MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

AI Working Paper 144 April 1977

A THEORY OF PLANS FOR ELECTRONIC CIRCUITS

by

Johan de Kleer

ABSTRACT:

A plan for a device assigns purposes to each of the more primitive components and explains how these components interact to achieve the desired behavior of the composite device. Such an information structure is critically important in analyzing, designing or troubleshooting devices. The first goal of this research is to develop a theory of plans for electronic circuits which can be used for these purposes. The second goal is the construction of a system which can automatically recognize a plan for a circuit from a geometrical representation of the circuit's schematic diagram.

Recognition is a process which recaptures the plan the designer originally had in mind. A theory of schemata will be introduced in which recognition is viewed as the identification of an instance of a schema in the library with the particular circuit being recognized. This process is guided by topological and geometric evidence extracted from the circuit schematic. Causal reasoning, using the technique of propagation of constraints, provides further evidence. One important use of causal reasoning is the confirmation of tentative instantiations based on topological and geometric evidence alone.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

INTRODUCTION

Goals

A plan for a device assigns purposes to each of the more primitive components and explains how these components interact to achieve the desired behavior of the composite device. Such an information structure is critically important in analyzing, designing or troubleshooting devices. The first goal of this research is to develop a theory of plans for electronic circuits which can be used for these purposes. The second goal is the construction of a system which can automatically recognize a plan for a circuit from a geometrical representation of the circuit's schematic diagram, using only minimal annotation.

Recognition relies on two fundamentally different sources of information: it relies on the plan library to determine whether the circuit (or any of its fragments) has been seen before, and it relies on causal reasoning to analyze circuit fragments for which no plan is found. Since any new circuit presented to the system will only be partially described by the plans in the library, causal reasoning plays a crucial role in the recognition process. In order to interface these two kinds of knowledge a common description language must be developed so that the information provided by the static plan structures can be combined with the information obtained by the more dynamic causal reasoning. This is the fundamental theoretical problem which must be addressed if the goals of the previous paragraph are to be met.

After a plan for the circuit has been recognized the system should be able to answer questions about the circuit. This plan can also be passed to a troubleshooting program such as WATSON [Brown 76] to repair the circuit. When combined with the technique of propagation of constraints [Sussman & Stallman 75] [Stallman & Sussman 76] [de Kleer 76], this research is a fundamental step towards building a general circuits expert. One design constraint imposed on this expert is that it should be *articulate*, able to explain both its deductions and the methods it used to deduce them. Hence, this research will necessarily involve some exploration into the problems involved in answering questions and generating explanations. Such an expert should also be able to supply expertise in a tutoring system [Brown *et.al.* 74] [Brown *et.al.* 76] [Goldstein 77].

The Importance of Plans

Every physical or informational object has a structure, not necessarily directly reflected in the more primitive elements which compose it, which abstracts the important aspects and interactions of these primitive objects. The power of the plan is that it is simpler than the object it is describing, yet few of the features of the described object relevant to the intended function are lost, and so is more convenient to work with. The plan is also useful since other information about the device can be attached to it. For example, hints about how to recognize the device and typical bugs it might have can be attached to the plan so that such information can be conveniently retrieved when needed.

There are many direct benefits from developing a theory of plans for circuits. The

technology of electronics is expanding at such a rapid rate that the complexity of devices is increasing by orders of magnitude every few years. This increase in complexity is making it more and more difficult to describe, understand and maintain these complex devices. One of the goals of this research is to formalize a language for plans which can be used by designers to describe complex devices. Such a precise and uniform method for describing devices would be useful for understanding and maintaining them.

The notion of plan can have pedagogical impact. Plans have always been used to describe complex devices. Unfortunately, the use of plans is not made explicit and the plan idea is never formalized. The student has to learn these ideas largely through osmosis. For example, one of the experiences with SOPHIE [Brown *et.al.* 74] has been that the major source of students' difficulties is an inability to assimilate a plan for a circuit rather than a lack of specific troubleshooting strategies. In developing a clear theory of plans we lay a foundation for conveying information about complex devices, and this is extremely useful in education.

Related Work

The proposed investigation builds on three major areas of previous research: research on plans for circuits for design and troubleshooting, the technique of propagation of constraints for circuit analysis, and the more sophisticated notions of plans developed in the programming domain.

DESI [McDermott 76] uses plans to design simple circuits. Aside from being important as another example of the use of plans, the fact that circuits are purposefully designed provides a great deal of insight into their plan structure. In the design process the relation between the primitive elements and the composite device becomes clear, since, in design, purpose typically precedes the choice of particular primitive elements. Hence design motivates the powerful idea that primitive elements have purposes which can be related to the purpose of the entire circuit.

Brown [Brown 76] demonstrates the importance of plans to troubleshooting. Unfortunately, Brown's and McDermott's perspectives on plans aren't entirely compatible, and neither notion of plan incorporates any structure to accommodate recognition. They also do not admit partial plans for the circuit. We will attempt to develop a more general notion of plan which can be useful in recognition, and yet also be suitable for troubleshooting and design.

A large part of circuit recognition involves analyzing circuit fragments to determine what they do and checking the hypotheses against the actual circuit schematic. This requires an knowledge of the components and how they interact. The technique of propagation of constraints in EL [Sussman & Stallman 75] and INTER [de Kleer 76] provides this expertise. However, to deal with partial information and to have a flexible interface with other phases of recognition, requires a more sophisticated, qualitative, propagation of constraints.

One of the outgrowths from EL has been AMORD, [Doyle 76] [Doyle, Steele & Sussman 77] a dependency-based problem-solving language. AMORD is a generalization of the

system EL was based on, and thus is ideally suited for a more flexible kind of propagation of constraints. AMORD will also be useful in recognition. It provides methods for maintaining beliefs and dependency-directed backtracking, both of which are extremely important in the matching process required for recognition.

The study of plans as applied to the programming domain has a longer history [Goldstein 74] [Sussman 73]. Recently, far more sophisticated and precise theories of plans have been developed in the programming domain [Waters 76] [Rich & Shrobe 76]. Many of these notions can also be applied to circuits once two differences in these domains have been understood. First, values passed in variables in programs are different than signals transmitted through wires in circuits. Second, programs execute in a strict time order, while circuits operate continuously and simultaneously. The second difference is perhaps the more crucial since it requires a fundamental rethinking of the notions of input and output. It also explains why the reasoning mechanism appropriate for circuits is propagation of constraints; while for programs it is symbolic evaluation.

It is important to distinguish the current work on plans from Artificial Intelligence research on planning [Hewitt 71] [Sacerdoti 75] [Rulifson 72] [Fikes, Hart & Nilsson 72]. Effecting changes on the environment is expensive and often undoable, and therefore a possible path to the goal should be found before attempting any actions. Usually, this path is described to a limited level of detail. The process of finding such a path is called planning. Planning is concerned with chronology and the interactions between an environment and actions upon it. The current research focusses on describing the device which is produced as the result of a design process, and is not particularily interested in the process by which that design was arrived at. Consider the construction of a house. The blueprints of that house are a part of a plan for that house (the complete plan would mention, for example, why the bathroom was there). The builder must do some planning to determine that the foundation should be laid before putting on the roof. The result of this planning is not a blueprint for the house, but a course of action which will plausibly lead him to successfully construct the house.

MODULES, PLANS AND PLAN FRAGMENTS

A Simple Theory of Modules and Plans

The simplicity of a description comes from two sources: first, a description has a particular point of view which excludes details irrelevant to that point of view; second, the description refers to intermediate objects. The level of detail of a particular description can be expanded by examining the descriptions of the intermediate objects it refers to. A device can be described at many different levels. We can describe the radio in terms of stages and the transistor in terms of the Ebers-Moll model. We can describe the physical layout of the components on the pc-board and the color bands on the resistor. At any level of description we speak in terms of objects which may not have any direct physical referent in the device. For example, the dependent sources of the Ebers-Moll model have no physical reality. Similarly, the components of a receiver

stage may be found physically close together, but there is no distinguished physical characteristic which identifies members of the stage. The most important observation about any description, however, is that it ignores many irrelevant details. To get more detail a lower-level description must be used. For example, the description of a radio will not mention resistors, but the description of the AVC for that radio will. Every description has a particular point of view which excludes many details as irrelevant. A radio, from the point of view of a receiver, is described in terms of stages. A radio, as a piece of art, is described by the styling of its cabinet. The same radio can also be described as a weight, scaffold, weapon, heirloom, etc.

A rather simple-minded theory of description can be constructed from these observations. It requires some modifications, but contains the basic ideas. The idea of a module, taken from programming and electronics, crystalizes the notion of intermediate object. A module has well-defined boundaries, inputs, and outputs. Beyond these inputs and outputs the module is treated as a black-box. The description, or plan, of a module has three main parts: a *decomposition*, a *behavioral description* and a *functional description*. The decomposition describes how the module is hierarchically decomposed into submodules. The behavioral description describes the input-output behavior of the module without going into the details of its internal behavior. The functional description indicates how the submodules cooperatively function to achieve the behavioral description of the module. Since the purposes of any module are described by the functional description of its parent, the functional descriptions provide a teleological description of the entire device. A complete plan for a device determines a particular point of view; this plan is constructed by using the point of view to establish a top-level module and then including all its descendant submodules. These submodules refer to features of the physical device, and thus explain the device in terms of these physical features and from the particular point of view.

The existence of an intermediate structure between the primitive level and the composite device makes it much easier to understand the device since the purposes of the primitives need only be related to their most immediate parent module, rather than requiring the consideration of arbitrary interactions. Behavioral descriptions allow the designer to use black-boxes in his design without having to understand the device inside the black-box. Similarly, the behavioral description aids the troubleshooter since he need only look inside the modules which violate their behavioral descriptions.

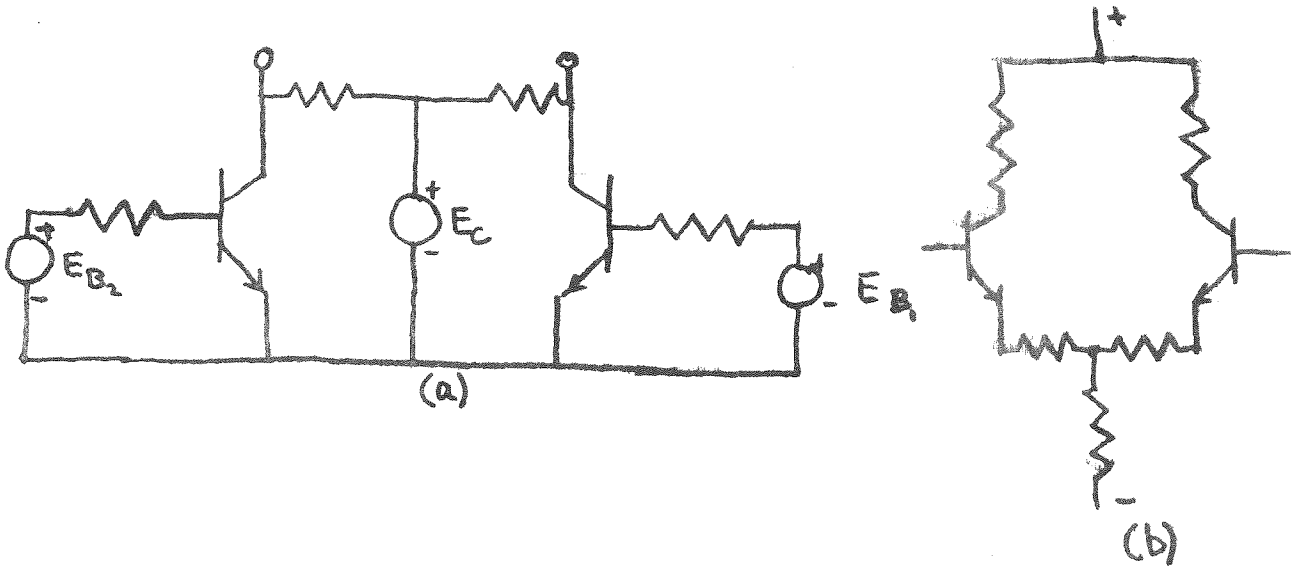
Some Inadequacies of the Simple Theory

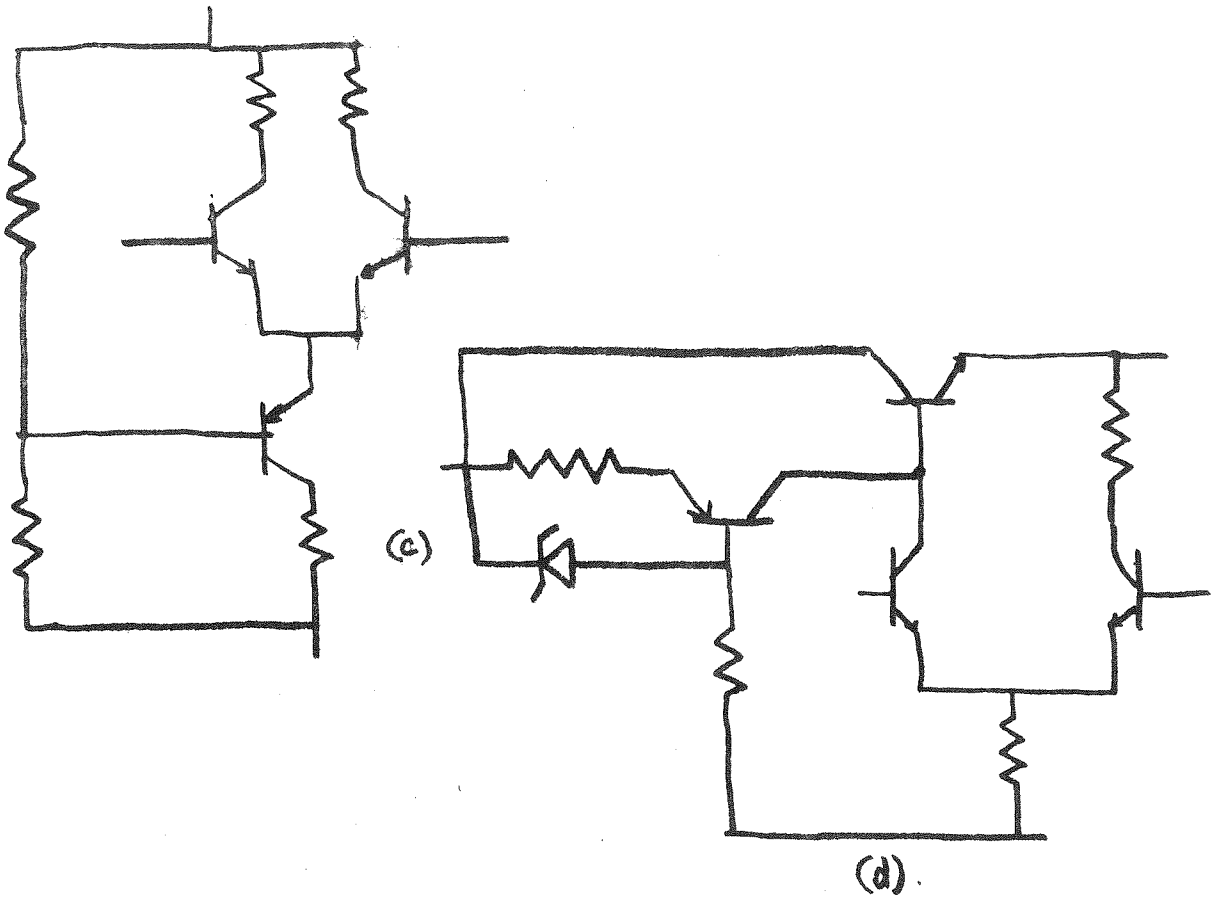
Although the simple theory presented in the previous section is a large step towards our goal of a theory of plans, it is inadequate. The strict module-submodule hierarchy does not admit shared submodules and primitives, and does not allow a module to be related to anything but its most immediate parents and submodules. The theory also lacks a notion of similarity. For example, the similarity between two different emitter-coupled pairs cannot be expressed.

In the strictly hierarchical theory the purposes of a module can only be related to its most immediate parent. However, brother-brother relations are also important. For example, the desired gain of a two-stage amplifier does not necessarily constrain the gain of each stage, but only imposes a composite constraint on the product of the two gains. The only purpose expressible in the hierarchical theory is that each stage must be an amplifier. Another example is a power supply, where we have the choice of including a more robust transformer or building a more sophisticated regulator section.

Another problem with the simple theory is illustrated by one power supply which is shared among many modules. Another example is that a single stage amplifier may have a decoupling capacitor on its input and output, but in a two-stage amplifier the output capacitor of the first stage can be combined with the input capacitor of the second. In a direct-coupled amplifier the DC-component of the previous stage can be used to provide the bias for the next stage. These examples demonstrate that submodules can be shared, which necessitates a revision of the simple plan theory in which a module can have only one parent. This problem also raises difficulties with recognition, since any component already included in one module must still be available for inclusion in other modules.

Our plan theory does not have any mechanism to express the similarity between circuits. For example, an emitter-coupled pair can be constructed in many different ways: the emitters can be tied directly to ground, they can be tied to ground through resistors, they can be tied to a current source, etc.





This requires a different plan be specified for every new emitter-coupled pair. Diagrammatically, the distinguishing feature of an emitter-coupled pair is that its transistors are normally drawn back-to-back. Only after one has noticed this feature are the other components connected to the transistors examined. In short, recognition demands that the the general type be identified before the specific instance, but the simple theory has no structures to accommodate this.

Plans, Plan Fragments and Plan Fragment Schemata

The previous section motivates two modifications to the simple theory, the first of which is that plans should be allowed to be non-hierarchical. Since the usual definition of module implies a strict hierarchy, we introduce the term *plan fragment* to refer to the generalized non-hierarchical modules. The term module is still a useful one since it refers to a plan fragment whose immediate structure is purely hierarchical.

Plan fragments are much like modules in that they have a decomposition, behavioral

description and functional description. This decomposition is in terms of other plan fragments. Their behavioral descriptions and functional descriptions are much more complex due to the possibility of shared subfragments. However, the introduction of plan fragment schemata will alleviate many of these complexities.

There are a large number of possible superheterodyne receivers which are all different. On the other hand all these receivers are of the superheterodyne type and this fact is critical in designing and understanding all such receivers. Understanding how one superheterodyne receiver works makes it much easier to understand other superheterodyne receivers. Similarly, in the design process, it is convenient to speak of a superheterodyne receiver without actually having built one. The above theories of plans and modules do not recognize the similarities between two different superheterodyne receivers.

However, to some extent the plan fragment can be used to describe the similarities. For example, a plan fragment for a superheterodyne might describe it has having an rf-stage, converter, i-f-stage, detector and audio-stage. Two superheterodyne receivers which differ in i-f or audio-stage output power may have identical plan fragments at this level. However, a superheterodyne receiver can have more or less rf-stages, it can have multiple i-f-stages and i-f's.

For this reason we introduce the second modification of the simple theory, the notion of a *plan fragment schema*. The superheterodyne plan fragment schema describes those characteristics of a circuit essential to it being a superheterodyne receiver. There is only one superheterodyne plan fragment schema and the plan fragments of all superheterodyne plan fragments are instantiations of it. The schema for a superheterodyne requires that there is some piece of circuitry that functions as a converter producing a fixed intermediate frequency.

As with plan fragments, schemata have decompositions, behavioral descriptions and functional descriptions. There are, however, some fundamental differences. The decomposition of a schema is into plan fragment patterns which describe other schemata by function rather than by name. The behavioral description describes how instantiated plan fragments of the schema behave. Notice that this is an *intrinsic* description, while the plan fragment patterns of a schema are *extrinsic* descriptions. The functional description of any plan fragment refers primarily to the functional description of its schema. In short, the three main parts of a plan fragment are derived from instantiating the corresponding parts of its schema.

The entire network of plan fragments derived from starting with the top-level plan and expanding all of its terminals until the components are reached is called the *plan* for the circuit. Thus a plan fragment can be considered a node in the plan. Note that *plan fragment* refers to the description of a particular circuit, while *plan fragment schema* refers to a description of a type of *plan fragment*. Plan schemata are found only in the *library* which is used in design and recognition, and plan fragments are present only when a particular circuit is being considered.

The schemata in the library contain information which describes the circuit after it has been parsed by that schema, but it is important to note that it can also have attached to it other

kinds of information. In particular it can contain information about how to recognize that schema in a larger circuit and typical bugs the circuit can have. In general, the plan provides a very convenient structure for attaching annotation about the circuit.

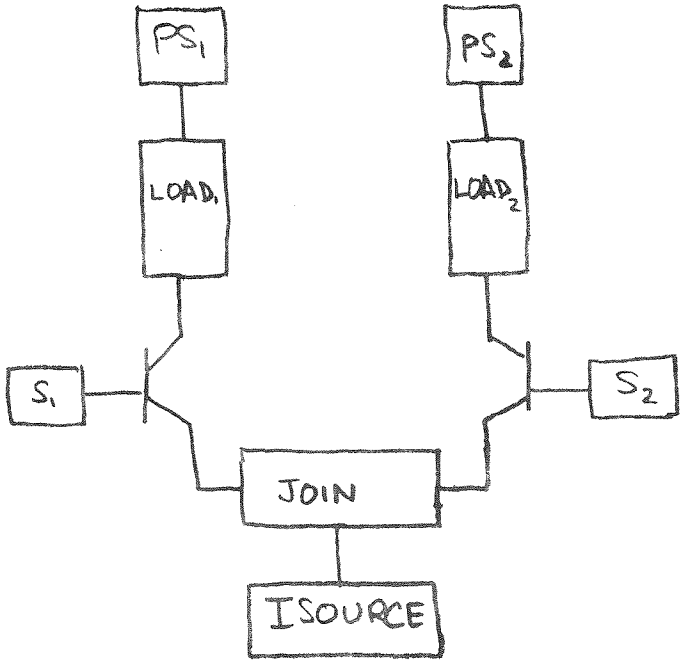
In the design process, plan fragment schemata are selected and partially instantiated. As design progresses, the nature of the desired goals and the plan fragments available to fill the terminals of the schemata add constraints to cause further instantiation. Finally, the plan fragments are completely instantiated and the circuit design is complete.

Recognition involves reconstructing this design. Thus recognition can be executed much the same as design, except that the schematic is used to distinguish between possible instantiations. The idea of instantiating schemata in the recognition process originates from Minsky's frame theory [Minsky 74].

Plans for Emitter-Coupled Pairs

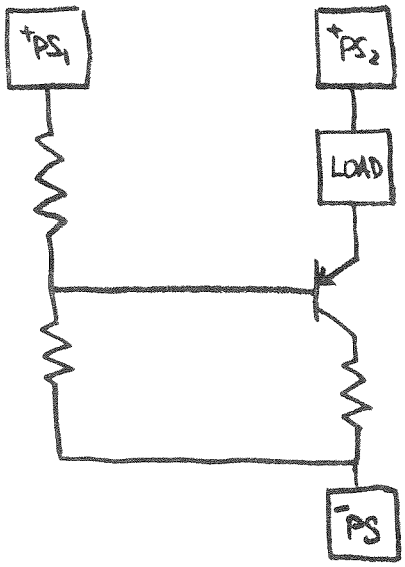
To illustrate these concepts some plans for the emitter-coupled circuits discussed earlier will be presented. Note that these are just tentative sketches of what the plans could be like: the plans are only partial, and not enough of them are included to handle all the examples.

The following is a schema for an emitter-coupled pair:

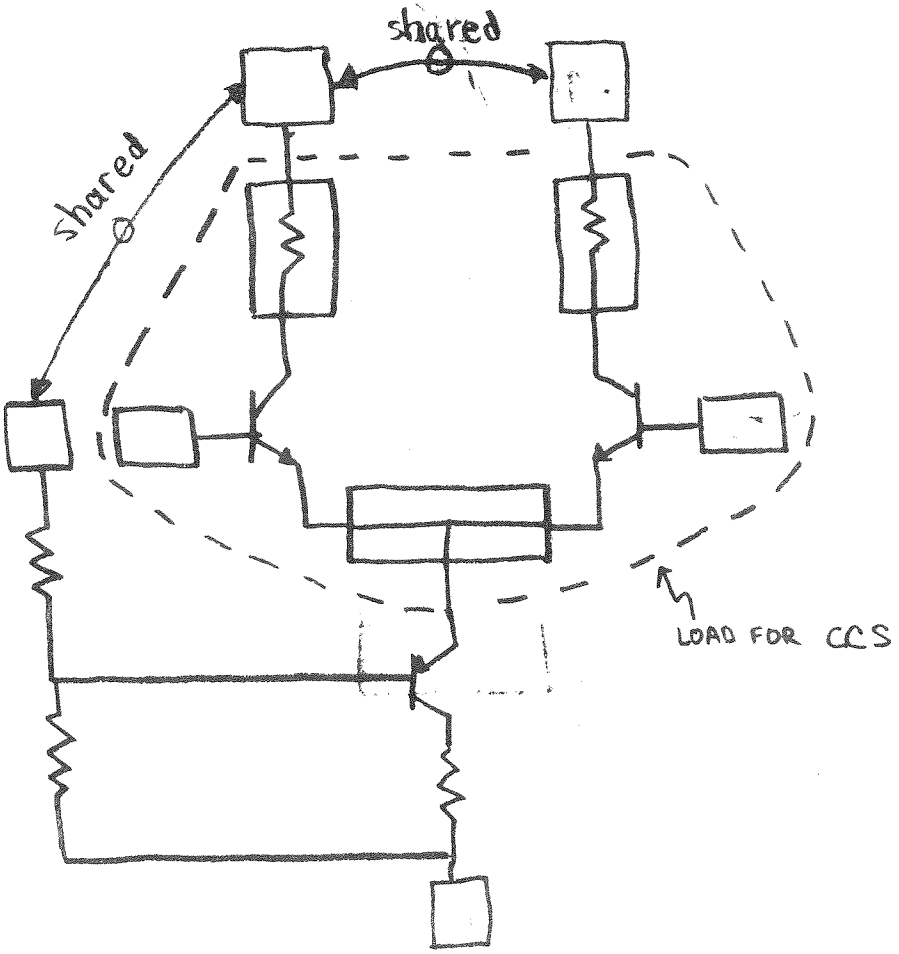


The objects in this diagram refer to the terminals of the schema, the lines indicate the connectivity between them. When a component is indicated it must be included in any instantiation. The boxes are labeled by extrinsic descriptions of other schemata that must be inserted there. For example, the above schema requires something that acts as a current source between the emitters and ground.

A schema for a particular current source is:



Using the resistor as a load, and sharing the various power sources, these two schemata can be instantiated on circuit Ic. Note that the load terminal of the current source must be identified with almost the entire emitter-coupled pair.



A resistor also serves as a simple current source, and a join can be accomplished by two resistors: thus we have a plan for circuit Ib. A very poor current source is just a power source, and a power source can be a voltage source in series with a resistor: thus we have a plan for circuit Ia. Since a load can be arbitrarily complicated, a partial emitter-coupled plan can be constructed for circuit Id.

In the above discussion we often used knowledge about electronics to argue why some fragment achieved some particular function. Although the introduction of more schemata would have sufficed, it is unnecessary to have a schema for every small circuit fragment. Causal reasoning relates the schemata to the particular circuit, and can be employed to analyze small circuit fragments without the use of schemata.

CAUSAL REASONING

Propagation of Constraints

Knowledge of electronics is needed to relate a particular plan schema to a particular circuit. Although the examples of the previous section would be understandable to anyone familiar with electronics, they were stated in terms of primitive elements which were not explained by the plan theory. Another theory is needed to bridge the gap between the circuit and the primitives of the plan theory. This bridge is provided by causal reasoning. Since schemata rarely mention other schemata explicitly, causal reasoning forms the glue which holds the fragments of the plan together.

One role for causal reasoning is to determine the behavior of the particular circuit fragment. Another is to determine whether this behavior is consistent with its desired function. This involves, in large part, the accumulation of constraints on a particular circuit fragment and deducing other constraints from these. This causal reasoning must be able to record its deductions in precise and well-defined ways, for both the purpose of explaining the ultimate deductions, and for the necessity of understanding the exact relation between imposed and deduced constraints. The technique of propagation of constraints is one way to implement causal reasoning and meets all these criteria. This technique has proven extremely useful for circuit analysis in EL and INTER. Nevertheless a far more sophisticated propagator will have to be developed to deal with the problems encountered in recognition.

Qualitative Propagation

The original propagation of constraints idea in EL will have to be developed further in flexibility and ability to deal with partial knowledge. Since the recognition process involves a constant switching back and forth between causal reasoning and instantiating schemata, the causal reasoning must be able to limit its attentions to fragments of circuits and then change its analysis as the instantiating process examines the different possibilities. Much of this flexibility should be provided by a new implementation of EL in AMORD. Dealing with partial knowledge is far more

difficult and requires much more study.

EL presumes that every circuit quantity has an eventual precise numerical value, and in the course of its deductions it describes this value either explicitly as a number or as a variable. Variables provide no direct information, but when a sufficient number of constraints on a variable are discovered its value is determined. EL has very little ability to deal with partial information: a quantity is either known or unknown, and constraints provide no new information about a variable until enough are known to determine it. These constraints do not admit inequality relations. INTER, on the other hand, does not presume that it ever will know every circuit quantity precisely and thus describes each circuit quantity by a numerical range and propagates these ranges. Unfortunately, this introduction of inequality relations makes the introduction of variables and their required algebraic solution techniques very difficult to accommodate.

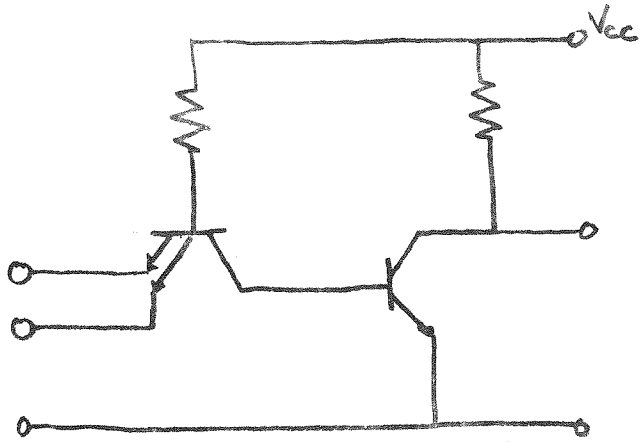
The problems of partial knowledge in the recognition process are far more difficult than these. First, we have the problem (recognized in INTER) that circuit parameters have inherent tolerances associated with them, for example, resistances can be known only to within 10% and measured values to within measurement accuracy. This is the simplest kind of partial knowledge to deal with. Second, causal reasoning should still be possible without knowing any of the circuit parameters directly. For example, in engineering textbooks one rarely finds precise component parameters on schematics, yet the engineer can usually recognize how the circuit works. Third, because causal reasoning is invoked within the recognition process, the topology of the circuit fragment being analyzed may not be precisely known.

The first problem is the simplest to deal with, and INTER suggests mechanism for this. The third problem requires that changes in topology be dealt with as easily as changes in the operating region of a transistor. This can be accomplished via the capabilities of AMORD. Before discussing the second problem in more detail it should be noted that we are not discussing propagating only at the component level: the plan fragments may be sufficiently instantiated that their behavior can be used in causal reasoning. Causal reasoning can deduce little from topology alone. However, even if no circuit parameters are given, the expectancies of the already partially instantiated schemata can provide sufficient constraints to allow causal reasoning to make useful deductions.

Causal reasoning is used in two different ways in the recognition process. The first is to test whether a particular circuit fragment can achieve a desired function. The second is to determine the behavior of a circuit fragment to see whether its behavior matches any known behavior. Even if the circuit fragment could be completely analyzed in the EL sense, this second use of causal reasoning requires a description language for circuit behavior. The first use of causal reasoning suffers from no such impediment since there the expected behavior can be explicitly specified. If the circuit fragment is not completely analyzable in the EL sense, a more sophisticated kind of causal reasoning is required.

Analysis of a circuit fragment can fail for two reasons: external circuit quantities

imposed on the fragment may be unknown or propagation of constraints may fail, as it is not complete. More often, the analysis will fail for the first reason. If further knowledge of the fragment's behavior is desired, various external inputs can be hypothesized and fragment behavior inferred from resulting propagations. Presented with an amplifier, this strategy would try various inputs, and, noticing that every constraint on one terminal produces correspondingly larger valued constraints on another terminal, it would infer that the circuit amplifies. This somewhat haphazard strategy is prone to problems, however. In particular, it provides little information on how the fragment achieved its behavior. For example, consider a simplified TTL NAND gate:



Knowing that the circuit was some kind of TTL gate, causal reasoning need only do four propagations. The output values prove that this is a NAND gate, but this analysis does not show that the first transistor actually computes the NAND and that the second transistor only acts as an amplifier. A more sophisticated kind of propagator would attempt to force propagation from some inputs and whenever an indeterminacy was reached, split the propagation into the different possibilities, thus producing a description of output behavior for various regions of input. For example, whenever a transistor's state could not be determined, all possibilities are assumed and a separate propagation carried out for each assumption.

We have been using inputs and outputs rather loosely in this discussion. However, a few simple expectancies about a particular circuit can provide a large amount of information about what are inputs and outputs. For example, knowing that a circuit is a TTL gate, inputs and outputs can be immediately determined due to the characteristics of TTL.

RECOGNITION

Theory

Recognition relates the particular object under examination to what is known about such objects in general. More specifically, recognition recaptures the plan the designer originally had in mind. Recognition and design are both instantiation processes. Design instantiates plan schemata in the library in an attempt to achieve some goal, while recognition instantiates the

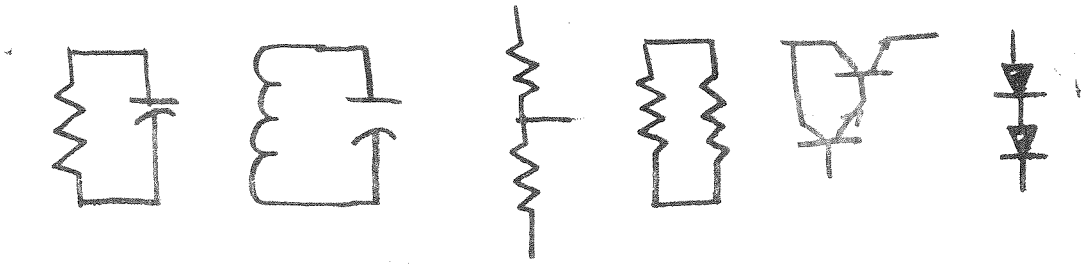
schemata and matches these to the circuit schematic in an attempt to find the schema which best explains the circuit. The result of the recognition process is a fully instantiated plan schema or plan fragment description -- the plan of the circuit.

This process is analogous to parsing English sentences. In both, the goal is to analyze a complex structure, the sentence or circuit, in order to better understand its meaning. There are also many differences between these two processes. Linguistic parsing employs grammars to encode structure, while plan recognition uses schemata. The words of a sentence have an explicit linear ordering, while it is impossible to impose any order on the components of a circuit. However, there are two important reasons why the parsing of circuit diagrams is much easier than parsing English sentences. The ordering of English is the only relation between words and is relatively weak, while the wires of a circuit diagram explicitly denote which components interact directly with each other. The technique of propagation of constraints provides a nearly complete model to test the validity of possible parses; there is no such model for English.

Before discussing a tentative structure for the recognizer, the various kinds of evidence available from a circuit will be summarized. The schematic diagram provides three main kinds of evidence: topological, functional and geometric.

Topological Evidence

The topology of the circuit consists of the actual components of the circuit and their interconnections with no concern about how these components are arranged on the schematic or how they behave. This topological evidence is easily represented by a graph. The recognizer could have a graph matcher which tries to match the graph of a given circuit to stored plans and plan fragments, but this would fail unless the circuits were identical. Further, the best match problem on graphs is extremely difficult. Therefore if the recognizer employs any direct match at all, it should be used only at the lowest level, recognizing only simple common combinations or *phrases*.



Although it would be hopeless to attempt to match the graph of a new circuit to every circuit that has been seen before, the schemata are relatively few in number and can be employed for matching. A schema contains a description of the essential topological features that any of its instantiations must contain. Although the schemata cannot describe all their instantiations directly,

a great many possibilities can be ruled out, and hypotheses generated by simple matching of the schemata with the given circuit topology.

The most basic topological property is that of connectivity, and the terminals (components and subschemata) of a schema are always connected. Most schemata have components and thus the components of a particular circuit provides an index into the library for possible matches. These two rather elementary topological features already provide a tremendous reduction in the size of the search space. Combined with geometric evidence, topological instantiation should be a relatively efficient process.

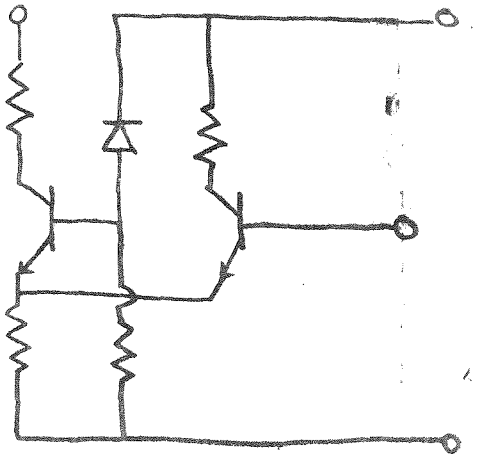
Functional Evidence

Although topological evidence can be used to determine possible schemata for the circuit fragment, causal reasoning is needed to validate the tentative instantiations. This process involves extracting functional evidence about the circuit. There are three different ways functional evidence can be used in the recognition process. Instantiating the topology of a schema results in instantiating its behavioral description; causal reasoning must be used to confirm that the behavior of the fragment is consistent with this behavioral description. The topology of a schema is in terms of other circuit fragments which must achieve some particular function; causal reasoning checks candidate subfragments. Sometimes topological evidence alone does not suggest any schemata; then causal reasoning must be used to determine the fragment's behavior and this observed behavior matched against the behavior of known schemata.

There is another use of functional evidence, which is seen in people parsing circuits, that this theory does not immediately account for. Our theory uses causal reasoning to examine *behavior* and then compares this behavior against other facts. However, causal reasoning can also be used to determine the *mechanism* by which the circuit achieves that behavior. This mechanism provides an extremely powerful way of describing schemata. In fact, mechanism transcends the distinctions between different domains. Consider, a mechanical engineer parsing a negative-feedback amplifier. He has never seen a negative-feedback circuit before, so he causally reasons about the circuit and notices output linearity is maintained by feeding a fraction of the output into the input. He is familiar with this idea from mechanical engineering and instantly realizes what other parts of the circuit to look at and what the limitations of the amplifier are. Even remaining within the circuits domain it could be argued that mechanism is as important a description scheme as behavior: is the essential structure of a superheterodyne that it is an instantiation of a particular schema, or that it uses the heterodyning principle. It is clear that a complete theory of recognition should employ both behavioral and mechanism descriptions. Unfortunately, it is unclear as to what a mechanism description might look like. For this reason this research will refrain from considering mechanism until the exact limitations of a recognizer based only on behavior are determined.

Geometric Evidence

A circuit diagram provides much more than just topological evidence about the circuit. For example, it is well known how difficult it is to analyze a scrambled circuit diagram. The engineer who draws up a circuit diagram employs many general conventions which make the schematic easy to understand. Components which are connected together are usually drawn close together. The engineer tries to make the topology as planar as possible. Signals tend to flow from left to right in relatively horizontal fashion. General busses such as ground and power-supply are usually drawn as horizontal lines. The ground is usually at the bottom and power at the top. Of particular interest is the fact that schemata are usually drawn in similar ways. For example, the emitter-coupled pair is almost always drawn as two back-to-back transistors; rarely is it drawn as:



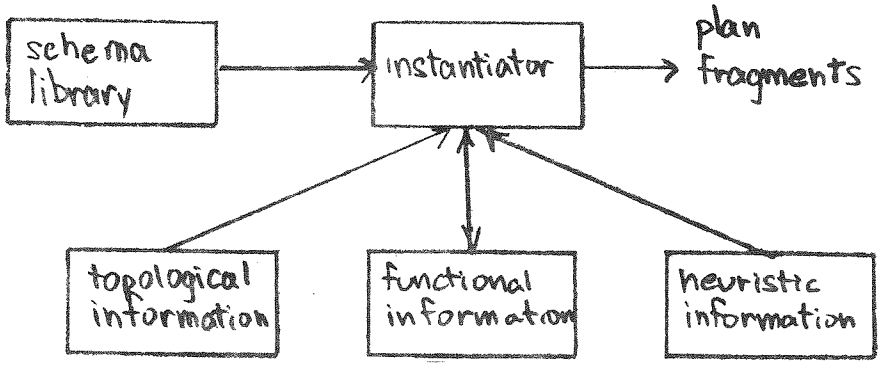
This evidence is extremely useful to a circuit recognizer. In particular, the recognizer will try to incorporate knowledge about common ways to draw instantiations of particular schemata.

Since circuit schematic conventions are so universal, geometric evidence is potentially the strongest kind of evidence the recognizer has available. Unfortunately, the extent to which these conventions are followed depends on the skill of the designer, and, furthermore, not all of the conventions are necessarily satisfiable in any particular circuit. Therefore, although geometric evidence should be utilized to a great extent in recognition, it can not be completely relied on, and thus is only a good heuristic.

A plausible scheme is to use geometric evidence to order schemata suggested by topological evidence. Consider the emitter-coupled pair. Rudimentary topological evidence proposes a push-pull amplifier, a two-stage amplifier, a darlington pair, as well as an emitter-coupled pair. If the schematic was drawn in a haphazard way, each of these possibilities would have to be examined in more detail. However, if the schematic was drawn in the conventional way, geometric evidence would suggest that the emitter-coupled pair be tried first. In this scheme, well drawn schematics are recognized quickly, while those drawn poorly take much longer, but are still eventually recognized.

The Structure of the Recognizer

The flow of information within the recognizer can be summarized by a simple block diagram:



The recognizer will use both top-down and bottom-up techniques. Top-down techniques are more powerful, since parent structures impose many restrictions on possible subschemata. The recognizer will therefore be biased towards top-down techniques. The four main top-down entry points of the recognizer are: parse circuit fragment x as schema y, parse circuit fragment x as having function y, discover behavior of circuit fragment x, and discover schema of circuit fragment x. The latter two entry points are included for completeness and for the limited cases in which the recognizer calls itself recursively. We will not consider the very general problem of recognizing the plan for an arbitrary circuit.

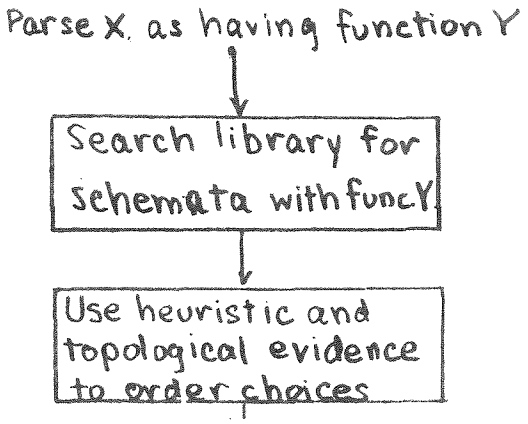
Bottom-up techniques involve discovering plan fragments before their parents have been identified. This helps limit the possibilities for the parent schema. Sometimes top-down techniques may fail completely and bottom-up techniques must be employed. More often the recognizer will have to make an explicit choice of which techniques to employ. Bottom-up techniques are necessary when attempting to recognize the plan for a circuit without having been given its function. The bottom-up techniques are based on geometric evidence and causal reasoning.

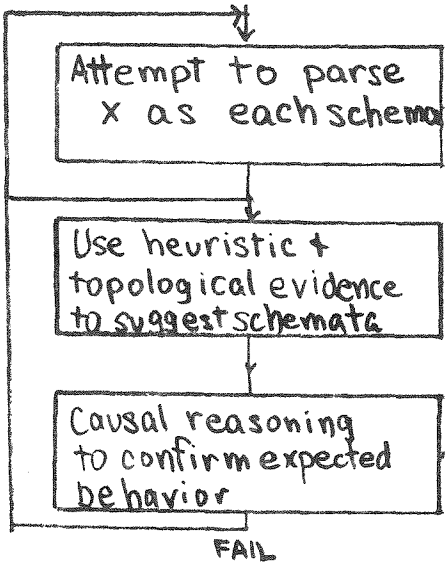
Recognition uses causal reasoning to confirm instantiations are consistent with their behavioral descriptions, to confirm the desired behavior of circuit fragments for which no schemata are known, and, in the case where no top-down information is given, to determine the behavior of a circuit fragment in order to suggest which possible schemata might match. In the first use of causal reasoning the schemata have been completely instantiated, and thus causal reasoning is fairly inexpensive since it can work with the behavioral descriptions of the sub plan fragments instead of the circuit components. However, in the second two cases the fragment under consideration has no schema, and causal reasoning can become very expensive. In these cases,

before applying causal reasoning, bottom-up techniques should be employed to recognize as many sub fragments in that fragment. This gives causal reasoning the simplest possible topology to work with.

Propagation of constraints has particular difficulties in dealing were certain kinds of schemata. For example, it cannot adequately analyze resistors in series, voltage dividers, and emitter-coupled pairs. The original EL program had to be explicitly told where these schemata occurred in the circuit if it was to successfully analyze the circuit. Therefore the recognizer should always attempt find such schemata in the circuit before attempting any causal reasoning.

Having discussed the techniques the recognizer has available to it we can examine a possible control structure. Consider the problem of parsing a particular circuit fragment x performing function y. First, the schema library is examined to find those schemata which could possibly achieve the desired function. Then geometric and topological evidence is used to prune these possibilities, and suggest which schemata are most likely to match. The recognizer then tries to parse the circuit fragment as each of the possible schemata. If the schema library does not contain any schemata whose behavior matches the desired function, bottom-up techniques can be used to suggest possible schemata that may match. If no possible schemata are discovered or every schema suggested fails to match, causal reasoning must be employed to see if the given circuit fragment at least achieves function y. Since the circuit fragment has no structure imposed on it, the causal reasoning will be very difficult and thus should be left until as much as possible of the rest of circuit is parsed. Then when causal reasoning is employed, particular attention should be paid to those components which have not yet been assigned any purpose.

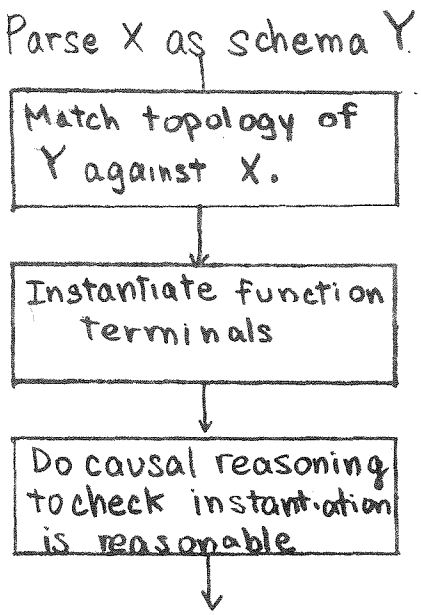




This discussion demonstrates that the recognizer need not have a schema for every part of the circuit. This is important in accommodating unexpected features in circuits. Unfortunately, employing causal reasoning without a schema can be very expensive. In particular, for causal reasoning to succeed it should be given a simple topology, and that requires bottom-up recognition within the circuit fragment to identify composite behavior of collections of components.

The recognition process is largely an instantiation process; that of trying to instantiate schemata which best match the given circuit. When the recognizer is asked to recognize a circuit fragment as a particular schema, it tries to identify an instantiation of that schema with the circuit fragment. To do this the topology of the schema is matched against the circuit fragment. If this match fails, it cannot be of the given schema type. Next the terminals of the schema are expanded. Each terminal is in terms of a desired function, and the recognizer invokes itself recursively to attempt to find a schema in the circuit fragment which achieves the given function. After the schema has been completely instantiated, causal reasoning checks that the circuit fragment referred

to by the plan fragment actually behaves as described. This is necessary because the very generality of the schemata language admits plan fragments that may not achieve their stated behavior. For example, an extra wire shorting the output of an amplifier would probably not be detected until the final causal reasoning check.



The previous discussion of the recognizer has superficially discussed many details which will have to be thought out much further. One detail which deserves some attention here is the choice of circuit fragment on which to invoke the recognizer recursively. Specifying a circuit fragment is extremely difficult in general. Probably a large section of the circuit would be passed along with an annotation mentioning which fragments already have been assigned purposes and thus most likely do not have a second purpose. When the recognizer calls itself recursively to instantiate a particular schema in a circuit fragment, it need not exhaust all the components in the circuit fragment. Only if this were the top level question would that be necessary. However, that topology of the schema that has already been identified with the circuit fragment provides boundary conditions on the fragments that match for a particular function. In the example of the emitter-coupled pair, the collector circuit needs something that functions as a load between the collector and the power source. This load can be found anywhere in the circuit fragment, but the recognizer will try to find the smallest possible fragment which matches and connects the collector and the power source. Even though the circuit fragment within which the schema must be found is not clear, these boundary conditions impose sufficient constraints to effectively limit the search space.

EXPLANATION

Explanation in General

The result of the recognition process is a plan for the circuit which can be used in troubleshooting and design. The use of plans for these two purposes has been investigated by other researchers [Brown 76] [McDermott 76]. Unfortunately, the use of plans in troubleshooting has not yet been implemented. Therefore, the test of the recognizer is to what extent it can answer questions about the circuit and explain its deductions. It is important to distinguish between two kinds of questions: those that ask about the actual circuit just recognized and those about the recognition process itself. We propose to investigate only the former.

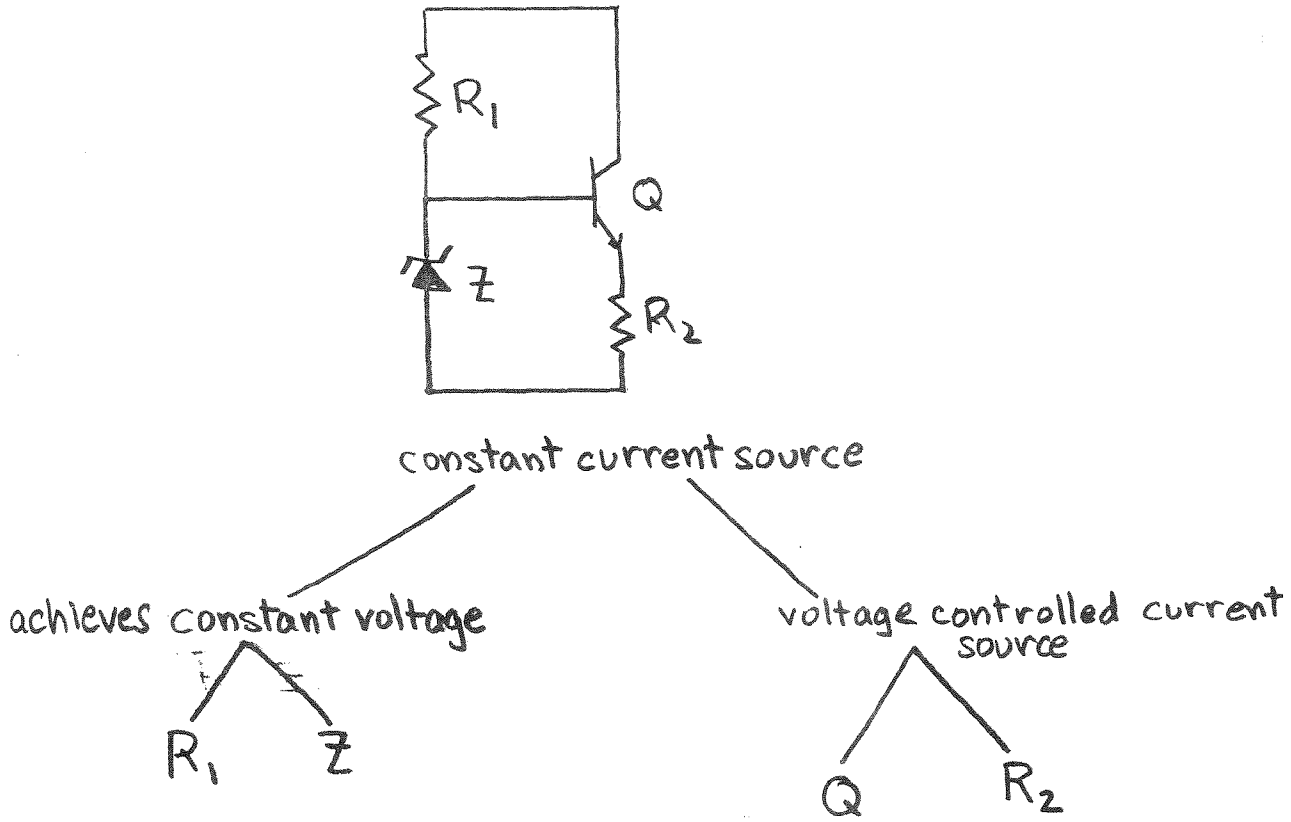
Throughout this paper the choice of techniques has been governed by the necessity to make explanations. Propagation of constraints has been shown to admit very detailed explanations. Similarly, the plan expresses circuit teleology directly so that explanations can be easily generated from it.

The proposed program is far from being a general purpose circuit expert which can be used with students. For instance, it lacks many human engineering interfaces. It can neither understand or reply in English. Both of these are major research topics in their own right. A more serious limitation is that a tutor has to have a model of the student indicating what the student does and does not know so that appropriate explanations can be given. This is also a topic of current research [Goldstein 77]. In short, our recognizer will produce explanations which are primarily oriented to people familiar with electronics or to other programs, such as tutors. It is the tutor's responsibility to further interact with the expert to produce an explanation appropriate for the student. In the proposed recognizer the explanation facility will be included, but it will converse in a rather ad-hoc fashion.

Answering Questions About a Circuit

This section describes some of the different kinds of questions that can be answered once the plan has been discovered. For expository reasons the examples will be in English, although this research does not propose to construct such a well-engineered human interface. After the recognizer has been presented the schematic and has constructed the plan for the circuit, we are in a position to ask questions about that plan.

We will discuss simple "how", "why" and "what" questions. Consider a possible parse for the simple constant-current source:



The user can ask questions how the circuit achieves its purpose. This can be answered by refering to the particular plan fragment.

> How does the circuit provide a constant current?
>
> One possible constant-current source is a constant-voltage source connected to a voltage controlled current source. R1 and Z function as a constant-voltage source and Q and R2 function as a voltage controlled current source.

Now the user can step downwards through the plan:

> How do R1 and Z provide a constant voltage?
>
> ...

In general, the "how" question requests an explanation of how the internal behavior achieves the desired function. This implies looking downward in the plan structure.

Why is R1 and Z a constant-voltage source?

To provide a constant voltage to the constant-current source schema which also uses a voltage-controlled current source. Why is Z in the circuit? Assuming a sufficient amount of current is flowing through Z it has a constant voltage across it and thus with R it forms a constant-voltage source.

The "why" question asks for the purpose of a fragment and thus looks up the plan to see what function it serves in parent schemata.

What are Q and R2?

A voltage controlled current source.

"What" questions are fairly straight-forward and can be answered by looking at the appropriate plan fragment node in the plan.

This discussion of question types is admittedly very superficial and ignores the deep difficulties of understanding questions. It does, however, demonstrate some of the fundamental ways in which a plan can be used to answer questions. There are two difficulties which are worth further discussion here. One is that the recognizer may not necessarily have to have a schema for every fragment of the circuit. In these cases, direct causal reasoning has to be used to construct the plan, as seen in the second "why" question. The schemata themselves encode a large amount of compiled causal reasoning, and if the user does not know some particular schema he may prefer this causal explanation instead. This may require extra causal reasoning on the circuit. The second difficulty concerns how the user is to refer to particular fragments. All the examples mentioned the fragment explicitly, but the fragment can often be referred to indirectly:

What is connected to the emitter of Q?
Why is there a constant voltage at the base of Q?

These difficulties, among many others, will have to be resolved if the expert is to be successfully used in a tutoring environment.

THE PROPOSED RESEARCH

Ultimate Goals

This research has two main goals. The first is to develop a theory of plans which can be used for explaining, troubleshooting and designing circuits. The second is to construct a recognizing system which can discover a plan of a circuit from its schematic. One test of whether this recognition is successful will be how well the program can answer questions about the circuit.

This research intends to apply the ideas to a wide range of DC circuits. In particular, the recognizer will be expected to construct plans for power-supplies, amplifiers, and TTL gates.

Intermediate Milestones

In this final section we will present some intermediate milestones which will have to be reached if the goals of the research are to be achieved. This research draws heavily on the work of Brown, McDermott, Sussman and Doyle. In particular it will rely directly on a new deduction system (AMORD) currently being developed by Doyle. This system is an outgrowth of some of the ideas of EL and NASL and supports many techniques for reasoning about electronics, in particular causal reasoning. With this as a starting point, we will work towards to goal of a circuit recognizer.

Theory of Schema Topology

Although this paper has outlined many of the features of plans, more research is required. The plan theory will evolve as we try to use it. We will first develop a theory for expressing just the topology of the schemata.

A Small Library of Topological Schemata

We will eventually consider a large subset of DC circuits. The theory of schema topology will be tested on a simple class of circuits -- voltage and current supplies. Included in the library will be the possible functions for each schemata so that causal reasoning will not be required in the instantiation process.

A Pure Top-Down Topological Instantiator

Using the schemata in the library, a top-down parser will be constructed. Since the function of the schemata will always be explicitly found in the library, causal reasoning is not needed to produce tentative parsings. The parsings will, however, only be tentative since causal reasoning is needed to confirm them.

Geometric Evidence

For the simple subset of circuits in the library, a theory of geometric information evidence will be developed and included to aid the top-down parser.

Bottom-Up Techniques

Bottom-up techniques based on topological and geometric evidence will be developed. Limited bottom up parsing is required for propagation of constraints to succeed.

Qualitative Propagator

We will develop a flexible qualitative propagator to perform causal reasoning.

Theory of Schema Behavior

To do causal reasoning about circuit fragments, their behavior must be describable. A language must be developed for this. The instantiation process instantiates both the topology and the behavior of a schema. With a theory of behavioral descriptions, behavioral instantiation can be developed.

Hypothesis Evaluation

Once the expected behaviors of fragments are describable, causal reasoning (qualitative propagations) can be done to check hypothetical parsing based on topological and geometric evidence. Causal reasoning can also use the behaviorial descriptions of subcircuit fragments.

Extrinsic and Intrinsic Behavioral Descriptions

Schemata describe their terminals by function, and the instantiation process has to find other schemata that can fulfill these functions. The schema terminal describes the required extrinsic behavior of the subschema, but the schemata themselves are described intrinsically. The instantiation process must match required extrinsic descriptions against intrinsic descriptions. In the beginning of this research, this problem will be finessed by storing appropriate extrinsic descriptions in the library also. With a theory of behavioral description, the mechanism for matching extrinsic descriptions against intrinsic descriptions can be explored.

REFERENCES

[Brown 74]
Brown, A.L., "Qualitative Knowledge, Causal Reasoning, and the Localization of Failures -- a Proposal for Research", Artificial Intelligence Laboratory, WP-61, Cambridge: M.I.T., 1974.

[Brown 76]
Brown, A.L., "Qualitative Knowledge, Causal Reasoning, and the Localization of Failures", Artificial Intelligence Laboratory, TR-362, Cambridge: M.I.T., 1976.

[Brown & Sussman 74]
Brown, A.L. and G.J. Sussman, "Localization of Failures in Radio Circuits a Study in Causal and Teleological Reasoning", Artificial Intelligence Laboratory, AIM-319, Cambridge: M.I.T., 1974.

[Brown *et.al.* 74]
Brown, John Seely, Richard R. Burton and Alan G. Bell, *SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting (An example of AI in CAI)*, Final Report, B.B.N. Report 279, A.I. Report 12, Cambridge: B.B.N., 1974.

[Brown *et.al.* 76]
Brown, John Seely, Richard Rubinstein and Richard Burton, *Reactive Learning Environment for Computer Assisted Electronics Instruction*, B.B.N. Report 3314, I.C.A.I. Report 1, Cambridge: B.B.N., 1976.

[de Kleer 76]
de Kleer, Johan, "Local Methods for Localizing Faults in Electronic Circuits", Artificial Intelligence Laboratory, AIM-394, Cambridge: M.I.T., 1976.

[Doyle 76]
Doyle, J., "The Use of Dependency Relationships in the Control of Reasoning", Artificial Intelligence Laboratory, WP-133, Cambridge: M.I.T., 1976.

[Doyle, Steele & Sussman 77]
Doyle, Jon, Guy L. Steele and Gerald Jay Sussman, "AMORD A Dependency Based Problem Solving Language", forthcoming.

[Fikes, Hart & Nilsson 72] Fikes, R. E., P. E. Hart and N. J. Nilsson, "Learning and Executing Generalized Robot Plans", *Artificial Intelligence 3*, Winter, 1972.

[Goldstein 74]
Goldstein, Ira, "Understanding Simple Picture Programs", Artificial Intelligence Laboratory, TR-294, Cambridge: M.I.T., 1976.

[Goldstein 77]
Goldstein, Ira, "The Computer as Coach: an Athletic Paradigm for Intellectual Education", Artificial Intelligence Laboratory, AIM-389, Cambridge: M.I.T., 1977.

[Hewitt 71] Hewitt, Carl, "Description and Theoretical Analysis (using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot", Artificial Intelligence Laboratory, AIM-251, Cambridge: M.I.T., 1971.

[McDermott 76]
McDermott, D.V., "Flexibility and Efficiency in a Computer Program for Designing Circuits", Artificial Intelligence Laboratory, TR-402, Cambridge: M.I.T., 1976.

[Miller & Goldstein 76] Miller, Mark L. and Ira P. Goldstein, "PAZATN: A Linguistic Approach to Automatic Analysis of Elementary Programming Protocols", Artificial Intelligence Laboratory, AIM-388, Cambridge: M.I.T., 1976.

[Minsky 74]
Minsky, Marvin, "Frame-Systems: A Framework for Representation of Knowledge", Artificial Intelligence Laboratory, AIM-306, Cambridge: M.I.T., 1973.

[Rulifson 72] Rulifson, J.F., "The QA4 Language Applied to Robot Planning", FJCC, 1972.

[Rich & Shrobe 76]
Rich, C. and H.E. Shrobe, "Initial report on a LISP Programmer's Apprentice", TR-354, Cambridge: M.I.T., 1976.

[Sacerdoti 75]
Sacerdoti, Earl D., "The Non-Linear Nature of Plans", Stanford Research Institute, A.I. Group Technical Note 101, 1975.

[Stallman & Sussman 76]
Stallman, R.M., and G.J. Sussman, "Forward Reasoning and Dependency-Directed Backtracking In a System for Computer-Aided Circuit Analysis", Artificial Intelligence Laboratory, AIM-380, Cambridge: M.I.T., 1976.

[Sussman 73]
Sussman, G.J., "A Computational Model of Skill Acquisition", Artificial Intelligence Laboratory, TR-297, Cambridge: M.I.T., 1973.

[Sussman & Stallman 75]
Sussman, G.J., and R.M. Stallman, "Heuristic Techniques in Computer Aided Circuit Analysis", Artificial Intelligence Laboratory, AIM-328, Cambridge: M.I.T., 1975.

[Waters 76]
Waters, Richard C., "A System For Understanding Mathematical FORTRAN Programs", Artificial Intelligence Laboratory, AIM-368, Cambridge: M.I.T., 1976.