figure 5.12. Effect of learning on reading times.

Furthermore, the new rules were slower to match than the old ones. This is partly due to the multiple instantiation problem (see Figure 5.3), since multiple instantiation of the structural definition often leads to a series of related (and expensive) queries. As an example, consider T2 in the voltage regulator. While processing reading VR1 in the non-learning condition, the system applies rule R-TUBE to T2 and generates a single query regarding the voltage between the tube's grid and cathode. In the learning condition, as a result of the two instantiations of the cathode bias amplifier, there is also a query about grid voltage relative to the potentiometer midpoint and another about grid voltage relative to ground. One might attribute this multiplication of queries to the very conservative strategy of maintaining all structure instantiations. In this case, T2 serves as principal component for instantiation of two cathode bias amplifiers and also for a series tube controller not shown in Figure 5.3. A better strategy might be to generate a preference for retaining only one of these.

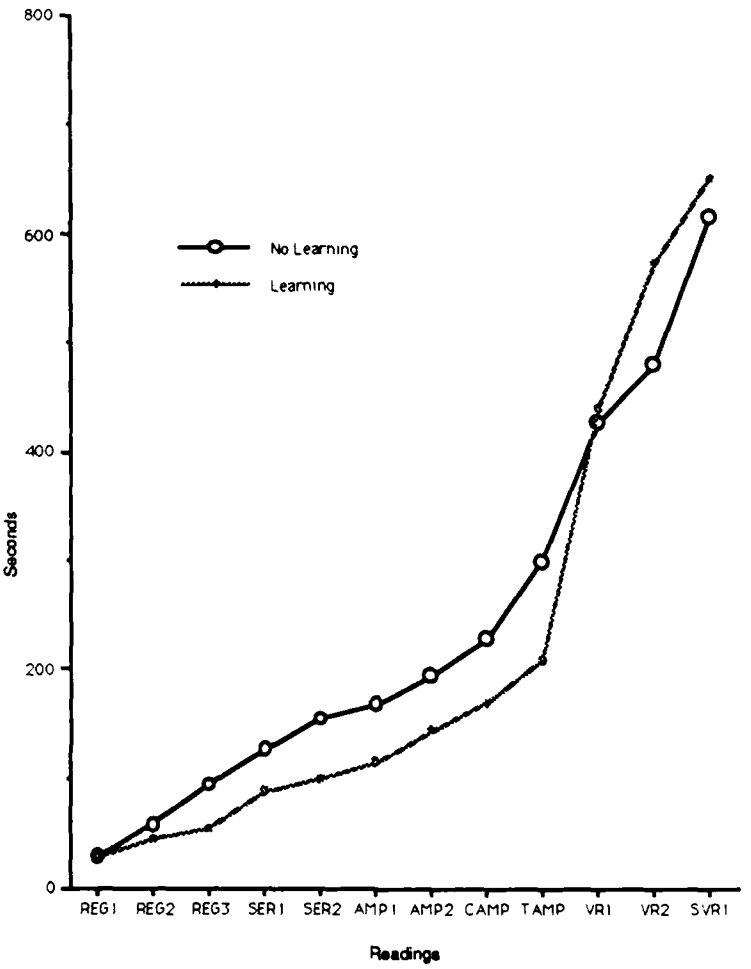


Figure 5.13. Effect of learning on cumulative reading times.

Another reason for the expense of the three new rules NR-SER, NR-CAMP, and NR-AMPB is that each of them incorporates R-TUBE, which is by far the most expensive rule in the original rule set. R-TUBE has a precondition which lends itself to a long search through transitive acausal reasoning. The ambiguity created by the multiple instantiations interacts with the high cost of the these rules to generate a number of costly, and essentially pointless queries, greatly slowing the processing.

Although the overall utility of the acquired schemas in reducing reading times was negative in this system, we cannot conclude what their net effect would be in an implementation optimized for performance. The greatest losses were due to repeatedly matching preconditions of acquired schemas. By the nature of the EBL process, this matching is highly redundant. It should be emphasized that the current system only begins to take advantage of this redundancy. With a rete-like matching architecture, the cost would be considerably reduced.

Chapter 6

Problems and Limitations

During development of the reader program, a number of its limitations became apparent. In this chapter, I discuss the most notable ones. The first relates to reasoning about circuits at different levels of abstraction. By learning schemas, the system is able to reason about circuit behavior in terms of large subcircuits rather than individual components. In Section 6.1, I will show that the readings presented here did not always take advantage of this.

In using EBL for learning concepts as complex as circuit schemas, it has become apparent that despite the technique's broad applicability, it does have certain limitations, which I will discuss in Section 6.2. This became particularly noticeable when I tried to apply EBL to the multi-state simulations required for analyzing A.C. circuits (Section 6.3). Finally, work on the domain theory has shown that practical electronics is by no means a straightforward, easily implemented theory. I will present some of the subtleties of reasoning in the domain in Section 6.4.

6.1 Treatment of Acquired Schemas: Glass Box vs. Black Box

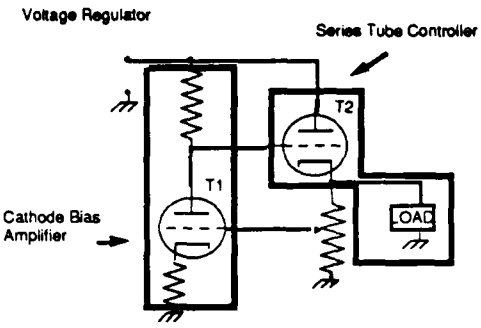
The schemas used to organize the electronics domain are hierarchical. The concepts lowest in the hierarchy correspond to single components while the highest may describe a large subassembly of parts. It is therefore possible to reason about a given circuit at different levels of abstraction. That is, if the cathode bias amplifier or the series tube controller really constitute useful subcircuits, then it ought to be possible to reason about them in terms of inputs and outputs, ignoring the details of intermediary behaviors; but many of the readings violate this principle.

As an example of a reading which delves into details, consider the component-based explanation taken from the test suite shown in Figure 6.1. In lines 4 and 6, it mentions two events that take place strictly inside the major subcircuits. The change in current through T1 is a detail about how the cathode bias amplifier works, and the changing resistance of the series control tube T2 is taken from the middle of the series controller tube reading. By citing these events, the reading is reasoning in terms of individual components rather than in terms of major subcircuits. Thus the schematic subcircuits have to be treated as glass boxes with internal events available rather than as black boxes.

The explanation on the right shows how a few modifications can correct this. The two events in lines 4 and 6 are deleted, but since the change in grid voltage of T2 corresponds to the output for the amplifier, line 5 remains. Likewise, changing output voltage is part of the function of the series tube controller, so mentioning it in line 7 is consistent with reasoning about the circuit at the more abstract level.

Thus one guideline that could be suggested for instructional prose in this domain is that an explanation should be crafted in terms of black boxes, i.e. the most abstract schemas that the student is supposed to know. More specific schemas will amount to rehashing the internal behavior of devices the student has already understood.

In addition to the voltage regulator reading, other readings in the test suite fail to meet this principle, and I therefore designed the system to support comprehension of these sub-schema digressions. The interior behaviors of the acquired schema are saved so that later discussion of them can be matched to the learned rule. The systems performs well whenever a later passage happens to match the argument of the training instance. However, suppose that a given domain theory supports more than one explanation for the device's behavior. If the rule is formed on the basis of one explanation and a subsequent reading sketches the alternate rationalization, then the system will be unable to match any claim where the accounts diverge. Instead, it will be reduced to rederiving the behavior of the device from first principles. This effect was observed in SVR1 (see Section 5.4).



	Component-Based	Black-Box Based
1	If more current begins to flow through the load,	If more current begins to flow through the load,
2	the output voltage begins to decrease.	the output voltage begins to decrease.
3	This makes the voltage on the grid of T1 more negative relative to the cathode	This makes the voltage on the grid of T1 more negative relative to the cathode.
4	causing less current to flow through T1.	DELETED
5	The grid of T2 then becomes less negative,	The grid of T2 then becomes less negative,
6	decreasing the resistance of T2,	DELETED
7		ADD: which tends to decrease the output voltage
8	and thus keeping the output voltage the same.	and thus keeps the output voltage the same.

Figure 6.1. Explanations and schema abstraction. The black-box based reading never refers to the behaviors within a schematic subcircuit.

An alternate way of handling intra-schema behavior is to treat claims about internal behaviors differently. For instance they could be ignored, provided that the balance of the passage was verified. Or perhaps such material is a source of difficulty both for people and EBL-based systems and should be deleted by the author. If the readings followed the black box principle, considerably less would need to be stored in the rules, since internal behavior would no longer be included. Perhaps more importantly, if learned rules did not add internal behaviors to the data base, time required to compute their implications during simulation would be saved.

6.2 Limitations of EBL

This work has been based on the decision to use only EBL in modeling transfer of knowledge during reading and as a result, we are in a position to identify some of possible limitations for the application of EBL to problems similar to those treated here. First and foremost, EBL produces rules which are redundant with the domain theory. In a search for a single solution, this redundancy may lead to a faster solution. But in a computation of all solutions, there can only be a deterioration in performance through addition of redundant operators. Efficient pattern-matching can limit this problem, but not eliminate it. This sort of all-solutions approach has seemed a natural approach to parsing electronic circuits, but it is a style of computation where EBL cannot confer a computational advantage. The disutility of redundant learning when goal failure is expected has also been documented in Markovitch and Scott (1989) which suggests using derived rules only when a goal is expected to succeed.

A closely related limitation in the applicability of EBL is how it can disrupt search for an optimal solution. In this project, reading comprehension has been modeled as proof completion. This becomes complicated as soon as the system can produce more than one explanation for a given text assertion. The explanations may differ in several ways. They may refer to different pieces of circuit structure and behavior, leading to significantly different schemas and simulation rules depending on which explanation is used as input to the EBL generalizer. Not all explanations are equally good. When reasoning is heuristic, a shorter explanation is generally more reliable than a longer one. A common way of implementing this is to choose the briefest explanation, i.e. the first one produced by a breadth first search.

Unfortunately, this design is inherently at odds with the use of rules obtained from macro-creating techniques such as EBL. Consider Figure 6.2. In the original problem solving that supported acquisition of a new rule (Figure 6.2 a) a solution three cycles in length was found. In subsequent problem solving (Figure 6.2 b), the new rule is successfully used on the first cycle, but a solution two cycles in length would have been found if search had continued. This shorter solution has been hidden by the new rule, which although it requires only one cycle to apply, is based on three cycles of original reasoning and therefore suboptimal.

This effect can also interfere with the use of breadth-first expansion to simulate time. When this approach is used, circuit behavior can be compared to the effect of a stone thrown in a pond where ripples of change extend uniformly in all directions (i.e. along all possible causal paths). Learned rules can wreck this metaphor by introducing a set of changes that races ahead of other developments in the circuit. If there is some interaction between the different effects, introduction of EBL-acquired rules may actually change the nature of what other behavior can be inferred from the circuit. This is surprising since generally one would like to assume that introduction of redundant rules cannot change the inferential behavior of the system; but under these circumstances, EBL has just this effect.

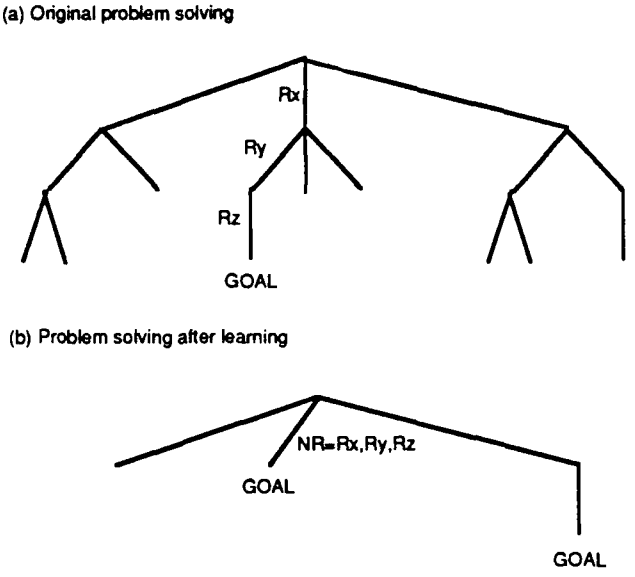


Figure 6.2. EBL and search for optimal solutions.

These last two problems are very similar. They both assume that the size of the solution is relevant, in one case because short solutions are more reliable and in the second case because steps in the solution correspond in a general way to the passage of time. Once macro operators are introduced, it is no longer clear how to judge the size of solutions using them.

One approach to solving these problems is to abandon the commitment to learning a single rule for each schema structure's behavior and instead use EBL to learn or refine knowledge about when to use individual operators, i.e. by refining the preconditions of individual operators. As seen in systems like LEX (see Section 2.3), this approach uses problem solving traces to better define the conditions under which a single operator is likely to lead to a solution. Since learning is confined to control knowledge, the number of operators required for a solution does not change. The drawback to this approach is that it may conflict with our intuitions about what a student knows after reading an explanation of a voltage regulator. In a LEX-type approach, the system would improve in its application of first principles in the context of a voltage regulator, and therefore may end up reasoning more efficiently about a voltage regulator in the future. But it learns no single rule or concept that describes how a voltage regulator works and this seems inconsistent with the goal of instruction in the practical electronics domain.

A final limitation on the technique involves the complexity of the search spaces to which EBL is applied. Ideally, EBL operates in a large homogeneous search space in which a great deal of search is performed to construct fairly long solution paths and the computational effort of reproducing it is replaced by the firing of a single new macro rule. In contrast, a macro rule derived from a short solution path can replace only a modest amount of computing. One of the striking aspects of this work has

been that searches through the operator space in this domain have typically produced solutions one and two operators deep, giving EBL only limited potential for improving processing.

The short solution paths reflect the fact that the operator space is only part of the problem solving required to process a reading. A good deal of the important inferencing that supports the short operator search is done during the circuit parsing routine which precedes text processing. In all the simpler circuits, this pre-reading phase requires more time than the actual text processing, but due to the exhaustive nature of the circuit-parsing computation, EBL is not applied to circuit parsing. Another large portion of the problem-solving is search among backward-firing rules. When they occur in the explanation structure, EBL includes them in the single new operator derived from the explanation. Hence, the operator includes a composition of some backward-firing rules, but this composition is not available independently of the new operator. It is used only to satisfy a goal generated by the new operator. When any other operator or backward-firing rule generates that goal, the system resorts to a less focused search through the original domain theory.

6.3 Combining EBL and Multi-State Simulations

Although EBL worked well when applied to the D.C. circuits of the last chapter, I discovered serious problems when I tried to extend the learning technique to cover certain A.C. circuits. This has important implications of the applicability of EBL and suggests that EBL may not be suited to some domains.

At the beginning of this project, I planned on demonstrating the feasibility of the reader/lerner program by using it to acquire all the basic circuits needed for a radio. I began working toward that goal by focusing on tuned circuits and oscillators, shown in Figure 6.3. The instructional materials I have worked with (VanValkenburgh, Nooger, and Neville; 1966) analyze the tuned circuite in order to explain the basic mechanism of oscillation, neglecting any losses due to the resistance of the coil and wire. They then point out these losses and explain the necessity of feeding the signal back into the tuned circuit to maintain oscillations, as in the Armstrong oscillator. The behavior of each circuit is traced out in detail, and familiarity with the first passage is clearly considered a prerequisite to comprehending the second passage.

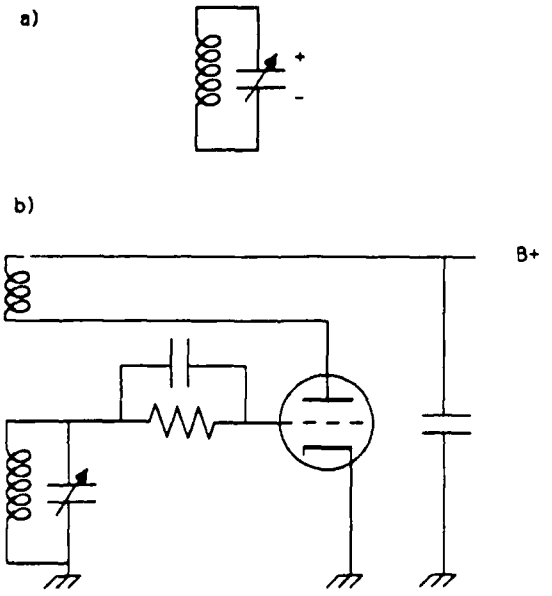


Figure 6.3. A tuned circuit (a) and the Armstrong oscillator (b).

Simulating these circuits is considerably more complicated than simulating D.C. circuits. D.C. circuits respond to change with a simple qualitative increase or decrease in output. whereas the tuned circuits generate signals. It was nevertheless possible to simulate them by representing the signal as a sequence of qualitative state as described in Section 4.7. An earlier version of the reader/learner model used such a simulation of the tuned circuit to devive a rule predicting its behavior. This rule successfully predicted the behavior of the tuned circuit during ismulation of the Armstron oscillator, reducing both the number of cycles and the CPU time required for simulation. On the basis of those results, I assumed that EBL would apply to both D.C. ad A..C. circuits.

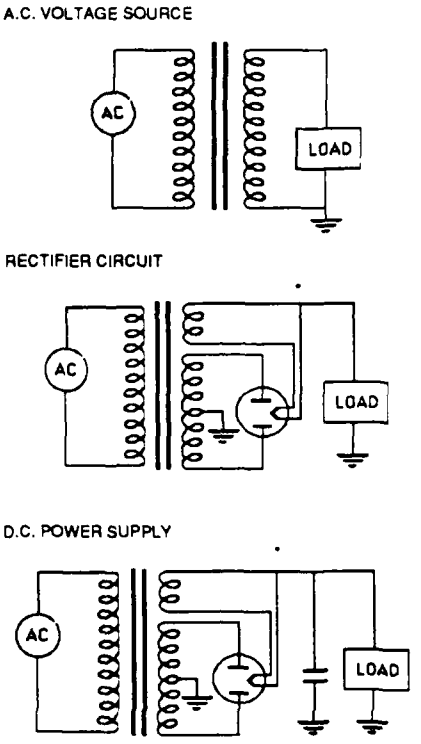


Figure 6.4. Three structurally related A.C. circuits.

I next turned to power supplies. another important radio component. The instructional materials contained a sequence of three increasingly complex circuits designed to gradually lead a student to an understanding of a D.C. power supply. That sequence is shown in Figure 6.4 and consists of an A.C. power supply, a full-wave rectifier circuit, and finally, the D.C. power supply itself. These three circuits have very similar structure. The second includes the first as a subcircuit and the third includes the second. Intuitively the behavior of the circuits is also closely related. This can be seen by inspecting the (qualitative) wave forms they generate as shown in Figure 6.5. Each wave form shows the circuit's output voltage. The rectifier inverts the negative portions of the A.C. wave, making the voltage always positive. The capacitor of the power supply smooths the wave leaving a relatively modest ripple. Each of these circuits prepares the way for analysis of the next circuit and actual instructional materials discuss and compare all three. Since studying this sequence in order is supposed to facilitate human learning, I assumed they would also lead to a learning effect when processed by the reader/learner system.

Like the oscillator, these three A.C. circuits were simulated using the multi-state techniques of Section 4.8, and resulted in the qualitative A.C. signals shown in Figure 6.5. The explanations produced by the simulation wre prefectly satisfactory and would have been adequate input to the EBL procedure. However, it was clear that a rule obtained from the second circuit could not apply to the third circuit despite their similarity. This can be seen in Figure 6.5. In isonlation, the rectifier circuit produces pulsating D.C. at thits output (marked * in Figure 6.4), whereas in the D.C. power supply, this same voltage is a ripple. If the system learns to predict a pulsating pattern from analysis of the second circuit, then it has constructed a rule that either fails to apply to tht third circuit, or does apply but makes an incorrect prediction.

In these two circuits, the rectifier subcircuit is not consistently associated with a single behavior. Instead, the capacitor interferes with it, so that it generates a new behavior. This points out the basic limitation of the model of transfere that underlies the system and to some extent EBL itself. Generally I have assumed that a circuit which exhibits some behavior in isolation will exhibit the same behaior when it appears in combination with other components. In that case, the behavior of a complex device can be derived by *composing* the behavior of its subcircuits. Let us cal lthis the composability assumption.

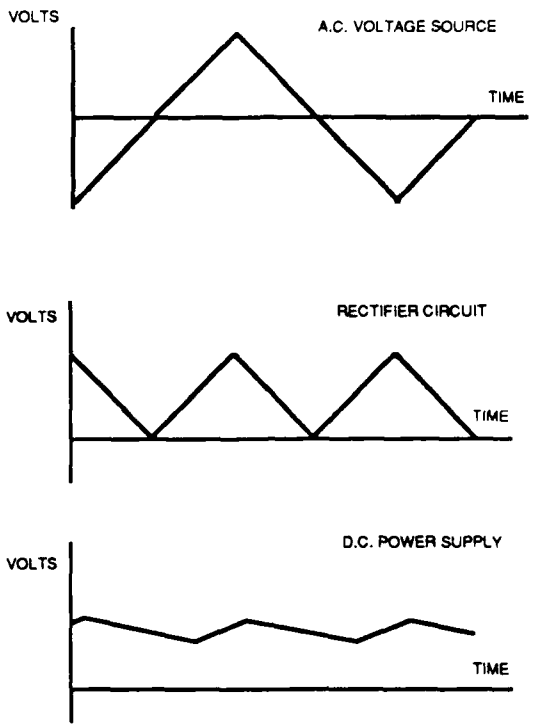


Figure 6.5. Output voltage for three A.C. circuits.

Each circuit in the test suite satisfies the coposability assumption. This is probably due to the fact that all the circuits of the test suite could be simulated in a single state. Single-state simulations are shorter than multi-state simulations and offer less chance for interactions among the componenets. Whithout these interactions, the behavior of the components can be determined locally so that the same structure produces the same behavior regardless of its environment, thus justifying the composability assum[tion.

Clearly the non-composable A.C. circuits have something in common, but for a student to benefit from this similarity may require tha ability to recall the behavior of the earlier circuit and then reason about how the addition of other sturctures, such as the capacitor in the third circuit, leads to a modification of it. The textbook graphic of the third circuit (Figure 4.16) invites the reader to do some analysis along these lines by overlaying the output of the third circuit with the output of the second circuit, shown as a dotted line.

In summary, all D.C. circuits investigated here gave rise to useful EBL-generated rules. Furthermore, at lease some A.C. circuits can be expected to generate rules which will transfer successfully to more complex devices, but more importantly, there are clearly some A.C. circuits whose behavior cannot be caputred in this manner.

6.4 The Subtleties of Practical Electronics

The domain theory used here is based on a novice understanding of practical electronics. This lead to a number of problems. One of the most subtle concepts dealt with by common-sense electronics is feedback. In this system, feedback is represented only in terms of the concept "constant". Keeping some voltage constant in the face of variation in other voltages is a common theme in several of the readings. Intuitively, the concept of feedback, which in the readings is always negative, is a causal chain that doubles back on itself with a negation of its first proposition. Thus, if a decreasing voltage at some point initiates a series of changes leading up to an increasing voltage at that same point, we have established a pattern of negative feedback and can argue that voltage between the two points will be constant. However, formally we have had to sanction a contradiction as being meaningful. This means we cannot include in the architecture the general assumption that contradictory propositions may not be added to the database!

Another subtle point is the use of non-monotonic reasoning which is especially clear in connection with the mythical fixed-voltage battery. A useful approximation in common-sense electronics is that the voltage across a voltage source is fixed. It is not always justified, however. The voltage regulator readings VR2 and SVR2 assume as an initial perturbation that the source voltage changes. To provide for this, some of the rules have taken on a nonmonotonic flavor. Thus rather than basing some inference on the requirement that a voltage source be present, they may require both a voltage source and no assertion that its voltage has changed. A piecemeal approach to non-monotonic logic has proved adequate with respect to the reading suite examined here, but has probably slowed development.

Chapter 7

Conclusions

The goal of this project has been to test the feasibility of automated reading as a knowledge acquisition method for technical domains such as electronics. This is a promising approach for several reasons. It offers the possibility of using existing technical literature as a knowledge source to minimize the burden of knowledge engineering. Furthermore, the development of qualitative reasoning means robust general domain theories will be available for technical domains sooner than for most other knowledge domains. Finally, given the availability of a domain theory and the prevalence of explanation as a way of conveying technical concepts, these fields seem to offer a chance to apply EBL with almost no need for inductive techniques.

7.1 Conclusions Relating to EBL

1. The system used EBL to acquire a set of circuit schemas from the practical electronics domain and was able to use those acquired concepts in comprehending new, more complex circuits. These schemas captured the intuitive content of the instructional materials, and their role in enhancing comprehension of subsequent passages has been documented by the resulting reduction in simulation cycles.

2. The computational benefits of using EBL depend critically on efficient pattern-matching to overcome the effects of increased processing as more rules are learned.

3. Integrating the macro-like rules generated by EBL into more primitive domain theory reasoning can raise a number of technical problems by making the significance of solution length unclear.

7.2 Conclusions Relating to the Practical Electronics Domain

1. Domain theory alone is not sufficient to eliminate all ambiguity from explained concepts. Partly this is because the domain theory is bound to be approximate and therefore will lead to overgeneralized concepts. In addition there are constraints beyond the domain theory which reflect the fact that the circuit is a designed entity. This knowledge of design goals is typified by two broad principles for learning device structure: constraints that were satisfied by a single component in the training instance should also be satisfied by a single component in the later instances; and constraints satisfied by distinct components in the training instance should be satisfied by distinct components in later instances.

2. Acquired knowledge of a single new complex concept such as a category of circuits should not be represented by addition of just one rule. Rather, reasoning with complex concepts almost certainly involves searching through more than one problem space and new rules should be learned for each space. The domain theory used here was very simple, but it included three spaces: one all-solutions, forward-chaining search for schema instantiations, a forward-chaining application of operators, and a backward-chaining search for acausal inferences. The new rules represented problem solving in the last two spaces and were integrated only in the second of the two, so that problem solving improvements could only be realized there.

Extending the system to handle multi-state simulations makes this problem more acute since the sort of envisionment approach outlined in Chapter 4 creates additional search spaces due to the distinction between the knowledge used to elaborate the implications of any one state, which is domain-specific, and the knowledge used to reason about state transitions, which is largely domain-independent. Thus, rather than limit the new concept to be incorporated into just one search, it can be represented by acquisition of many new rules, one for each goal achieved in a homogeneous subspace.

3. The chief bottleneck in developing systems such as the present one is not the learning mechanism, but the domain theory. Initially, I assumed that a very simple domain theory would be sufficient to understand the fairly simple circuits I was working on, since the reading describing the circuit's behavior should allow the system to disambiguate an approximate theory. Furthermore, in hopes of also using the system as a cognitive model, I wanted to restrict the domain theory to the typical student's understanding of basic electronics. This restriction made it impossible to simply adopt qualitative reasoning theories pioneered by other researchers since those theories rely on some rather sophisticated heuristics and use of a truth maintenance system.

4. The link between the structure and function of electronic circuits is complex. The use of EBL in this project assumed that the behavior of a complex circuit can be computed by composing the behavior of familiar subcircuits. The attempt to extend the system to A.C. circuits such as the D.C. power supply has shown that this assumption is not always justified.

Nevertheless I have found it necessary to incorporate more and more of the rigor of qualitative reasoning to ensure the validity of the simulations. The reading was not especially valuable as a means of eliminating ambiguities and in retrospect, beginning the project with a full-fledged qualitative reasoning system like QSIM (Kuipers, 1984) might have sped development of the domain theory.

In many ways, this work has confirmed a finding generally occurring in educational applications of AI: it is more difficult to represent a student's understanding of a domain than to represent the expert's understanding. An example of this is the distinction between negative feedback and contradiction. For a student, negative feedback is usefully approximated by a change which ultimately causes a countervailing change, as when the increasing output voltage of a voltage regulator leads to a decrease in that same voltage. But sometimes this pattern occurs where no feedback is intended, in which case it represents a contradiction. The student has no principle for distinguishing these two interpretations. Another difficulty in developing the theory has been nonmonotonicity, discussed above.

7.3 Conclusions Relating to Instruction

Because an automated reading system must address some of the same problems a human reader faces, there is some potential for using the system as a basis for modeling human learning from text. Equipped with such a model, it might be possible to develop authoring aids able to critique explanatory prose in terms of its content. This project offers such a model for the specific case of instructional readings from the practical electronics domain. Electronics has a fairly small, fairly complete domain theory. The remaining content of the field can be organized as a hierarchy of higher-level concepts, each of which represents an application of the domain theory. More generally, the current EBL-based approach could be applied to any domain in which a small, more or less complete domain theory is presented prior to study of a series of increasingly complex applications of that theory. This may include such fields as math, geometry, chemistry, and engineered systems. Given a domain of this type, instruction can naturally be developed around a series of hierarchically related explanations.

Future research may show the learning processes of the computer system are similar to those at work in human learning from such instruction. We may venture that if the performance of the computer system corresponds to human learning, then it suggests the following guidelines for the design of explanatory text:

1. The hierarchy should be presented systematically. Development of a concept should follow development of the lower-level concepts on which it depends.

2. Each explanation should present one higher-level concept from the hierarchy. This enables the student to identify the concept to be learned. Another system making use of this principle while learning from examples is VanLehn's Sierra (1987).

3. Each part of the example for a concept should be included in the explanation to ensure its inclusion in the acquired concept. Explanations that omit important components give rise to overgeneral schemas (see Figure 5.4) which hurt comprehension by allowing the associated behavior rules to fire inappropriately.

4. Assist the student to correctly instantiate the acquired schemas, for example, by identifying instances graphically. This is another chance to avoid the problems resulting from parsing with overgeneral schemas.

5. Once a higher-level concept has been acquired, avoid delving into the lower level details that originally justified it. Experience with the current system suggests that previously learned schemas should be black boxes; materials used here were sometimes glass box explanations, resulting in lower efficiency for the automated reader. Explanations based on black box schemas should also be easier for human readers.

REFERENCES

American Radio Relay League, (1961). *Radio Amateur's Handbook*, West Hartford, Conn.

DeJong, G. F. and Mooney, R. (1985). "Learning Schemata for Natural Language Processing," Proceedings of IJCAI-85. Morgan Kaufman, Los Angeles.

DeJong, G. F. and Mooney, R. (1986). "Explanation-Based Learning: An Alternative View," *Machine Learning* **1**, 2, 145-176.

DeKleer, J. (1984). How Circuits Work. *Artificial Intelligence*, **24**, 205-280.

Ellman, T. (1985). "Generalizing Logic Circuit Designs by Analyzing Proofs of Correctness," Proceedings of IJCAI-85. Morgan Kaufman, Los Angeles.

Forgy, C. (1982). Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, **19**, 17-37.

Kieras, D. and Mayer, J. (1989). Explanation-Based Knowledge Acquisition of Electronics. Proceedings of the Workshop on Models of Complex Human Learning. Cornell University, Ithaca.

Kuipers, B. (1984). Commonsense Reasoning about Causality: Deriving Behavior from Structure. *Artificial Intelligence*, **24**, 169-203.

Laird, J. E., Newell, A., and Rosenbloom, P.S. (1987) Soar: An architecture for general intelligence. *Artificial Intelligence*, **33**, 1-64.

Lebowitz, M. (1986). "Integrated Learning: Controlling explanation." *Cognitive Science*, **10**, 2, 219-240.

Markovitch, S. and Scott, P. (1989) "Utilization FIltering: a method for reducing the inherent harmfulness of deductively learned knowledge," Proceedings of IJCAI-89. Morgan Kaufman, Detroit.

Mahadevan, S. (1985). "Verification-Based Learning: A Generalization Strategy for Inferring Problem-Decomposition Methods," Proceedings of IJCAI-85. Morgan Kaufman, Los Angeles.

Mitchell, T. (1983). "Learning and Problem Solving." Proceedings of IJCAI-83. Morgan Kaufman, Karlsruhe, West Germany.

Mitchell, T. M., Keller, R., and Kedar-Cabelli, S. (1986). "Explanation-Based Generalization: A Unifying View," *Machine Learning* **1**, 1 47-80.

Mitchell, T. M., Mahadevan, S., and Steinberg, L.I. (1985). "LEAP : A Learning Apprentice for VSLI Design," Proceedings of IJCAI-85. Morgan Kaufman, Los Angeles.

Mostow, J. (1983) "A Problem Solver for Making Advice Operational," Proceedings of the National Conference on Artificial Intelligence, 1983. Morgan Kaufman, Washington, D.C.

Moon C.E., and Lytinen, S.L. (1989)'The function of examples in learning a second language from an instructional text," 11th Annual Conference of the Cognitive Science Society, 1989. University of Michigan, Ann Arbor, Michigan.

O'Rorke, P. (1984). "Generalization for Explanation-Based Schema Acquisition," Proceedings of AAAI-84. Morgan Kaufman, Austin.

Pazzani, M. (1988). "Integrated Learning with Incorrect and Incomplete Theories," Proceedings of the Fifth International Conference on Machine Learning. Morgan Kaufman, Ann Arbor.

Segre, A., and DeJong, G. (1985). "Explanation Based Manipulator Learning: Acquisition of Planning Ability Through Observation," Technical Report, AI Research Group Working Paper 62, University of Illinois at Urbana-Champaign.

Stallman, R. and Sussman, G. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis, *Artificial Intelligence*, **9**, 135-196.

VanValkenburgh, Nooger, and Neville, Inc. (1955). *Basic Electronics*. Hayden, Rochelle Park, NJ.

Winston, P. (1982). "Learning New Principles from Precedents and Exercises,"*Artificial Intelligence* ,**19**, 321-350.

Winston, P., Binford, T., Katz, B., and Lowry, M. (1983). "Learning Physical Descriptions From Functional Definitions, Examples, and Precedents." AAAI-83.

APPENDIX 1

OPERATORS OF THE DOMAIN THEORY

Explanation of Prolog formulas used in the domain theory

Prolog Formulas	Meaning
bcircuit(Circuit_Path)	Circuit_Path is a biased circuit. (see Section 4.3)
bcomp(Name,Type,List-of-Ports)	There is a biased component with name, type, and ports.
change(Trend,Quantity)	Quantity is changing as indicated by Trend. Trend is either increase or decrease.
opp_change(Trend2,Trend1)	Trend1 and Trend2 are opposite.
current(P1,P2)	the current between points P1 and P2
voltage(P1,P2)	the voltage between points P1 and P2
resistance(P1,P2)	the resistance between points P1 and P2

Summary of operators

R-RES - The Resistor Rule (Ohm's Law):

A change in the current through a resistor causes the same change (i.e. increase or decrease) in voltage across it.

If +(i),
then +(v)

```
bcomp(R,resistor,[P1,P2]),
change(C,current(P1,P2))
        ->      change(C,voltage(P1,P2)).
```

R-RV - The Resistance/Voltage Rule:

A change in the resistance of any component or subcircuit causes the same change in voltage across it, unless the component is connected in parallel with a voltage source (battery), which would hold the voltage constant.

If +(r),
then +(v)

```
bcomp(X,_,[P1,P2]),
change(C,resistance(P1,P2)),
not(bcomp(VS,voltage_source,[P1,P2]))
        ->      change(C,voltage(P1,P2)).
```

R-CHOKE - Changing Resistance in a Series:

A change in the resistance of one series element causes an opposite change in the current through each of the elements.

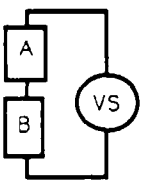
If +(rB),
then -(iA),
     -(iB),
     -(iC)


```
bcomp(X,_,[P1,P2]),
change(C,resistance(P1,P2)),
circuit(Circuit_Path),
member([P1,P2],Circuit_Path),
member([P3,P4],Circuit_Path),
opp_change(C,OC)
        ->      change(OC,current(P3,P4)).
```

R-COMPV - Compensating Voltage Changes I:

A change in the voltage across one half of a voltage divider causes an opposite change in the voltage across the other half if the voltage divider is connected in parallel with a fixed voltage source.

If ~+(vVS),
   +(vA),
then -(vB)



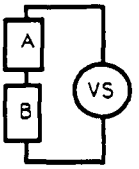

```
bcomp(VSPL,voltage_divider,[HI,MID,LO]),
bcomp(VSRC,voltage_source,[HI,LO]),
change(C,voltage(HI,MID)),
not(change(C,voltage(HI,LO))),
opp_change(C,OC),
        ->      change(OC,voltage(MID,LO)).
```

~ similarly for change(C,voltage(MID,LO))

R-COMPV - Compensating Voltage Changes II:

A change in the voltage across one half of a voltage divider causes an opposite change in the voltage across the other half if the voltage divider is connected in parallel with a "fixed" voltage source. This can be true even when voltage across the source is changing, provided the change within the voltage divider is greater.

If +(vVS),
  +(vA),
  +(vA) > +(vVS),
then -(vB)



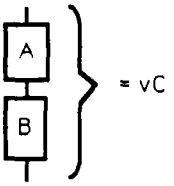

```
bcomp(VSPL,voltage_divider,[HI,MID,LO]),
bcomp(VSRC,voltage_source,[HI,LO]),
change(C,voltage(HI,MID)),
change(C,voltage(HI,LO)),
greater(change(C,voltage(HI,MID)),change(C,voltage(HI,LO))),
opp_change(C,OC),
        ->      change(OC,voltage(MID,LO)).
```

~ *similarly for change(C,voltage(MID,LO))*

R-VDIV - The Voltage Divider Rule:

A change in voltage between the outer terminals of a voltage divider causes the same change in voltage across each half.

If +(vC),
then +(vA),
  +(vB)

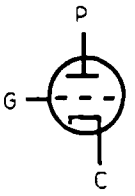



```
bcomp(VSPL,voltage_divider,[HI,MID,LO]),
change(C,voltage(HI,LO))
        ->      change(C,voltage(HI,MID)).
~ similarly for change(C,voltage(MID,LO))
```

R-TUBE - The Vacuum Tube Rule:

A change in grid voltage causes an opposite change in the tube's resistance. The second change is proportionally greater than the first, i.e. the tube amplifies.

If +(vGC),
then -r(PC),
-(rPC) > +(vGC)

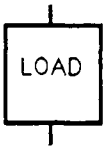



```
bcomp(VT,vacuum_tube,[PLATE,GRID,CATHODE]),
change(C,voltage(GRID,CATHODE)),
opp_change(C,OC),
        ->      change(OC,resistance(PLATE,CATHODE))
                greater(change(OC,resistance(PLATE,CATHODE))),
                        change(C,voltage(GRID,CATHODE))).
```

R-LOAD The Load/Current Rule

A change in current through the load causes an opposite change in voltage across it.

If +(i),
then -(v)




```
bcomp(L,load,[P1,P2]),
change(C,current(P1,P2)),
opp_change(C,OC)
        ->      change(OC,voltage(P1,P2)).
```

R-REGTUBE The Regulator Tube Rule

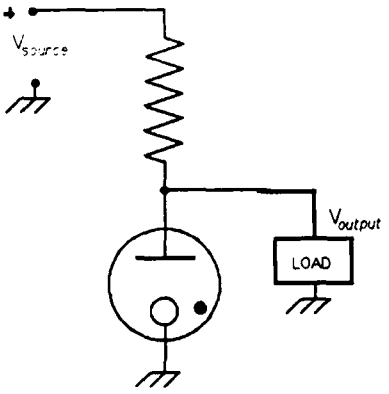
A change in current through a regulator tube causes an opposite change in its resistance.

If +(v),
then -(r)

```
bcomp(L,regulator_tube,[PLATE,CATHODE]),
change(C,voltage(PLATE,CATHODE)),
opp_change(C,OC),
        ->      change(OC,resistance(PLATE,CATHODE)).
```

APPENDIX 2
THE READINGS - TEXT AND PROPOSITIONS

READING 1. REGULATOR TUBE CIRCUIT (REG)



REG1.
If the load current increases, the voltage across the tube (Voutput) will start to decrease, which immediately increases the resistance of the tube, bringing Voutput back up. Therefore, in this circuit, Voutput is the same regardless of the fluctuation in Vsource or the load current.

```
change(increase,curr(loada,loadb,[[load,loada,loadb]]))]
change(decrease,vage(rtplate,rtcathode))
change(increase,resistance(rtplate,rtcathode,[[rt,rtplate,rtcathode]]))
change(increase,vage(loada,loadb))
constant(vage(loada,loadb))
```

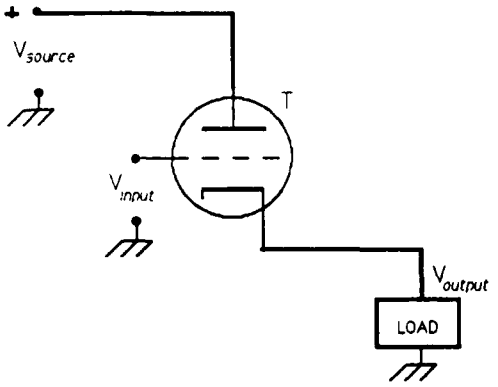
REG2.
Likewise, if Vsource goes up, the voltage across the tube (Voutput) will also start to increase. However, the resistance in the tube immediately decreases, which brings down Voutput.

```
change(increase,vage(vspos,ground))
change(increase,vage(rtplate,rtcathode))
change(decrease,resistance(rtplate,rtcathode,[[rt,rtplate,rtcathode]]))
change(decrease,vage(loada,ground))
```

REG3.
Conversely, if Vsource drops, the resistance in the tube goes up, which brings up Voutput.

```
change(decrease,vage(vspos,ground))
change(increase,resistance(rtplate,rtcathode,[[rt,rtplate,rtcathode]]))
change(increase,vage(loada,ground))
```

READING 2. SERIES TUBE CONTROLLER CIRCUIT (SER)



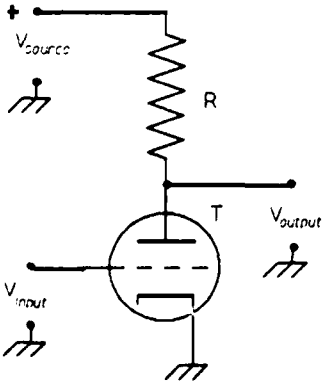
SER1.
If Vinput increases, the grid of T becomes less negative and so the resistance of T decreases causing Voutput and the current through the load to increase.

```
change(increase,vage(stc_in,ground))
change(increase,vage(stc_in,ground))
change(decrease,resistance(triplate,tricathode,
     [[_,triplate,tricathode]]))
and([change(increase,vage(loada,ground)),
     change(increase,curr(loada,loadb,[[load,loada,loadb]]))])
```

SER2.
Likewise, if Vinput decreases, the resistance of T increases, and Voutput decreases. Thus, the input voltage controls the output voltage.

```
change(decrease,vage(stc_in,ground))
change(increase,resistance(triplate,tricathode,
     [[_,triplate,tricathode]]))
change(decrease,vage(loada,ground))
```

READING 3. TRIODE AMPLIFIER (AMP)



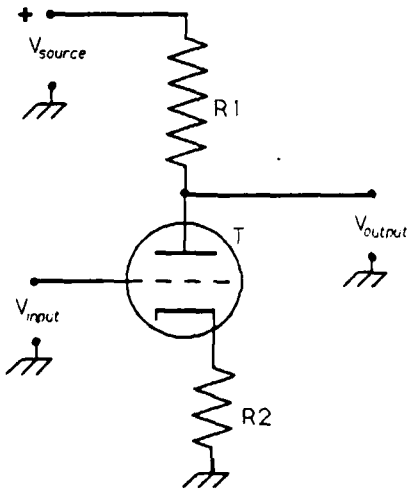
AMP1.
When Vinput is increased, the grid in the tube becomes less negative, which lowers the resistance of the tube. This causes Voutput to drop.

```
change(increase,vage(amp_in,ground))
change(increase,vage(tri_grid,ground))
change(decrease,resistance(tri_plate,tri_cathode,
     [[_,tri_plate,tri_cathode]]))
change(decrease,vage(amp_out,ground))
```

AMP2.
If Vinput decreases, the grid becomes more negative, and the tube's resistance increases, so Voutput rises. Thus the output voltage changes in the opposite direction as the input voltage. Because a small change in grid voltage can produce a large change in the tube current, the change in Voutput is very much larger than the change in Vinput, so this circuit is said to amplify the input.

```
change(decrease,vage(amp_in,ground))
change(decrease,vage(tri_grid,ground))
change(increase,resistance(tri_plate,tri_cathode,
     [[_,tri_plate,tri_cathode]]))
change(increase,vage(amp_out,ground))
and([produces(change(C1,vage(amp_in,ground)),
               change(C2,curr(tri_plate,tri_cathode,
                    [[tri,tri_plate,tri_cathode]]))),
     large(change(C2,curr(tri_plate,tri_cathode,
               [[tri,tri_plate,tri_cathode]]))),
     greater(change(C3,vage(amp_out,ground)),
               change(C1,vage(amp_in,ground)))])
```

READING 4. CATHODE BIAS AMPLIFIER (CAMP)

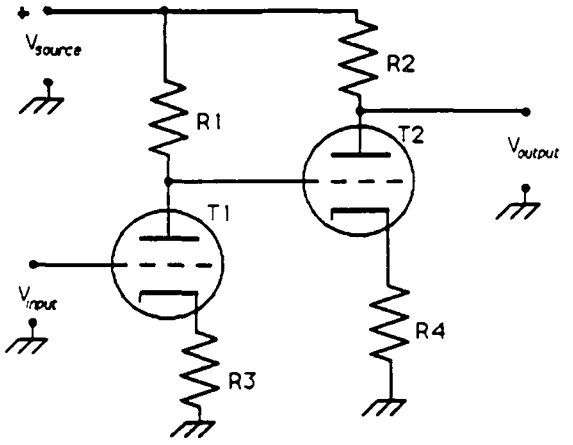


CAMP.

When Vinput increases, Voutput drops. The change in Voutput is larger than the change in Vinput. The value of R2 is chosen so that the voltage at the cathode is positive with respect to ground by a number of volts greater than the highest positive value of the input voltage. Thus the grid will always be negative relative to the cathode.

```
change(increase,vage(amp_in,ground))
and([change(decrease,vage(amp_out,ground)),
     greater(change(C1,vage(amp_out,ground)),
             change(C2,vage(amp_in,ground))),
     higherv(tri_cathode,ground)])
```

READING 5. TWO-STAGE TRIODE AMPLIFIER (TAMP)

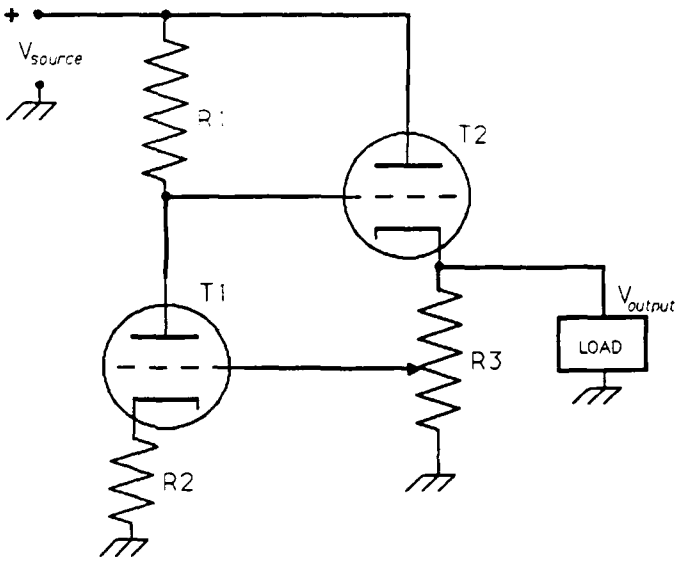


TAMP.

If the input signal, Vinput, increases, the resistance of T1 decreases, and the voltage on the plate of T1 goes down. This causes the resistance of T2 to increase, causing the output voltage, Voutput, to increase. The changes in plate voltage of T1 are much greater than the changes in Vinput, and the changes in plate voltage of T2 are even greater. Thus the total amplification effect is much greater than for a single triode.

```
change(increase,vage(amp_in,ground))
change(decrease,
         resistance(tri_plate,tri_cathode,[[_,tri_plate,tri_cathode]]))
change(decrease,vage(tri_plate,ground))
change(increase,resistance(tri2_plate,tri2_cathode,
          [[_,tri2_plate,tri2_cathode]]))
change(increase,vage(amp_out,ground))
greater(change(C1,vage(tri_plate,ground)),
        change(C2,vage(amp_in,ground)))
greater(change(C3,vage(tri2_plate,ground)),
        change(C2,vage(amp_in,ground)))
```

READING 6. VOLTAGE REGULATOR (VR)



VR1.

If more current begins to flow through the load, the output voltage begins to decrease. This makes the voltage on the grid of T1 more negative relative to the cathode causing less current to flow through T1. The grid of T2 then becomes less negative, decreasing the resistance of T2, and thus keeping the output voltage the same.

```
change(increase,curr(loada,loadb,[[load,loada,loadb]]))
change(decrease,vage(loada,ground))
change(decrease,vage(tri_grid,ground))
change(decrease,
      curr(tri_plate,tri_cathode,[[_,tri_plate,tri_cathode]]))
change(increase,vage(tri2_grid,ground))
change(decrease,
      resistance(tri2_plate,tri2_cathode,[[_,tri2_plate,tri2_cathode]]))
constant(vage(loada,ground))
```

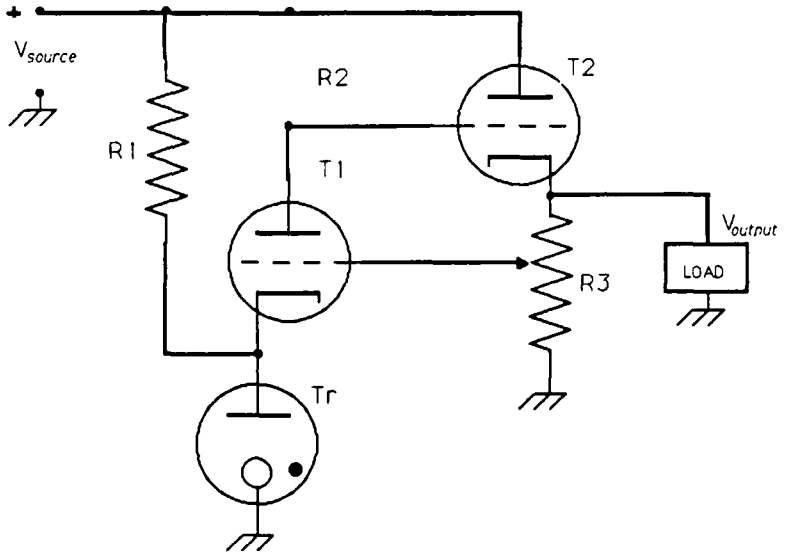
VR2.

Notice that if the source voltage decreases, the voltage on the grid of T1 will also decrease, but so will the voltage on the cathode of T1. The voltage on the grid of T1 relative to the T1 cathode stays constant. Therefore, the current through T1 will remain constant and the resistance of T2 will also remain constant. Since the source voltage has decreased, and the resistance of T2 is the same, the output voltage will stay decreased. So this circuit does not compensate for changes in source voltage.

```
change(decrease,vage(vs_pos,ground))
change(decrease,vage(tri_grid,ground))
change(decrease,vage(tri_cathode,ground))
not(change(decrease,vage(tri_grid,tri_cathode)))
not(change(increase,vage(tri_grid,tri_cathode)))
not(change(decrease,
       curr(tri_plate,tri_cathode,[[_,tri_plate,tri_cathode]])))
not(change(increase,
```

```
      curr(tri_plate,tri_cathode,[[_,tri_plate,tri_cathode]])))
not(change(increase,
     resistance(tri2_plate,tri2_cathode,[[_,tri2_plate,tri2_cathode]])))
not(change(increase,
     resistance(tri2_plate,tri2_cathode,[[_,tri2_plate,tri2_cathode]])))
change(decrease,vage(vs_pos,ground))
not(change(increase,
     resistance(tri2_plate,tri2_cathode),[[_,tri2_plate,tri2_cathode]]))
not(change(increase,
     resistance(tri2_plate,tri2_cathode),[[_,tri2_plate,tri2_cathode]]))
not(change(increase,vage(load_a,ground)))
```

READING 7. STABILIZED VOLTAGE REGULATOR (SVR)



SVR1.
Same as VR1.

SVR2.
If the source voltage decreases the output voltage and the voltage on the grid of T1 will start to decrease. However, the voltage on the cathode of T1 is kept constant by the voltage regulator tube Tr. Thus, the grid of T1 will become more negative relative to thecathode, causing the grid of T2 to become less negative, and the output voltage to remain constant.

```
change(decrease,vage(vs_pos,ground))
change(decrease,vage(loada,ground))
change(decrease,vage(tri_grid,ground))
constant(vage(tri_cathode,ground))
change(decrease,vage(tri_grid,tri_cathode))
change(increase,vage(tri2_grid,ground))
constant(vage(loada,ground))
```

APPENDIX 3

THE CIRCUITS REPRESENTED AS PROPOSITIONS

Regulator Tube Circuit

```
comp(pr,resistor,[pra,prb]),
  conn(pra,vs_w),
  conn(prb,output_w),
comp(rt,volt_reg_tube,[rtplate,rtcathode]),
  conn(rtplate,output_w),
  conn(rtcathode,ground_w),
comp(load,load,[loada,loadb]),
  conn(loada,output_w),
  conn(loadb,ground_w),
comp(vs,voltage_source,[vspos,vsneg,vsmax]),
  conn(vspos,vs_w),
  conn(vsneg,ground_w),
status(vs,on),
conn(ground_w,ground),
conn(source,vs_w),
conn(output,output_w),
isa(vs_w,wire),
isa(output_w,wire),
isa(ground_w,wire),
new_device_component(pr),
new_device_component(rt)
```

Series Tube Controller

```
comp(tri,triode,[triplate,tricathode,trigrid]),
  conn(triplate,vs_w),
  conn(trigrid,input_w),
  conn(tricathode,output_w),
comp(load,load,[loada,loadb]),
  conn(loada,output_w),
  conn(loadb,ground_w),
comp(vs,voltage_source,[vspos,vsneg,vsmax]),
  conn(vspos,vs_w),
  conn(vsneg,ground_w),
status(vs,on),
conn(ground_w,ground),
conn(stc_in,input_w),
conn(stc_out,output_w),
isa(vs_w,wire),
isa(input_w,wire),
isa(output_w,wire),
isa(ground_w,wire),
```

```
    new_device_component(tri),
    new_device_component(load)

Triode Amplifier

    comp(r,resistor,[r_a,r_b]),
      conn(r_a,vs_w),
      conn(r_b,output_w),
    comp(tri,triode,[tri_plate,tri_cathode,tri_grid]),
      conn(tri_plate,output_w),
      conn(tri_grid,input_w),
      conn(tri_cathode,ground_w),
    comp(vs,voltage_source,[vs_pos,vs_neg,vsmax]),
      conn(vs_pos,vs_w),
      conn(vs_neg,ground_w),
    status(vs,on),
    conn(ground_w,ground),
    conn(amp_in,input_w),
    conn(amp_out,output_w),
    isa(vs_w,wire),
    isa(input_w,wire),
    isa(output_w,wire),
    isa(ground_w,wire),
    new_device_component(r),
    new_device_component(tri)

Cathode Bias Amplifier

    comp(pr,resistor,[pr_a,pr_b]),
      conn(pr_a,vs_w),
      conn(pr_b,output_w),
    comp(tri,triode,[tri_plate,tri_cathode,tri_grid]),
      conn(tri_plate,output_w),
      conn(tri_grid,input_w),
      conn(tri_cathode,bias_w),
    comp(br,resistor,[br_a,br_b]),
      conn(br_a,bias_w),
      conn(br_b,ground_w),
    comp(vs,voltage_source,[vs_pos,vs_neg,vsmax]),
      conn(vs_pos,vs_w),
      conn(vs_neg,ground_w),
    status(vs,on),
    conn(ground_w,ground),
    conn(amp_in,input_w),
    conn(amp_out,output_w),
    isa(vs_w,wire),
    isa(input_w,wire),
    isa(output_w,wire),
```

```
    isa(ground_w,wire),
    isa(bias_w,wire),
    new_device_component(pr),
    new_device_component(tri),
    new_device_component(br)

Two-Stage Amplifier

    comp(pr,resistor,[pr_a,pr_b]),
      conn(pr_a,vs_w),
      conn(pr_b,input2_w),
    comp(tri,triode,[tri_plate,tri_cathode,tri_grid]),
      conn(tri_plate,input2_w),
      conn(tri_grid,input_w),
      conn(tri_cathode,bias_w),
    comp(br,resistor,[br_a,br_b]),
      conn(br_a,bias_w),
      conn(br_b,ground_w),
    comp(pr2,resistor,[pr2_a,pr2_b]),
      conn(pr2_a,vs_w),
      conn(pr2_b,output_w),
    comp(tri2,triode,[tri2_plate,tri2_cathode,tri2_grid]),
      conn(tri2_plate,output_w),
      conn(tri2_grid,input2_w),
      conn(tri2_cathode,bias2_w),
    comp(br2,resistor,[br2_a,br2_b]),
      conn(br2_a,bias2_w),
      conn(br2_b,ground_w),
    comp(vs,voltage_source,[vs_pos,vs_neg,vsmax]),
      conn(vs_pos,vs_w),
      conn(vs_neg,ground_w),
    status(vs,on),
    conn(ground_w,ground),
    conn(source,vs_w),
    conn(amp_in,input_w),
    conn(amp_out,output_w),
    isa(vs_w,wire),
    isa(input_w,wire),
    isa(input2_w,wire),
    isa(output_w,wire),
    isa(ground_w,wire),
    isa(bias_w,wire),
    isa(bias2_w,wire)
```

Voltage Regulator

```
comp(pr,resistor,[pr_a,pr_b]),
  conn(pr_a,vs_w),
  conn(pr_b,input2_w),
comp(tri,triode,[tri_plate,tri_cathode,tri_grid]),
  conn(tri_plate,input2_w),
  conn(tri_grid,input_w),
  conn(tri_cathode,bias_w),
comp(br,resistor,[br_a,br_b]),
  conn(br_a,bias_w),
  conn(br_b,ground_w),
comp(vs,voltage_source,[vs_pos,vs_neg,vsmax]),
  conn(vs_pos,vs_w),
  conn(vs_neg,ground_w),
comp(tri2,triode,[tri2_plate,tri2_cathode,tri2_grid]),
  conn(tri2_plate,vs_w),
  conn(tri2_grid,input2_w),
  conn(tri2_cathode,output_w),
comp(po,potentiometer,[poa,pob,poc]),
  conn(poa,output_w),
  conn(pob,input_w),
  conn(poc,ground_w),
comp(load,load,[loada,loadb]),
  conn(loada,output_w),
  conn(loadb,ground_w),
status(vs,on),
conn(ground_w,ground),
conn(output,output_w),
conn(source,vs_w),
isa(vs_w,wire),
isa(input_w,wire),
isa(input2_w,wire),
isa(output_w,wire),
isa(ground_w,wire),
isa(bias_w,wire),
new_device_component(pr),
new_device_component(tri),
new_device_component(br),
new_device_component(tri2),
new_device_component(po),
new_device_component(load)
```

Stabilized Voltage Regulator

```
comp(pr,resistor,[pr_a,pr_b]),
  conn(pr_a,vs_w),
  conn(pr_b,input2_w),
comp(tri,triode,[tri_plate,tri_cathode,tri_grid]),
  conn(tri_plate,input2_w),
  conn(tri_grid,input_w),
  conn(tri_cathode,bias_w),
comp(tri2,triode,[tri2_plate,tri2_cathode,tri2_grid]),
  conn(tri2_plate,vs_w),
  conn(tri2_grid,input2_w),
  conn(tri2_cathode,output_w),
comp(po,potentiometer,[poa,pob,poc]),
  conn(poa,output_w),
  conn(pob,input_w),
  conn(poc,ground_w),
comp(rtr,resistor,[rtr_a,rtr_b]),
  conn(rtr_a,vs_w),
  conn(rtr_b,bias_w),
comp(rt,volt_reg_tube,[rtplate,rtcathode]),
  conn(rtplate,bias_w),
  conn(rtcathode,ground_w),
comp(vs,voltage_source,[vs_pos,vs_neg,vsmax]),
  conn(vs_pos,vs_w),
  conn(vs_neg,ground_w),
comp(load,load,[loada,loadb]),
  conn(loada,output_w),
  conn(loadb,ground_w),
status(vs,on),
conn(ground_w,ground),
conn(output,output_w),
conn(source,vs_w),
isa(vs_w,wire),
isa(input_w,wire),
isa(input2_w,wire),
isa(output_w,wire),
isa(ground_w,wire),
isa(bias_w,wire)
```

Technical Document Center
AFHRL/LRS-TDC
Wright-Patterson AFB
OH 45433-6503

Dr. Robert Ahlers
Code N711
Human Factors Laboratory
Naval Training Systems Center
Orlando, FL 32813

Dr. Robert M. Aiken
Computer Science Department
038-24
Temple University
Philadelphia, PA 19122

Mr. Tejwansh S. Anand
Philips Laboratories
345 Scarborough Road
Briarcliff Manor
New York, NY 10520

Dr. Thomas H. Anderson
Center for the Study of Reading
174 Children's Research Center
51 Gerty Drive
Champaign, IL 61820

Dr. James D. Baker
Director of Automation and Research
Allen Corporation of America
209 Madison Street
Alexandria, VA 22314

Dr. Meryl S. Baker
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Isaac Bejar
Law School Admissions
Services
P.O. Box 40
Newtown, PA 18940-0040

Dr. Thomas G. Bever
Department of Psychology
University of Rochester
River Station
Rochester, NY 14627

Dr. Lawrence Birnbaum
The Institute for the
Learning Sciences
Northwestern University
1890 Maple Avenue
Evanston, IL 60201

Dr. John Black
Teachers College, Box 8
Columbia University
525 West 120th Street
New York, NY 10027

Dr. Arthur S. Blaiwes
Code N712
Naval Training Systems Center
Orlando, FL 32813-7100

Dr. Deborah A. Boehm-Davis
Department of Psychology
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Sue Bogner
Army Research Institute
ATTN: PERI-SF
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Jeff Bonar
Guidance Technology, Inc.
800 Vinial Street
Pittsburgh, PA 15212

Naval Supply Systems Command
NAVSUP 5512
ATTN: Sandra Borden
Washington, D.C. 20376-5000

Dr. Hugh Burns
Department of English
University of Texas
Austin, TX 78703

Dr. Francis Butler
Center for the Study of
Education
145 Moore Hall
University of California
Los Angeles, CA 90024

Dr. Joanne Capper, Director
Center for Research into Practice
3545 Albemarle Street, NW
Washington, DC 20008

Dr. Jaime G. Carbonell
Computer Science Department
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Ruth W. Chabay
CDEC, Hamburg Hall
Carnegie Mellon University
Pittsburgh, PA 15213

Dr. Fred Chang
Pacific Bell
2600 Camino Ramon
Room 3S-450
San Ramon, CA 94583

Dr. Davida Charney
English Department
Penn State University
University Park, PA 16802

Dr. Charles Clifton
Tobin Hall
Department of Psychology
University of
Massachusetts
Amherst, MA 01003

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 N. Quincy Street
Arlington, VA 22217-5000

Mr. Michael Cowen
Code 142
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Meredith P. Crawford
3563 Hamlet Place
Chevy Chase, MD 20815

Dr. Hans F. Crombag
Faculty of Law
University of Limburg
P.O. Box 616
Maastricht
The NETHERLANDS 6200 MD

Dr. Kenneth B. Cross
Anacapa Sciences, Inc.
P.O. Drawer Q
Santa Barbara, CA 93102

Dr. Cary Czichon
Intelligent Instructional Systems
Texas Instruments AI Lab
P.O. Box 660246
Dallas, TX 75266

Brian Dallman
Training Technology Branch
3400 TCHTW/TTGXC
Lowry AFB, CO 80230-5000

Margaret Day, Librarian
Applied Science Associates
P.O. Box 1072
Butler, PA 16003

Dr. Gerald F. DeJong
Computer Science Department
University of Illinois
Urbana, IL 61801

Dr. Sharon Derry
Florida State University
Department of Psychology
Tallahassee, FL 32306

Defense Technical
Information Center
Cameron Station, Bldg 5
Alexandria, VA 22314
(2 Copies)

Dr. Pierre Duguet
Organization for Economic
Cooperation and Development
2, rue Andre-Pascal
75016 PARIS
FRANCE

Dr. Richard Duran
Graduate School of Education
University of California
Santa Barbara, CA 93106

Dr. Ralph Dusek
V-P Human Factors
JIL Systems
1225 Jefferson Davis Hwy.
Suite 1209
Arlington, VA 22201

Dr. Susan Epstein
144 S. Mountain Avenue
Montclair, NJ 07042

ERIC Facility-Acquisitions
2440 Research Blvd, Suite 550
Rockville, MD 20850-3238

Dr. Debra Evans
Applied Science Associates, Inc.
P. O. Box 1072
Butler, PA 16003

Dr. Lorraine D. Eyde
Office of Personnel Management
Office of Examination Development
1900 E St., NW
Washington, DC 20415

Dr. Jean-Claude Falmagne
Irvine Research Unit in
Mathematical & Behavioral Sciences
University of California
Irvine, CA 92717

Dr. Beatrice J. Farr
Army Research Institute
PERI-IC
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Marshall J. Farr, Consultant
Cognitive & Instructional Sciences
2520 North Vernon Street
Arlington, VA 22207

Dr. P-A. Federico
Code 51
NPRDC
San Diego, CA 92152-6800

Dr. Paul Feltovich
Southern Illinois University
School of Medicine
Medical Education Department
P.O. Box 3926
Springfield, IL 62708

Dr. Elizabeth Fennema
Curriculum and Instruction
University of Wisconsin
225 North Mills Street
Madison, WI 53706

Prof. Donald Fitzgerald
University of New England
Department of Psychology
Armidale, New South Wales 2351
AUSTRALIA

Dr. Michael Flaningam
Code 52
NPRDC
San Diego, CA 92152-6800

Dr. J. D. Fletcher
Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311

Dr. Linda Flower
Carnegie-Mellon University
Department of English
Pittsburgh, PA 15213

Dr. Barbara A. Fox
University of Colorado
Department of Linguistics
Boulder, CO 80309

Dr. Carl H. Frederiksen
Dept. of Educational Psychology
McGill University
3700 McTavish Street
Montreal, Quebec
CANADA H3A 1Y2

Dr. John R. Frederiksen
BBN Laboratories
10 Moulton Street
Cambridge, MA 02238

Department of Humanities and
Social Sciences
Harvey Mudd College
Claremont, CA 91711

Dr. Alfred R. Fregly
AFOSR/NL, Bldg. 410
Bolling AFB, DC 20332-6448

Dr. Michael Friendly
Psychology Department
York University
Toronto ONT
CANADA M3J 1P3

Col. Dr. Ernst Frise
Heerespsychologischer Dienst
Maria Theresien-Kaserne
1130 Wien
AUSTRIA

Dr. Philip Gillis
ARI-Fort Gordon
ATTN: PERI-ICD
Fort Gordon, GA 30905

Mr. Lee Gladwin
305 Davis Avenue
Leesburg, VA 22075

Dr. Sam Glucksberg
Department of Psychology
Princeton University
Princeton, NJ 08540

Dr. Joseph Goguen
Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dr. Susan R. Goldman
Peabody College, Box 45
Vanderbilt University
Nashville, TN 37203

Mr. Harold Goldstein
University of DC
Department Civil Engineering
Bldg. 42, Room 112
4200 Connecticut Avenue, N.W.
Washington, DC 20008

Dr. Sherrie Gott
AFHRL/MOMJ
Brooks AFB, TX 78235-5601

Dr. T. Govindaraj
Georgia Institute of
Technology
School of Industrial
and Systems Engineering
Atlanta, GA 30332-0205

Dr. Dik Gregory
Admiralty Research
Establishment/AXE
Queens Road
Teddington
Middlesex, ENGLAND TW110LN

Michael Habon
DORNIER GMBH
P.O. Box 1420
D-7990 Friedrichshafen 1
WEST GERMANY

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Mr. H. Hamburger
Department of Computer Science
George Mason University
Fairfax, VA 22030

Dr. Bruce W. Hamill
Research Center
The Johns Hopkins University
Applied Physics Laboratory
Johns Hopkins Road
Laurel, MD 20707

Dr. Patrick R. Harrison
Computer Science Department
U.S. Naval Academy
Annapolis, MD 21402-5002

Janice Hart
Office of the Chief
of Naval Operations
OP-11U2
Department of the Navy
Washington, D.C. 20350-2000

Dr. Wayne Harvey
Center for Learning Technology
Education Development Center
55 Chapel Street
Newton, MA 02160

Dr. Frederick Hayes-Roth
Teknowledge
P.O. Box 10119
1850 Embarcadero Rd.
Palo Alto, CA 94303

Dr. James E. Hoffman
Department of Psychology
University of Delaware
Newark, DE 19711

Dr. Melissa Holland
Army Research Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Ms. Julia S. Hough
Cambridge University Press
40 West 20th Street
New York, NY 10011

Dr. William Howell
Chief Scientist
AFHRL/CA
Brooks AFB, TX 78235-5601

Dr. Steven Hunka
3-104 Educ. N.
University of Alberta
Edmonton, Alberta
CANADA T6G 2G5

Dr. Jack Hunter
2122 Coolidge Street
Lansing, MI 48906

Dr. Janet Jackson
Rijksuniversiteit Groningen
Biologisch Centrum, Vleugel D
Kerklaan 30, 9751 NN Haren
The NETHERLANDS

Mr. Roland Jones
Mitre Corp., K-203
Burlington Road
Bedford, MA 01730

Dr. Michael Kaplan
Office of Basic Research
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. David Kieras
Technical Communication Program
TIDAL Bldg., 2360 Bonisteel Blvd.
University of Michigan
Ann Arbor, MI 48109-2108

Dr. Walter Kintsch
Department of Psychology
University of Colorado
Boulder, CO 80309-0345

Dr. Eugene Lee
Naval Postgraduate School
Monterey, CA 93943-5026

Dr. Yuh-Jeng Lee
Department of Computer Science
Code 52Le
Naval Postgraduate School
Monterey, CA 93943

Dr. Jill F. Lehman
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Dr. Doris K. Lidtke
Software Productivity Consortium
1880 Campus Commons Drive, North
Reston, VA 22091

Dr. Charlotte Linde
Structural Semantics
P.O. Box 707
Palo Alto, CA 94320

Dr. Jack Lochhead
University of
Massachusetts
Physics Department
Amherst, MA 01003

Vern M. Malec
NPRDC, Code 52
San Diego, CA 92152-6800

Dr. Mary Martino
Director, Educational Technology
HQ USAFA/DFTE
USAF Academy, CO 80840-5000

Dr. Elaine Marsh
Naval Center for Applied Research
in Artificial Intelligence
Naval Research Laboratory
Code 5510
Washington, DC 20375-5000

Dr. Sandra P. Marshall
Dept. of Psychology
San Diego State University
San Diego, CA 92182

Dr. Manton M. Matthews
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. James L. McClelland
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Kathleen McKeown
Columbia University
Department of Computer Science
450 Computer Science Building
New York, NY 10027

Dr. Douglas L. Medin
Department of Psychology
University of Michigan
Ann Arbor, MI 48109

Dr. Arthur Melmed
Computer Arts and
Education Laboratory
New York University
719 Broadway, 12th floor
New York, NY 10003

Dr. Jose Mestre
Department of Physics
Hasbrouck Laboratory
University of Massachusetts
Amherst, MA 01003

Dr. Alan Meyrowitz
Office of Naval Research
Code 1133
800 N. Quincy
Arlington, VA 22217-5000

Dr. Ryszard S. Michalski
Center for Artificial Intelligence
George Mason University
Science and Tech Room 301
4400 University Drive
Fairfax, VA 22030-4444

Dr. Vittorio Midoro
CNR-Istituto Tecnologie Didattiche
Via All'Opera Pia 11
GENOVA-ITALIA 16145

Dr. George A. Miller
Dept. of Psychology
Green Hall
Princeton University
Princeton, NJ 08540

Dr. Jason Millman
Department of Education
Roberts Hall
Cornell University
Ithaca, NY 14853

Dr. Andrew R. Molnar
Applic. of Advanced Technology
Science and Engr. Education
National Science Foundation
Washington, DC 20550

Dr. William Montague
NPRDC Code 13
San Diego, CA 92152-6800

Dr. Melvin D. Montemerlo
NASA Headquarters
Code RC
Washington, DC 20546

Dr. William R. Murray
FMC Corporation
Central Engineering Labs
1205 Coleman Avenue
Box 580
Santa Clara, CA 95052

Dr. T. Niblett
The Turing Institute
George House
36 North Hanover Street
Glasgow G1 2AD
UNITED KINGDOM

Library, NPRDC
Code P201L
San Diego, CA 92152-6800

Librarian
Naval Center for Applied Research
in Artificial Intelligence
Naval Research Laboratory
Code 5510
Washington, DC 20375-5000

Dr. Harold F. O'Neil, Jr.
School of Education - WPH 801
Department of Educational
Psychology & Technology
University of Southern California
Los Angeles, CA 90089-0031

Dr. Judith Reitman Olson
Graduate School of Business
University of Michigan
Ann Arbor, MI 48109-1234

Office of Naval Research,
Code 1142CS
800 N. Quincy Street
Arlington, VA 22217-5000
(6 Copies)

Dr. Judith Orasanu
Basic Research Office
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. John Oriel
Navy Training Systems
Center (Code 212)
12350 Research Parkway
Orlando, FL 32826-3224

Dr. Glenn Osga
NOSC, Code 441
San Diego, CA 92152-6800

Dr. Nancy N. Perry
Naval Education and Training
Program Support Activity
Code-047
Building 2435
Pensacola, FL 32509-5000

Dept. of Administrative Sciences
Code 54
Naval Postgraduate School
Monterey, CA 93943-5026

Dr. Mary C. Potter
Department of Brain and
Cognitive Sciences
MIT (E-10-039)
Cambridge, MA 02139

Dr. Joseph Psotka
ATTN: PERI-IC
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333-5600

Dr. J. Wesley Regian
AFHRL/IDI
Brooks AFB, TX 78235

Dr. Charles M. Reigeluth
330 Huntington Hall
Syracuse University
Syracuse, NY 13244

Dr. Lauren Resnick
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Edwina L. Rissland
Dept. of Computer and
Information Science
University of Massachusetts
Amherst, MA 01003

Mr. William A. Rizzo
Code 71
Naval Training Systems Center
Orlando, FL 32813

Dr. Linda G. Roberts
Science, Education, and
Transportation Program
Office of Technology Assessment
Congress of the United States
Washington, DC 20510

Lowell Schoer
Psychological & Quantitative
Foundations
College of Education
University of Iowa
Iowa City, IA 52242

Dr. Janet W. Schofield
816 LRDC Building
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Nuria Sebastian
Dep. Psicologia Basica
Univ. Barcelona
Adolf Florensa s.n.
08028 Barcelona
SPAIN

Dr. Judith W. Segal
OERI
555 New Jersey Ave., NW
Washington, DC 20208

Dr. Robert J. Seidel
US Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Michael G. Shafto
NASA Ames Research Ctr.
Mail Stop 239-1
Moffett Field, CA 94035

Dr. Valerie L. Shalin
Department of Industrial
Engineering
State University of New York
342 Lawrence D. Bell Hall
Buffalo, NY 14260

Mr. Colin Sheppard
AXC2 Block 3
Admiralty Research Establishment
Ministry of Defence Portadown
Portsmouth Hants P064AA
UNITED KINGDOM

Dr. Randall Shumaker
Naval Research Laboratory
Code 5510
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Dr. Derek Sleeman
Computing Science Department
The University
Aberdeen AB9 2FX
Scotland
UNITED KINGDOM

Ms. Gail K. Slemon
LOGICON, Inc.
P.O. Box 85158
San Diego, CA 92138-5158

Dr. Robert Smillie
Navy Personnel R&D
San Diego, CA 92152-6800

Dr. Edward E. Smith
Department of Psychology
University of Michigan
330 Packard Road
Ann Arbor, MI 48103

Dr. Alfred F. Smode
Code 7A
Research and Development Dept.
Naval Training Systems Center
Orlando, FL 32813-7100

Dr. Elliot Soloway
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Linda B. Sorisio
IBM-Los Angeles Scientific Center
11601 Wilshire Blvd., 4th Floor
Los Angeles, CA 90025

N. S. Sridharan
FMC Corporation
Box 580
1205 Coleman Avenue
Santa Clara, CA 95052

Dr. Marian Stearns
SRI International
333 Ravenswood Ave.
Room B-5124
Menlo Park, CA 94025

Dr. Thomas Stichl
Applied Behavioral and
Cognitive Sciences, Inc.
2062 Valley View Blvd.
El Cajon, CA 92019-2059

Dr. David E. Stone
Computer Teaching Corporation
1713 South Neil Street
Urbana, IL 61820

Dr. M. Martin Taylor
DCIEM
Box 2000
Downsview, Ontario
CANADA M3M 3B9

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1205 Coleman Avenue, Box 580
Santa Clara, CA 95052

Dr. Sharon Tkacz
Allen Corporation
209 Madison Street
Alexandria, VA 22314

Dr. Douglas Towne
Behavioral Technology Labs
University of Southern California
250 N. Harbor Dr., Suite 309
Redondo Beach, CA 90277

Dr. Harold P. Van Cott
Committee on Human Factors
National Academy of Sciences
2101 Constitution Avenue
Washington, DC 20418

Dr. Frank L. Vicino
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Jerry Vogt
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Thomas A. Warm
FAA Academy AAC934D
P.O. Box 25082
Oklahoma City, OK 73125

Dr. Beth Warren
BBN Laboratories, Inc.
10 Moulton Street
Cambridge, MA 02238

Dr. Diana Wearne
Department of Educational
Development
University of Delaware
Newark, DE 19711

Dr. Douglas Wetzel
Code 51
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Barbara White
School of Education
Tolman Hall, EMST
University of California
Berkeley, CA 94720

Dr. David Wilkins
University of Illinois
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL 61801

Dr. Marsha R. Williams
Applic. of Advanced Technologies
National Science Foundation
SEE/MDRISE
1800 G Street, N.W., Room 635-A
Washington, DC 20550

Dr. Robert A. Wisher
U.S. Army Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Frank B. Withrow
U.S. Department of Education
Room 504D, Capitol Plaza
555 New Jersey Avenue, N.W.
Washington, DC 20208

Dr. Merlin C. Wittrock
Graduate School of Education
UCLA
Los Angeles, CA 90024

Dr. Wallace Wulfeck, III
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Masoud Yazdani
Dept. of Computer Science
University of Exeter
Prince of Wales Road
Exeter EX44PT
ENGLAND

Frank R. Yekovich
Dept. of Education
Catholic University
Washington, DC 20064

Dr. Joseph L. Young
National Science Foundation
Room 320
1800 G Street, N.W.
Washington, DC 20550

Dr. Uri Zernik
General Electric
Research & Development Center
Artificial Intelligence Program
PO Box 8
Schenectady, NY 12301