A PARADIGM FOR BUILDING DIAGNOSTIC EXPERT SYSTEMS BY SPECIALIZING GENERIC DEVICE AND REASONING MODELS[1]

M. Hofmann, G. Collins, J. Vargas,
J. Bourne, and A. Brodersen

Box 1804, Station B,
The Center for Intelligent Systems
Vanderbilt University,
Nashville, Tennessee 37235

ABSTRACT

This paper describes a generic schematic knowledge representation, acquisition, and manipulation system (SKRAM) applied to diagnostic problem solving in analog circuits. SKRAM is a general purpose declarative, schema-oriented system composed of five layers: (1) an object-oriented implementation layer, (2) a logical layer defining object features, i.e., relations and attributes, (3) an epistemological layer at which schemata are introduced as knowledge packages, (4) a semantic or conceptual layer where meaning is attached to schemata by defining an interpretation mechanism, and (5) a domain layer containing objects and schemata which represent knowledge of the application domain. SKRAM supports the acquisition and maintenance of general and specific knowledge and constructs models of the entities in the application domain, e.g., models of the structure and function of electronic circuits. An example of the use of the declarative representation in the domain of analog circuits is given and the architecture is contrasted with other architectures in similar and related domains.

[1]This research was supported, in part, by AFOSR Grant: AFOSR-87-0144.

A. INTRODUCTION

Although building expert systems for device diagnosis is currently one of the most popular application areas for expert systems, there remain numerous software engineering and application representation problems that need to be solved. Typical devices undergo design variations and revisions, and different types of similar devices are often used in constructing real-world systems. Expert systems designed to diagnose problems in specific types of devices are usually hard to maintain; rule bases are difficult to change, and reasoning strategies are often buried in the device-specific rules. The goal of the research reported in this paper is to design a knowledge representation, acquisition, and manipulation system which extends the traditional "expert system shell" concept with facilities to represent and maintain general domain knowledge and to specialize this knowledge to specific devices. As an example, the generic system for the diagnosis of analog electronic devices requires knowledge about the elements of the analog circuit domain and about diagnostic reasoning. A device-specific expert system is built from a declarative description of the device in terms of the generic knowledge.

All knowledge is represented in object-based semantic networks of objects and relations. In SKRAM, basic elements and useful collections of components, e.g., common circuit configurations, correspond to schemata, i.e., sub-networks, of the semantic network. In particular, these schemata represent models of the physical and functional structure and the causal dependencies of generic or actual devices, information about fault behavior and probabilities, and diagnostic strategies and plans. Control and application knowledge can be stored independently and declaratively.

SKRAM accepts knowledge input in the form of declarative statements which it translates into objects and relations in the semantic network. A parser and an interpreter analyze the input statements and a knowledge base builder creates the knowledge base for the specific system using the applicable generic schemata as exemplars.

The resulting device-specific system can be executed independently from the generic system. Execution is initiated by matching an observed situation with the control knowledge base; the best matching schema will post a goal and trigger a search for a plan by activating a method attached to a special slot in the goal object. Search is performed through matching provided by a fuzzy associative inference engine "FASIE" (Vargas 1988). Generic skeletal plans will be specialized according to the data and the components involved, and then applied. Thus, the execution of the specific systems is governed by the same paradigm as the creation of the specific systems. Again, generic knowledge is adapted to specific situations.

The modular declarative representation of knowledge at various levels of abstraction and the use of fuzzy associative reasoning form the basis for the SKRAM system described in this paper. Modularity disentangles knowledge domains. The declarative style allows the system to manipulate knowledge readily, and the abstraction hierarchy accelerates and simplifies reasoning.

B. BACKGROUND

A similar architecture was presented by Sathi et al. (1985) for a scheduling system. Knowledge representation in semantic networks has been broadly covered in Findler (1979). The epistemology of semantic networks, one type of which are networks of frames or schemata, has been addressed in that volume by Brachman (1979). Structuring domains into abstraction hierarchies and representing domain concepts declaratively has been an issue in planning system research for some time, see, for example, Sacerdoti (1979), Wilensky (1981), or Wilkins (1984).

Separation of domain and control knowledge has been universally acknowledged as increasing the clarity and maintainability of reasoning systems. For example, Stefik (1980) suggests this principle for planning systems, Pau (1986) documents it for the diagnostic domain, and all the work on device modeling, e.g., Davis (1984), Genesereth (1984), or Sembugamoorthy and Chandrasekaran (1984), enforces this modularity principle. Combining causal, model-based and heuristic reasoning has been addressed, for example, by Fink (1985).

Models of domain entities, e.g., device models in the electronics domain, have been proposed with widely varying semantics. Davis (1984) defines models for digital components which constrain the values at inputs and outputs. Unfortunately, the inverse function for the components are needed. Genesereth (1984) uses complete truth tables to model digital components. DeKleer and Williams (1986) uses qualitative reasoning to derive component behavior. None requires explicit fault models to perform diagnosis but only models for correct system behavior. Reasoning is achieved by detecting conflicts between the predicted and observed behaviors of components or circuits and trying to minimize the set of possible faulty components. Dague et al. (1987) also use models of correct behavior only, but since their work is in the analog circuit domain they supply several models for each device for the different states it can assume. Milne (1985) assigns responsibilities for parts of the analog output signal to individual components to reason about indivisible structures. Fault models are used, when faults cannot be traced along the structural decomposition. For example, Davis (1984) introduces a fault model for bridge faults. In the analog circuit domain, where complex structures are common, fault models can help to break open feedback loops and also guide the search by identifying likely failures.

We will show that our architecture is equally well suited for representing knowledge in the analog circuit domain and in control, e.g., diagnosis planning, domains. Not only are domain knowledge and diagnostic knowledge and strategies strictly decoupled but they can also be implemented in the same formalism and be interpreted by the same inference mechanism. In the next section we will present our design objectives for the system architecture, Section D describes the principles applicable to reasoning in the domains, and Section E introduces the representation methodology which supports the proposed architecture. The system architecture is described in Section F. Section G outlines the domain-specific relations needed to represent the analog circuit and the diagnosis planning domains. Section H contains an example from the analog circuit domain. In Section I we compare our approach to some of the relevant work of other researchers and in Section J we analyze the preliminary results and the directions for future improvements derived from the experience with the system architecture.

C. OBJECTIVES

Three principles have guided the design of our architecture: generality, uniformity, and modularity. Generality demands that the lower levels of the architecture are independent from the higher ones. The result is a generic knowledge processing system at the conceptual level which we use to implement a diagnostic system at the domain level, which, in turn, is specialized to deal with faults in analog circuits by adding the relevant domain knowledge.

Uniformity forces all knowledge to be stored in the same representation. In contrast to systems which define sophisticated device models but add a procedurally coded diagnosis process, we strive to implement all relevant knowledge in the framework of our representation. Therefore, models must support a wide variety of knowledge types, for example, models of device structure, function, and behavior; the characteristics of causal relationships; models of diagnostic procedures; and planning knowledge about how to combine diagnostic steps like measuring parameters or replacing components. The uniform schema representation allows us to reason with all types of knowledge via a single inference paradigm.

Modularity is important both for the im-

plementation strategy and the knowledge bases. On the system level we define a representation module, an inference module, and a control module. On the representation level, our architecture is divided into five layers according to Brachman (1979), and on the knowledge level there are an unlimited number of knowledge bases. Most importantly, there are the device models and the diagnostic models, but meta planning, explanation, and knowledge acquisition modules might be added in the future.

D. REASONING PRINCIPLES

Domain knowledge is represented as networks of interrelated concepts. Important components of this knowledge are common constellations of domain objects, i.e., a data base of prototypes of the domain. In the analog circuit domain we find a collection of commonly used subcircuits, such as differential amplifiers, filters, buffers, etc. A library of elementary components, such as operational amplifiers, transistors, resistors, etc., also exists.

The domain prototype descriptions incorporate knowledge of how to decompose objects, relate object structure to behavior and function, and heuristics about the object, e.g., fault probabilities, costs, and pointers to relevant diagnostic strategies which are stored separately in their own knowledge base. In the analog circuit world we do not have to use qualitative modeling to derive this information but instead match the structure of a particular device to the known structures in the library. There we find behaviors and common functions of circuits. We also propose to use fault models, partly to constrain the search for the fault and partly to direct the search towards the most promising alternative. We achieve a combination of model and heuristic-based reasoning unlike systems such as Fink's (1985), which use different subsystems for the two kinds of reasoning.

We feel that our approach is similar to the performance of an expert technician who recognizes subcircuits in the schematic which perform familiar functions and uses his/her experience with these entities to guide the diagnosis. It is important to remember, however, that the basic reasoning system architecture does not dictate any decisions about what kind of knowledge to represent in each domain. Representing domain knowledge declaratively, using several levels of abstraction, is, on the other hand, an integral part of the knowledge representation scheme.

By defining knowledge bases of domain prototypes we establish a device-independent system. Given a description of an actual circuit to be diagnosed, the system relates the description to what it knows about circuits in general and, with some human expert assistance, creates a diagnostic expert system for the particular given device.

E. KNOWLEDGE REPRESENTATION

All knowledge is represented as structures of objets, attributes, and relations. The knowledge structures are implemented in a semantic network. The semantic network representation will be described below in terms of the epistemological layers proposed to Brachman (1979). Figure 1 illustrates the concepts defined at each layer.

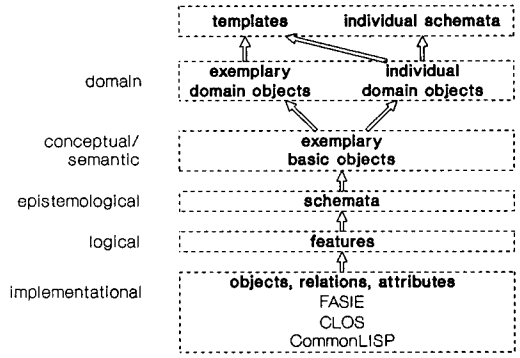

Figure 1: Knowledge Methodology Representation

The implementation layer of the representation framework is realized as a network of objects of an object-oriented language. There are three basic types of objects: concept objects, relations, and attributes. Each object, including relations and attributes, is an instance of a class in the sense of the object-oriented paradigm. However, the class definition and hierarchy do not determine the interpretation of the objects as perceived from the higher levels of the representation. The object structures are implemented as instances connected by pointers located in the instance variable slots and constitute a completely declarative knowledge representation.

In the logical layer the form of the basic knowledge structures is defined. Each relation points to a source and a target object to indicate that the source object has this relation with the target object. Attributes point to the object which has the value which they contain. Schemata are the smallest entities of the representation which can be interpreted. An object and its associated features, i.e., relations and attributes, is the kernel of the every schema structure.

At the epistemological level of the semantic network the schema is introduced as the standard unit of knowledge. A schema consists of a number of interrelated objects and their features. Schemata are used to represent perspectives of a concept. For example, a schema may represent the functional decomposition of an amplifier circuit.

The distinction between exemplary and individual objects is introduced at the logical or

semantic layer. Exemplary objects are used in schemata which represent typical or template concepts while individual schemata are made up of individual objects and correspond to real world entities. Also, the intrinsic characteristics of objects with respect to the knowledge base manipulation mechanisms are introduced here. Most importantly, the set of generic relations which support the construction of schemata for all application domains are defined. The generic relations specify transitivity and the inheritance semantics between schemata.

In the domain layer knowledge about the domain is formulated into domain models, e.g., electronic device and circuit models, represented as schemata. These schemata are more complex units of information than the simple relations mentioned above, similar to scripts (Schank and Riesbeck, 1981) but smaller grained. Exemplary schemata or templates contain typical or template concepts, e.g., typical amplifier configurations and their behavior. Templates can be created and augmented through the inheritance of features from other templates, mediated by the inheritance semantics of the relations between the templates. The most important and powerful inheriting relation is the "is-a" relation between schemata. Individual schemata can be derived from templates by the process of individuation which introduces specific data into the models. During the individuation process schemata can be nested until the desired granularity of the representation is reached.

F. ARCHITECTURE

The system architecture is organized into the following components: the domain models implemented in the representation scheme described above, a knowledge acquisition and maintenance module, an inferencing paradigm DECODIT (declarative knowledge controlling domain-independent tasks), and an executive paradigm MOLAR (modular layered control architecture).

The knowledge acquisition and maintenance module contains a parser, an interpreter, and a builder. The parser accepts statements which describe the domain in terms of objects, relations, and attributes according to a simple grammar. The interpreter determines what task to perform with the referenced knowledge structures. Statements can lead to addition, deletions, and changes in the knowledge base. The builder is responsible for exploiting inheritance, controlling the individuation of templates, and keeping track of interdependencies in the knowledge base.

DECODIT supplies the means to apply the knowledge stored in SKRAM to problems. DECODIT defines a set of domain-independent operators which exercise the knowledge. It is supported by mechanisms which retrieve data from the knowledge network. In contrast to the way rules interpret facts in rule-based systems, DECODIT does not contain domain-specific knowledge or rules but generic operators, such as "simulate," "execute," "explain," etc. This is possible because the inferential knowledge is stored explicitly and declaratively in the knowledge representation scheme. For example, planning knowledge can be executed which, in turn, may activate a simulation of some function of the modeled device. However, in order to test the device models before the system is complete, rule sets or procedures which operate on the represented knowledge directly may be substituted for these operators.

MOLAR is the control paradigm and can be implemented as a domain in the OBSTRUCT/SECSIST paradigm. A description of MOLAR is therefore not so much a description of a part of the architecture, but of an application domain. Although MOLAR is a standard component it can easily be changed or replaced simply by supplying a different description of control behavior. The initial version of MOLAR will contain a strategy and a planning layer. In our initial development phase MOLAR has been simulated by procedural LISP code.

The proposed architecture promises to render knowledge reusable from device to device and even among related domains because of the declarative modular representation. In contrast to production rule systems, the available knowledge is completely independent of its interpretation.

G. Analog Circuits and Diagnosis Planning

The most important relation types for the analog circuit domain are outlined below:

template	--	(circuit-)individual
template-parts	--	individual-parts
template-terminals	--	individual-terminals
circuit	--	subcircuit (structural decomposition)
circuit-terminals	--	subcircuit-terminals
circuit	--	subcircuit-components (parts)
component	--	component (connection of terminals)
template	--	subtemplate (functional decomposition)
function of whole	--	function of parts
concept	--	template (functional substitution) replace some entity by another (more complex) one: e.g., replace a resistor by a trim-pot (for zero adjust), these are "options" of the template!
components	--	generic components (similar to template circuits but not decomposable)

Interestingly, the types of relations in the diagnosis planning domain can be defined in strict analogy to the relations of the analog circuit domain enumerated above. This analogy assumes the planning will take place in an abstraction hierarchy (Sacerdoti, 1977) using skeletal plans (Friedland 1985).

templates	<->	skeletal plans (experience!)
circuit individuals	<->	actual plans
circuit decomposition	<->	plan refinement
basic components	<->	plan operators (plan steps)
signals	<->	state of the world
signal conditions	<->	preconditions, postconditions
function of circuit templates	<->	purpose of plans (goals, subgoals)

H. Example

A simple balanced to single ended operational amplifier stage will serve as an example to illustrate the relational representation technique. Figure 2 shows the circuit schematic, Figure 3 depicts the corresponding template, and Figure 4 lists a portion of the relations which describe the given circuit. This example is confined to the structure of the circuit for simplicity.

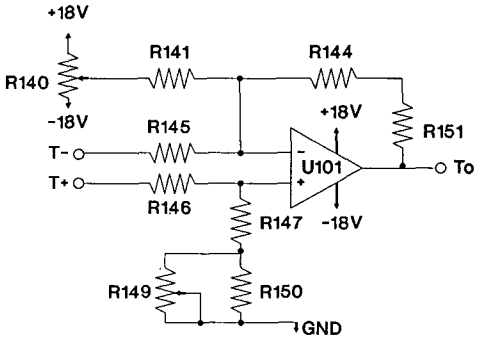

Figure 2. Amplifier Circuit Schematic

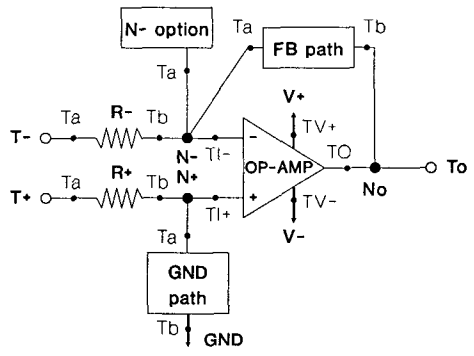

Figure 3. Operational Amplifier Circuit Template

TEMPLATE

Generic-Differential-OP-AMP (DO)

	has-terminal	T-, T+, To, V+, V-, GND
DO	has-parts	R+, R-, OP-AMP, FBpath, GNDpath, N-option
R+, R-	is-a	resistor
OP-AMP	is-a	operational-amplifier
N-, N+, No	is-a	node
N+	connects	Tb of R+, Ta of GNDpath, TI+ of OP-AMP
T-	corresponds-to	Ta of R-
T+	corresponds-to	Ta of R+

CIRCUIT

specific-OP-AMP (DO-s)

	implements	DO
R145	implements	R-
R146	implements	R+

Figure 4. Relational, Declarative Description

I. Critique

Our architecture bears the greatest resemblance to the representation paradigm described by Sathi et al. (1985). Although Sathi et al. deal with the seemingly unrelated domain of scheduling problems, they also define a five layer knowledge organization according to Brachman (1979) and define the inheritance semantics of a set of primitive relations in the epistemological layer. The close resemblance of Sathi's et al. and our knowledge representation architectures for systems intended for use in quite different domains is a good indication of the generality of our approach.

The diagnostic system presented by Dague et al. (1987) diagnoses faults in the analog circuit domain and is also implemented in an object-oriented language. We share with them the use of libraries of generic components, but we add libraries of common building blocks at different levels of abstraction and their typical implementations. Reasoning in abstraction hierarchies is an add-on in Dague's system but a basic principle in our architecture. We also try to uniformly describe all kinds of knowledge including diagnostic strategies. In the diagnostic process itself we are able to combine model and experience-based reasoning by attaching heuristics to the generic templates in the various domains. Dague et al. propose only a general-purpose diagnostic strategy.

Milne (1987) reviews the current trends in diagnostic systems and identifies four levels of diagnostic reasoning. The levels are characterized by the kind of knowledge they use to describe a device. The levels are structure, behavior, function, and compiled pattern matching. Knowledge at each level can be used to diagnose faults directly, e.g., structure can guide structural isolation techniques, but can also be transformed into the next higher level model. For example, DeKleer and Williams (1986) derive behavior and function from structure and use the resulting models to predict behavior. Our

architecture proposes to link structural descriptions to behavioral and functional models by mapping the structural description of a circuit to an exemplary circuit which stores these models. Looking up standard exemplars works well in the analog circuit domain where designers most often use similar exemplars to design devices. We are thus able to replace the complex step of qualitative simulation to derive functional and behavioral models by a matching procedure which in the first stage of the process will be accomplished by a human expert. Later we will automate this step. The exemplary models also carry heuristic information about the kinds of failures to expect and how they usually manifest themselves.

J. Conclusion

Ad hoc implementations of expert systems have always been plagued by brittleness and maintenance problems. Diagnostic reasoning is usually embedded in the domain knowledge. Procedural representations tend to be hard to generalize.

In this paper we have presented a declarative, uniform, and modular knowledge representation and reasoning architecture. Declarative knowledge can easily be organized into appropriate network structures. A uniform representation can accommodate knowledge domains as disparate as analog circuits and diagnosis planning. A modular layered architecture dispels representational ambiguity while separate knowledge modules can encapsulate domain knowledge chunks. Domains are organized into levels of abstraction and commonly used concepts are stored explicitly as exemplary schemata or templates. This architecture aids in establishing representational integrity and strives to model an expert technician's reasoning process. Although some aspects of the system remain incomplete, we feel that the system holds great promise for solving some of the most common problems encountered with traditional expert systems.

K. References

Brachman, R.J., "On the Epistemological Status of Semantic Networks," in Findler, N.V., (ed.), pp. 3-50, 1979.

Dague, P., Raiman, O., and Deves, P., "Troubleshooting: When Modeling is the Trouble," Proc. AAAI-87, Morgan Kaufman, California, pp. 600-605, 1987.

Davis, R., "Diagnostic Reasoning Based on Structure and Behavior," Artificial Intelligence, Vol. 24, pp. 347-410, 1984.

DeKleer, J., and Williams, B.C., "Reasoning about Multiple Faults," Proc. AAAI-86, Morgan Kaufman, California, pp. 132-139, 1986.

Findler, N.V., (ed.), Associative Networks, Academic Press, New York, 1979.

Fink, P.K., "Control and Integration of Diverse Knowledge in a Diagnostic Expert System," Proc. IJCAI-9, pp. 426-431, 1985.

Genesereth, M.R., "The Use of Design Descriptions in Automated Diagnosis," Artificial Intelligence, Vol. 24, pp. 411-436, 1984.

Milne, R., "Fault Diagnosis through Responsibilities," Proc. IJCAI-9, Morgan Kaufman, California, pp. 423-425, 1985.

Milne, R., "Strategies for Diagnosis," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-17, No. 3, pp. 333-339, 1987.

Pau, L.F., "Survey of Expert Systems for Fault Detection, Test Generation, and Maintenance," Expert Systems, Vol. 3, No. 2, pp. 100-111, 1986.

Sacerdoti, E.D., A Structure for Plans and Behavior, Elsevier, New York, 1977.

Sathi, A., Fox, M.S., and Greenberg, M., "Representation of Activity Knowledge for Project Management," PAMI-7, No. 5, pp. 531-552, 1985.

Schank, R.C., Riesbeck, C.K., Inside Computer Understanding, Erlbaum, New Jersey, 1981.

Sembugamoorthy, V., and Chandrasekaran, B., "Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems," technical report, Ohio State University, 1984.

Stefik, M., "Planning and Meta-Planning," HPP-80-13, Stanford University, June 1980.

Vargas, J.E., Hofmann, M.O., Bourne, J.R., Brodersen, A.J., and Collins, G.C., "Similarity-Based Reasoning about Diagnosis of Analog Devices," this issue.

Wilensky, R., "Meta-Planning: Representing and Using Knowledge about Planning in Problem-Solving and Natural Language Understanding," Cognitive Science, Vol. 5, pp. 197-233, 1981.

Wilkins, D.E., "Domain-Independent Planning: Representation and Plan Generation," Artificial Intelligence, Vol. 22, pp. 269-301, 1984.