Expert Systems With Applications, Vol. 3, pp. 303–315, 1991
Printed in the USA.

0957-4174/91 $3.00 + .00
© 1991 Pergamon Press plc

Knowledge Acquisition Based on Explicit Representation

JHYFANG HU

Tulane University, New Orleans

JERZY W. ROZENBLIT

University of Arizona, Tucson

Abstract—*In recent years, knowledge-based systems have become one of the most popular approaches for solving engineering problems. Conventionally, knowledge acquisition and representation are treated as separate processes. Knowledge engineers are responsible for preparing question patterns and setting up personnel interviews with domain experts for knowledge acquisition. All acquired knowledge is manually interpreted, verified, and transformed into a predefined representation scheme. In this paper, a new acquisition method, called knowledge acquisition based on explicit representation (KAR), is proposed to facilitate the elicitation of design knowledge. KAR approach employs a structured knowledge representation scheme to drive the acquisition process. By exploiting the structured nature and well-defined axioms of the representation scheme, acquisition queries and verification rules are generated automatically. The acquired knowledge is verified and transformed into a format ready for reference. The proposed framework is expected to reduce the cost and effort of the knowledge base development process. To illustrate KAR, a VLSI interconnect design example is presented.*

1. INTRODUCTION

KNOWLEDGE ACQUISITION is commonly recognized as the main bottleneck in design of expert systems. Several methodologies such as interviewing, protocol analysis, observation, induction, clustering, and prototyping, have been introduced that help elicit knowledge from experts (Gaines, 1987, 1988; Hart, 1985; Kessel, 1986; Olson & Rueter, 1987; Ritchie, 1984; Waterman & Newell, 1971). The most common method for knowledge acquisition is an interview (Hart, 1985). In face-to-face interviews, experts are asked questions and are expected to provide informative answers. All details of the interviews are recorded for further manual analysis and conversion so that essential knowledge can be extracted and translated into a desired representation. Problems have been discovered with the interview approach. For example, knowledge engineers do not know what questions to ask; knowledge engineers are unable to identify the essential knowledge; experts are unable to spell out their knowledge; or experts misunderstand questions asked by knowledge engineers because of different interpretations of the terminology. The problems listed above often result in the acquisition of conflicting knowledge. Some of the problems associated with the interview technique are resolved by a more structured approach termed protocol analysis (Waterman & Newell, 1971). With the protocol analysis, experts comment on specific examples from a given problem domain. For example, experts may look at a specific design example and comment on the question: "Why is this design good (bad)?". This is different from the interviewing approach, which may tackle the same problem by asking question patterns (e.g., "What made this design good (bad)?"). Often, it is easier to comment on a specific example in protocol analysis rather than to answer the general questions in the interviewing process. In protocol analysis, knowledge is extracted by detecting general patterns, for example, experts may always examine one particular characteristic first.

Computer induction (Ritchie, 1984) is another technique. With induction, experts provide a set of example cases called training sets, together with attributes of such sets. Then, a program is applied to induce rules from those training sets. The quality of the induced knowledge depends on the selection of training sets, attributes, and the induction algorithms.

Much of the difficulty in knowledge elicitation is caused by experts who cannot easily describe how they view a problem. This is essentially a psychological problem. Kelly viewed a scientist as a human being categorizing experiences and classifying his own en-

Requests for reprints should be sent to: Jhyfang Hu, Department of Electrical Engineering, Tulane University, New Orleans, LA 70118, USA.

vironment. Such a description is very suitable for an expert in his knowledge domain. Based on Kelly's personal construct theory (Kelly, 1955), the repertory grid technique (Boose, 1988) was developed for knowledge acquisition. Given a problem domain, experts build a model consisting of elements and constructs which are considered relevant and important. The constructs are similar to attributes except that they must be bipolar (e.g., good/bad, true/false, strong/weak). Elements are analogous to examples in induction. The grid is a cross-reference table between elements and constructs. For example, in acquiring knowledge for evaluating computer programs, experts may build a grid table with a number of programs (elements) and attributes (constructs) such as modularity, testability, portability, meaningful variables, and readable layout. Each square of the grid table is then filled with a quality value or index. Finally, the quality of programs is determined by experts based on the summing quality index.

None of the methodologies discussed above is commonly accepted. For acquiring complex knowledge, especially in engineering design applications, a more efficient acquisition method is needed. In this paper, an acquisition process called knowledge acquisition based on explicit representation (abbreviated KAR) is proposed. This approach is particularly suitable for eliciting knowledge required in system design problems. Thus, requirements for representing design knowledge are discussed first. Then, a representation scheme developed specifically for design applications is presented. We show how the proposed structured representation scheme can be used to guide the acquisition process. Acquisition query rules are generated based on the axioms which define the representation scheme and relationships among the objects encoded by the scheme. This provides a systematic and formalized approach to capturing knowledge about the system being designed. A comprehensive example of knowledge acquisition for VLSI packaging is used to illustrate the conceptual framework.

2. REPRESENTATION FOR DESIGN

Structured techniques based on the concepts of hierarchy, modularity, and regularity reduce the complexity of the design process (Weste & Eshraghian, 1985). The use of hierarchy involves dividing a system into subsystems and then repeating this operation on the subsystems until their complexity is at an appropriate or desired abstraction level. A modular design approach facilitates flexibility, team efforts, and future modifications. Regularity denotes the use of a regular structure at all design levels. To increase the efficiency of knowledge management and processing, design knowledge must be organized in a way that reflects decomposition, hierarchy, modularity, and regularity.

In general, to describe a system configuration, we need a structure that embodies knowledge about: decomposition, taxonomy, coupling, and attributes. Decomposition knowledge means that the structure has schemes for representing the manner in which an object is decomposed into components. By taxonomy, we mean a representation for the kinds of variants that are possible for an object, that is, how it can be categorized and subclassified. Coupling information provides communication links and indicates how component models are synthesized to form the system model. Attributes characterize static properties and dynamic behavior of the system.

In addition to declarative knowledge, procedural knowledge that can be used to support the designer in carrying out the design process is needed. Such knowledge includes rules for selecting alternative system components, procedures for evaluating performance of a design prototype, and methods for modifying and varying design parameters.

To be qualified for driving the KAR approach, the representation structure must satisfy the following requirements:

- The structure must be capable of encompassing declarative as well as procedural knowledge mentioned above. The structure must be able to denote common features used in system design.
- To facilitate the acquisition process, the structure should support well-defined axioms that enable automatic generation of query and verification rules.
- The structure should facilitate the inference process and provide efficient schemes for knowledge refinement and maintenance.

The increasing demand on the quality of knowledge-based systems has resulted in knowledge representation becoming a major topic in AI research. Although various schemes such as production rules (Newell & Simon, 1972), frames (Minsky, 1975), semantic networks (Quillian, 1968; Shastri, 1988), AND/OR trees (Nilsson, 1981), predicate logic (Chang & Lee, 1973), scripts (Schank & Abelson, 1977), and the system entity structure (Zeigler, 1984, 1986), have been introduced for knowledge representation and management, none of these schemes satisfies all the requirements that qualify a representation structure for the KAR approach. To enable KAR, the system entity structure representation has been refined by integrating it with frames and production rules. This integration results in a hierarchical, entity-based knowledge management scheme called Frames and Rules Associated System Entity Structure (FRASES) (Hu, 1989; Rozenblit, Hu, & Huang, 1989; Hu, Huang, & Rozenblit; 1989).

3. STRUCTURE OF FRASES

As a step toward a complete knowledge representation scheme for design support we have combined the decomposition, coupling, and taxonomic relationships in a knowledge representation scheme called the *system* *entity structure* (SES) (Zeigler, 1984, 1986). Knowledge representation is now generally accepted to be a key ingredient in designing artificial intelligence software. Previous work (Rozenblit, 1985; Rozenblit & Zeigler, 1986, 1988; Rozenblit, Hu, Kim, & Zeigler, 1990) identified the need for representing the structure and behavior of systems, in a declarative scheme related to frame-theoretic and object-based formalisms (Zeigler, 1987, 1990). The elements represented are motivated, on the one hand, by systems theory (Mesarovic & Takahara, 1975; Wymore, 1967) concepts of decomposition (i.e., how a system is hierarchically broken down into components) and coupling (i.e., how these components may be interconnected to reconstitute the original system). On the other hand, systems theory has not focused on taxonomic relations, as represented for example in frame-hierarchy knowledge representation schemes. In the SES scheme, such representation concerns the admissible variants of components in decompositions and the further specializations of such variants.

The interaction of decomposition, coupling, and taxonomic relations in an SES affords a compact specification of a family of system components in a given design domain. In a system entity structure, *entities* refer to conceptual components of reality for which models may reside in a model base. Also associated with entities are slots for attribute knowledge representation. An entity may have several *aspects*, each denoting a decomposition, and therefore having several entities. An entity may also have several *specializations*, each representing a classification of possible variants of the entity. Figure 1 illustrates the system entity structure of a lumped RC line model for VLSI interconnection design. It is important to realize that this representation is a specific instance of the system entity structure template, obtained after each entity and its corresponding decompositions/specialization have been identified by a VLSI design expert. The goal of KAR is to provide guidance in generating such instances.

As shown in the figure, an *RC-Line* is decomposed (|) into functional modules: *RC-Drivers/Receivers* and *RC-Segments*. Two design alternatives (specializations) (||), *Repeaters* and *Cascaded-Inverters*, are suggested for the realization of *Drivers/Receivers*. Since the number of *RC-Segments* may vary with the selection of *Drivers/Receivers*, a multiple decomposition (|||) is used for representing entities whose number may vary in the system.

The system entity structure organizes possibilities for a variety of system decompositions and, consequently, a variety of design model constructions. Its generative capability facilitates convenient definition and representation of system components and their attributes at multiple levels of aggregation and abstraction. More complete discussions of the system entity structure and its associated structure transformations are presented in (Rozenblit, 1985; Rozenblit & Zeigler, 1988; Zeigler, 1984).

FRASES is a system entity structure augmented with frame and rule representations. It satisfies the same set of axioms that apply to SES, namely:

- *Uniformity:* Any two nodes with the same names have identical attached attributes and isomorphic substructures.
- *Strict Hierarchy:* No nodes appear more than once down any path of the FRASES structure.
- *Valid Siblings:* No two nodes under the same parent have identical names.
- *Attached Attributes:* No two attributes attached to a FRASES node have the same name.
- *Alternating Mode:* The modes of a node and its successors are always different ("entity" vs. "specialization" or "decomposition"). The root is always an entity.

To encompass all required declarative and procedural knowledge for design applications, a frame object, termed *Entity Information Frame (EIF)* is associated with each FRASES node. Each *Entity Information Frame* consists of multiple slots for storing design knowledge. The number of slots can be modified to fit

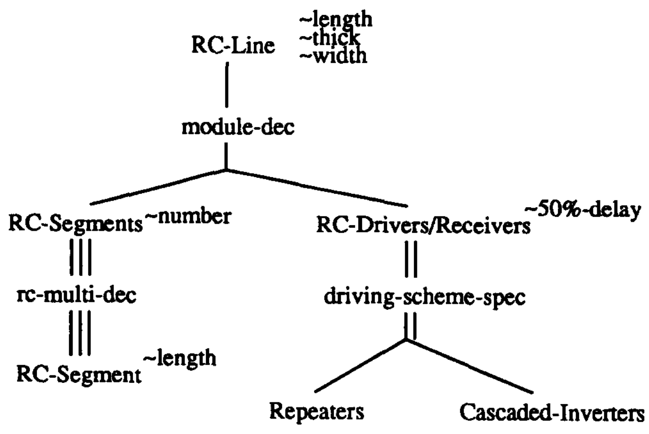


FIGURE 1. A system entity structure representation of RC line.

different application domains. Formally, an *Entity Information Frame (EIF)* is defined as follows:

$$EIF = \langle MD, AT, DS, SR, PR, LK \rangle,$$

where

MD: is the name of the associated model
AT: are attributes of the associated object
DS: is the design specification
SR: are the simulation requirements
PR: are production rules for pruning and synthesis
LK: are links to other FRASES nodes.

In model-based system design (Rozenblit, 1985), the behavior of a system component is described by a simulation model stored in the model base. To enable the synthesis of simulation models, MD (model) slot contains the pointer that can be used to extract the associated simulation model from the model base.

The AT-slot contains attributes that characterize the static and dynamic properties of the associated object (Rozenblit & Hu, 1988). In general, attributes of an object can be categorized into three types: static variables, design parameters, and performance indices. Static variables describe information about the object, which remains constant and should not be changed during the design process. Each static variable is usually initialized with a default value, a database query, or a user-provided function. Design parameters are variables related to behavioral characteristics of the associated object. Performance of the target system is determined by the combination of design parameters. Performance indices are variables used to evaluate the system's performance.

Formally, the AT-slot has the following frame structure:

```
⟨AT⟩ = (⟨static-variable⟩ ⟨design-parameter⟩ ⟨performance-index⟩)
⟨static-variable⟩ = (⟨name⟩ (value ⟨atom⟩))
⟨performance-index⟩ = (⟨name⟩ (value ⟨atom⟩) (tuning ⟨list⟩))
⟨design-parameter⟩ = (⟨name⟩ (value ⟨atom⟩) (if-needed ⟨list⟩) (boundary ⟨list⟩))
⟨name⟩ = string
⟨Atom⟩ = symbol|string|number
⟨list⟩ = function|logic-expression
```

The Design Specification (DS) form is used to accept design specifications such as design objectives, system requirements, and criteria preferences that must be satisfied by the target system.

The DS-slot has the following structure:

```
⟨DS⟩ = ( (constraints ⟨list⟩)
        (objectives (max {⟨name⟩}) (min {⟨name⟩}))
        (preference ⟨list⟩))
```

The Simulation Requirements (SR) form defines experimental circumstances under which a model can be observed (e.g., input segments and control schemes). The SR-slot is given by:

```
⟨SR⟩ = ((arrival ⟨list⟩) (control ⟨list⟩))
```

The PR-slot contains heuristic rules for pruning (i.e., selecting) design alternatives and configuring design model structures at different levels of abstraction. Constraints derived from the system architecture, technology, and resources are translated into production rules to assist in the design reasoning process. In FRASES, selection rules for pruning design alternatives are associated with specialization nodes; synthesis and coupling rules for configuring design models are associated with decomposition nodes. Formally, a PR-slot is given by:

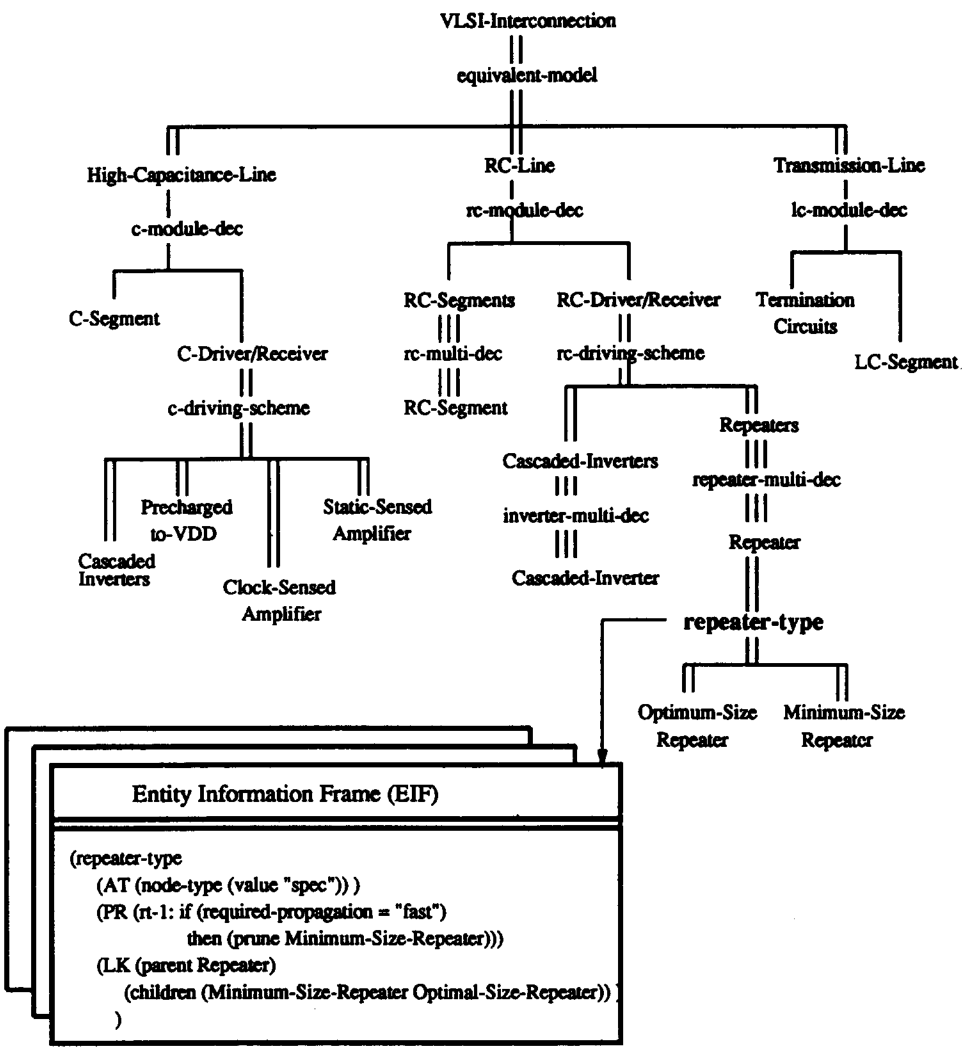
```
⟨PR⟩ = ([(⟨name⟩ if ⟨list⟩ then ⟨list⟩)])
```

The LK (link) slot defines the data flow relationship among FRASES nodes as follows:

```
⟨LK⟩ = ((children ⟨name⟩) (parents ⟨name⟩))
```

By exploiting the reasoning flexibility provided by production rules, the efficiency in representing declarative knowledge offered by frames, and the visibility and hierarchy supported by system entity structures, FRASES is a powerful and efficient scheme that supports modern system design (Hu, 1989). A typical example of FRASES for representing a VLSI interconnection is shown in Fig. 2.

In KAR, FRASES is used to acquire structured knowledge about a design domain in a systematic manner. Conceptually, the basis for such acquisition can be characterized as follows: Given a design problem in a specific domain, the designer is to develop an instance of FRASES for this domain, for example, VLSI Packaging. At this stage, FRASES is a generic "skeleton" representation which needs to be instantiated for the domain at hand. The domain knowledge must be elicited from experts in a manner that will classify it as objects, their decompositions, taxonomies, and attributes, and organize it as dictated by the FRASES axioms. Query rules must be defined that will guide



```
(repeater-type
   (AT (node-type (value "spec")) )
   (PR (rt-1: if (required-propagation = "fast")
              then (prune Minimum-Size-Repeater)))
   (LK (parent Repeater)
       (children (Minimum-Size-Repeater Optimal-Size-Repeater))
   )
```

FIGURE 2. FRASES representation of VLSI Interconnection.

the acquisition process so that an instance of a correct FRASES structure can be created. Thus, unstructured and fragmented expert knowledge can be organized into a format that expresses the hierarchy of decompositions and taxonomies of components of the system being designed. Such structured knowledge is used effectively to generate configurations of plausible designs (Rozenblit, 1985). The knowledge acquisition method is described in the following section.

4. KNOWLEDGE ACQUISITION PROCESS

The KAR acquisition process is divided into four phases: *entity, specialization, decomposition,* and *model/function.* At the *entity* phase, it acquires knowledge about "How to decompose the entity?", "How to classify the entity?", "What attributes are considered in the design of the entity?", and "What design methodology (e.g., top-down or bottom-up) is more appropriate for this design level?". At the *specialization* phase, KAR acquires knowledge about design variants and heuristic rules for selecting these design alternatives. At the *decomposition* phase, knowledge about subcomponents, synthesis, coupling, and design priority is acquired. The acquisition process is repeated for the entity, decomposition, and specialization phases until the desired level of abstraction is reached. Finally, at the *model/function* phase, behavioral models associated with all leaf nodes and quantitative functions specified during the acquisition process are defined.

To automate the generation of question patterns, queries are defined and associated with each acquisition phase. The structure of FRASES implies the following form of the rules:

To assist in understanding questions, explanation patterns such as WHY, WHAT, and HOW are associated with the queries. For example, the S-SR rule for a specialization node may have three explanation rules, termed S-SR.WHY, S-SR.WHAT, and S-SR.HOW, to explain "Why selection rules are required," "What a selection rule means," and "How to specify a selection rule." For illustration, a typical query rule base for the

```
(E-PD "What is the problem domain? ")
(E-PD.WHY "% This query is used to acquire the root of a FRASES tree. %")
(E-MD "Does the number of ~S vary with design requirements?)
(E-MD.HOW "Format: yes/no")
(E-AT "What attributes are considered when you design the ~S?")
(E-DP "Can you decompose the ~S based on certain aspect?")
(E_DP.HOW "% Format: [<atom>] %")
(E-SP "Can you classified the ~S based on certain specialization?")
(E-SP.HOW "% Format: [<atom>] %")
(E-PP "What's the processing priority for ~S?")
(E-PP.WHY  "% At each design level, it is important to determine if the"
          "top-down is better than bottom-up design approach %")
(E-PP.HOW "% Format: [<atom>] %")
(D-SC "What are these subcomponents when decompose ~S based on ~S?")
(D-SC.HOW "% Format: [<atom>] %")
(D-SR "Can you define synthesis rules for ~S?")
(D-SR.HOW "% Format: (<name> if <s-expression> then <s-expression>) %")
(D-CP "Please define coupling for ~S.")
(D-CP.HOW "% Format: (<name> if <s-expression> then <s-expression>) %")
(D-CP.WHAT "% Coupling inoformation defines how a coupled model is"
          "constructed from its component models. %")
(D-DP "Please rank the design priority for ~S.")
(D-DP.HOW "% Format: [<atom>] %")
(S-VA "What are these alternatives when classify ~S based on ~S?")
(S-VA.HOW "% Format: [<atom>] %")
(S-SR "Can you specify rules for selecting ~S of ~S?")
(S-SR.WHY "% To assist in selecting design alternatives %")
(S-SR.HOW "% Format: (<name> if <s-expression> then <s-expression>) %")
(M-MD "What is the model definition for ~S?")
(M-MD.HOW "% Format: [(s-expression)] %")
(M-FN "What is the function definition for ~S?")
(M-FN.HOW "% Format: [(s-expression)] %")
```

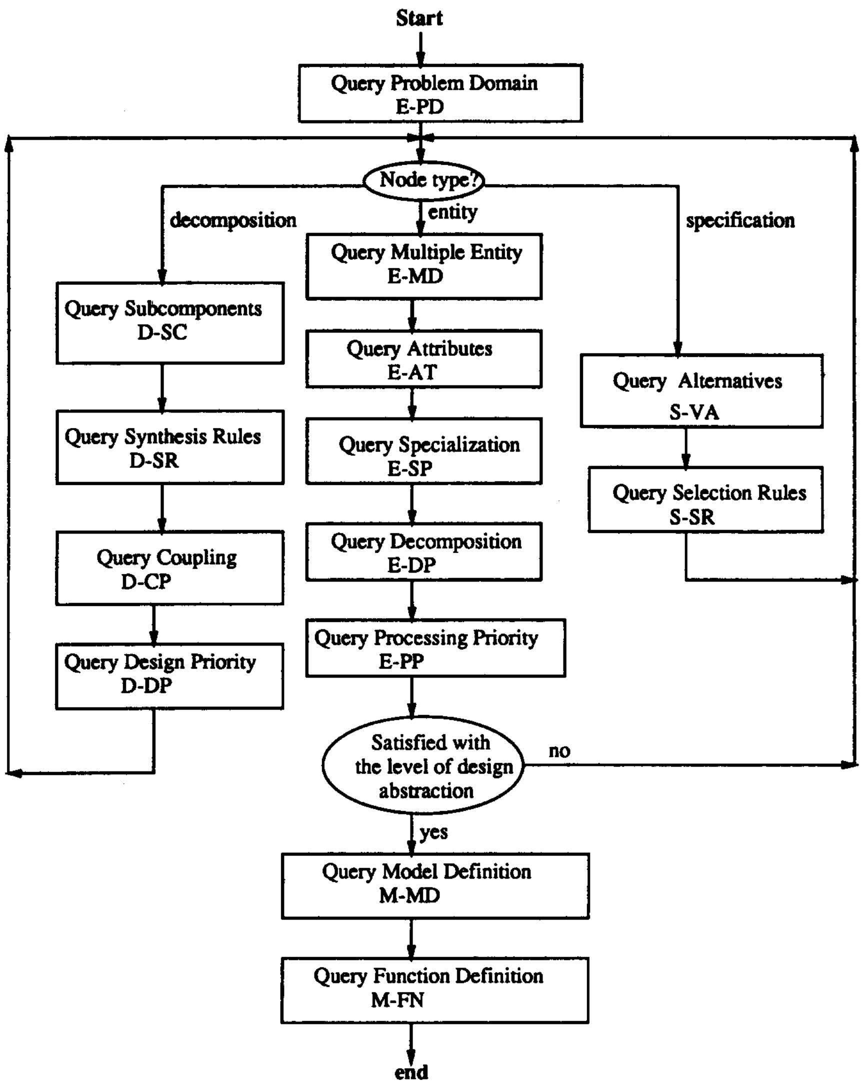
FIGURE 3. A typical KAR query rule base.

KAR using FRASES is given in Fig. 3. Figure 4 illustrates the query process of KAR.

5. KNOWLEDGE VERIFICATION AND VALIDATION

To ensure data/knowledge consistency, the acquired knowledge must be validated (with respect to the FRASES's axioms) after each query rule is applied. In KAR, this is accomplished by converting the axioms of FRASES into verification rules. At the end of each acquisition phase, related verification rules are applied to a) ensure the consistency of design knowledge, b) detect fragments or dead-ends of design knowledge, c) enhance the capability of acquisition, and d) validate configurations of design components at different abstraction levels. Whenever violations of verification rules is detected, error messages are signaled to users so that an appropriate correction or modification can be made. Verification rules of KAR are listed below:

- *Verification of uniformity* (*UF*): Any two nodes with the same labels must possess identical Entity Information Frames (EIF) and isomorphic substructures. The main function of the UF rule is to ensure the consistency of data and knowledge for a node that appears in different places of a FRASES structure. The *UF* verification is associated with the following query rules: E-PD, E-MD, E-DP, E-SP, S-VA, and D-SC (please see Table 1).
- *Verification of attached variables* (*AV*): No two variables have the same name. This rule prevents duplication of design attributes. The *AV* verification is attached to the query rule E-AT.
- *Verification of knowledge inheritance* (*KI*): All attributes and substructures can be inherited through a specialization node of FRASES. The application of the KI rule enables the expert to validate the inherited information and/or to add constraints to the inheritance scheme. The *KI* verification is attached to the query rule S-VA.
- *Verification of structure hierarchy* (*SH*): No label appears more than once down any path of the FRASES tree. The SH rule enforces the strict hierarchy axiom of FRASES. The *SH* verification rule is associated with the following query rules: E-MD, E-DP, E-SP, S-VA, and D-SC.
- *Verification of knowledge hierarchy* (*KH*): To simplify management and refinement of complex design knowledge, each node contains only information related to itself, its ancestors, and its descendants, but not siblings. The *KH* verification rule is associated



Start
Query Problem Domain E-PD
Node type?
decomposition
entity
specification
Query Subcomponents D-SC
Query Synthesis Rules D-SR
Query Coupling D-CP
Query Design Priority D-DP
Query Multiple Entity E-MD
Query Attributes E-AT
Query Specialization E-SP
Query Decomposition E-DP
Query Processing Priority E-PP
Query Alternatives S-VA
Query Selection Rules S-SR
Satisfied with the level of design abstraction
no
yes
Query Model Definition M-MD
Query Function Definition M-FN
end

FIGURE 4. Query process of KAR.

with the following query rules: E-AT, S-SR, D-SR, and D-CP.

- *Verification of rule conflicts* (*RC*): Rule clusters that results in conflicts or dead-ends (no selection) of synthesis must be pointed out for modification. The purpose of *RC* verification is to help develop a robust rule base. The *RC* verification rule is associated with the following query rules: S-SR, D-SR, and D-CP.
- *Verification of rule fragments* (*RF*): Rules that are logically related should be combined into clusters to reduce the complexity of the knowledge base. The *RF* verification is associated with query rules S-SR and D-SR.
- *Verification of valid siblings* (*VS*): No two entities under the same node have the same labels. The VS rule removes data/knowledge fragments and avoids possible inconsistencies in design knowledge. The *VS* verification is associated with the following query rules: E-DP, E-SP, E-PP, S-VA, D-DP, and D-SC.

6. EXAMPLE OF KAR

To illustrate KAR, the knowledge acquisition process for the electrical modeling design of Very Large Scale Integrated (VLSI) interconnections is traced in Fig. 5. In the figure, the left column describes the query/answer activities and the right column illustrates the refinement of the FRASES transition.

To acquire knowledge for system design applications, KAR begins with the entity phase. The E-PD query rule is first interpreted to acquire the problem domain (i.e., *VLSI-Interconnection*). Following the E-PD rule, KAR applies the E-MD rule to check if the number of entities may vary with system requirements. The E-MD query is then followed by the E-AT rule for querying attributes that characterize the focus entity. After the E-AT rule, E-SP and E-DP are used to acquire knowledge about how experts decompose and classify the focus entity. Since only one node is specified

TABLE 1
Query Rules of KAR

Acquisition Phases	Query Rules
Entity	Problem domain query (E-PD) Multiple decomposition query (E-MD) Decomposition query (E-DP) Design attributes query (E-AT) Processing priority query (E-PP) Specialization query (E-SP)
Specialization	Design variants query (S-VA) Selection rules query (S-SR)
Aspect	Subcomponents query (D-SC) Synthesis rules query (D-SR) Coupling information query (D-CP) Design priority query (D-DP)
Model/function	Association models query (M-MD) Association functions query (M-FN)

(i.e., *equivalent-model*) in our example, there is no need to invoke the E-PP rule. At the end of the entity acquisition phase for *VLSI-Interconnection,* the acquisition process proceeds to the specialization phase for *equivalent-model.* The query rule S-VA is first interpreted to elicit design variants under this classification. Following the S-VA query, heuristic rules for selecting design variants are acquired by applying the S-SR query. At the end of the S-SR query, the acquisition backtracks to the entity phase for *RC-Line, High-Capacitance-Line,* and *Transmission-Line* sequentially. For the *RC-Line,* query rules are applied as for *VLSI-Interconnection.* This time, a decomposition *rc-module-dec* is specified for the *RC-Line.*

After knowledge about *RC-Line* is acquired, the acquisition process moves to the decomposition phase for *rc-module-dec.* At *rc-module-dec,* the D-SC query is invoked to acquire subcomponents (i.e., *RC-Segments and RC-Drivers-Receivers*) for a *RC-Line.* Following the D-SC query, the D-SR and D-CP rules are used to acquire synthesis constraints and coupling information of subcomponents. Since the number of *RC-Segments* is determined by the realization of *RC-Drivers/Receivers,* a synthesis constraint about the number of *RC-Segments* is specified when the D-SR query rule is applied. As shown in the EIF of *rc-module-dec,* the coupling is defined to fit different implementations of *RC-Drivers/Receivers.* Finally, the D-DP rule is interpreted to inquire about the design priority. It is best to design the driver/receiver circuits first because this has a great impact upon the circuit properties. This priority information enables future design activities to start from the most essential component of the system.

At the end of the D-DP query, the acquisition process is continued for *RC-Drivers/Receivers* from where a specialization (i.e., *rc-driving-scheme*) is defined. With the S-VA query at *rc-driving-scheme,* two design variants are specified (i.e., *Repeaters* and *Cascaded-Inverters*). When the acquisition process focuses on *Repeaters,* the E-MD query is responded to with "yes" to indicate that the number of repeaters varies with design specifications. This situation is represented by a multiple decomposition (|||) in FRASES. Whenever a multiple decomposition is specified, the *number* attribute is added to the focus entity. A quantitative function must be specified to calculate the number of entities (i.e., *Repeaters* in our example). In order for repeaters to reduce the overall delay, the number of repeaters must be at least two. This is accomplished by specifying the boundary information associated with the *number* attribute. Following the *Repeaters,* coupling among individual repeaters is defined by applying D-CP query at the acquisition of *repeater-multi-dec.*

The acquisition process continues until the desired level of abstraction is reached (e.g., *Repeater* and *RC-Segment*) from where the behavral model must be defined. The same acquisition process is then performed on *High-Capacitance-Line* and *Transmission-Line.* At the end of the acquisition process, KAR moves to the model/function phase by reporting all leaf nodes and quantitative functions specified during the development of a FRASES knowledge base. For illustration, parts of the resulting Entity Information Frames (EIF) are shown in Fig. 6.

7. ADVANTAGES OF KAR

Several advantages are expected from using KAR with FRASES for knowledge acquisition:

- *Efficiency:* Question patterns required for acquiring essential knowledge such as decomposition, taxonomy, pruning, and synthesis are generated automatically by interpreting the associated query rules. On each query cycle, the acquired knowledge is validated automatically by applying verification rules. The validated knowledge is then translated into the Entity Information Frame (EIF) format.
- *Flexibility:* Conventional knowledge acquisition methods are efficient only in a specific problem domain. With KAR, the query and verification rules can be refined easily to fit a large class of systems exhibiting a hierarchical and modular structure. Beyond that, design knowledge can be easily formulated to facilitate different design approaches (e.g., top-down, bottom-up, or hybrid) by specifying appropriate design and processing priorities (Hu, 1989).
- *Manageability:* The hierarchical, entity-based structure confines all related knowledge to specific tree paths (i.e., from the focus node to the root and to the leaves). This is particularly useful for knowledge refinement. The drawback of a flat knowledge base that requires an exhaustive search for all related

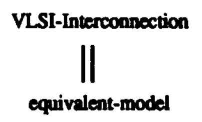
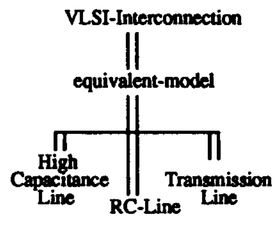
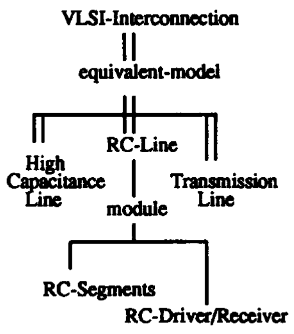
KAR/EXPERT INTERACTION	QUERY	FRASES
KAR> What is the problem domain? => VLSI-Interconnection	E-PD	VLSI-Interconnection
... KAR> What design parameters will be considered for VLSI-Interconnection? => (line-length (if-needed ask) (unit (value cm))) ... ... KAR> Can you classify VLSI-Interconnection based on certain specialization? => equivalent-model KAR> why => % This query acquires information indicating how an entity is classified. % ...	E-AT E-SP E-SP.WHY	
KAR> Can you specify possible equivalent-models of VLSI-Interconnection? => High-Capacitance-line RC-Line Transmission-Line KAR> Specify rules for selecting equivalent-model of VLSI-Interconnection. => (em-r1: if (signal-rise-time < 3*time-of-flight-delay) then (prune RC-Line High-Capacitance-Line)) (em-r2: if (signal-rise-time > 3*time-of-flight-delay) then (prune Transmission-Line)) (em-r3: if (line-resistance > transistor-on-resistance) then (prune High-Capacitance-Line)) ...	S-VA S-SR	 VLSI-Interconnection equivalent-model High Capacitance Line RC-Line Transmission Line
KAR> Can you decompose RC-Line based on certain aspect? => module ... KAR> What are these subcomponents when you decompose a RC-Line based on module? => RC-Segments RC-Drivers/Receivers KAR> List the order of design priority for RC-Segments and RC-Driver/Receiver => RC-Driver/Receiver RC-Segments ... KAR> Can you specify synthesis rules for RC-Line? => (rc-r1: if (RC-Driver/Receiver = Repeaters) then (RC-Segments.number = Repeaters.number)) (rc-r2: if (RC-Driver/Receiver = Cascaded-Inverters) then (RC-Segments.number = 1)) ...	E-DP D-SC D-DP D-SR	 VLSI-Interconnection equivalent-model High Capacitance Line RC-Line Transmission Line module RC-Segments RC-Driver/Receiver
KAR> Does the number of RC-Segments vary with design requirements? => yes ... KAR> Can you specify coupling information for RC-Segments? => (for (i=0; i<number; i++) (RC-Segments.in[i] -> RC-Segment[i].in) (RC-Segment[i].out -> RC-Segments[i].out)) ...	E-MD D-CP	** See Figure 2

FIGURE 5. KAR for VLSI Interconnection.

knowledge chunks to be updated is eliminated in our approach.

- *Cost-Effectiveness:* Conventional acquisition strategies require a labor intensive human intervention in interviews, verification, translation, and organization of knowledge, which increases the cost and reduces the reliability of the knowledge base. With KAR, the human intervention is minimized by an automatic acquisition process that reduces errors (e.g., misinterpretation of the acquired knowledge) and increases the reliability of a knowledge base.

8. SUMMARY AND FUTURE RESEARCH

Conventional acquisition methods which rely on knowledge engineers for the development of a knowledge base become inefficient when the complexity of systems grows. To improve the drawbacks of such methods, a new methodology called knowledge acquisition based on explicit representation (KAR) has been presented. To realize KAR, we have developed a knowledge representation scheme, termed FRASES. Through its automatic acquisition process, KAR reduces the cost and increases reliability of the knowledge base development for system design applications.

The methodology presented here is the conceptual basis for a prototype system for design knowledge acquisition. The system, when fully implemented, will require validation and empirical studies of efficiency with respect to different design domains.

In future research we plan to employ cognitive models that would enable KAR to learn from design

```
(VLSI-Interconnection
   (AT (node-type (value entity))
       (line-width (if-needed ask) (unit (value micron)))
       (line-length (if-needed ask) (unit (value cm)))
       (line-thick (if-needed ask) (unit (value micron)))
       (design-rule (if-needed ask) (unit (value micron)))
       (dielectric-thick (if-needed ask) (value (unit micron)))
       (dielectric (if-needed ask)
                 (boundary polymide silicon-dioxide epoxy-glass alumina))
       (IC-technology (if-needed ask) (boundary cmos nmos bipolar GaAs) )
       (package-technology  (if-needed ask)
         (boundary Wafer-Scale-Integration Ceramic-Hybrid
                  Thin-Film-Hybrid Printed-Wiring-Board))
       (conductor-material (if-needed ask) (boundary Au Al Cu Ag))
       (line-capacitance (if-needed (eval (/ (*
           (query dielectric dielectric-constant)
           line-width line-length) dielectric-thick))) (unit farad))
       (signal-rise-time
         (if-needed (data_query IC-technology rise-time)) (unit nsec))
       (time-of-flight-delay (if-needed (eval (/ (* (sqrt
           (query package-technology
           relative-dielectric-constant)) line-length) 30))) (unit nsec))
       (transistor-on-resistance
         (data_query IC-technology on-resistance) (unit ohm))
   (LK (parent nil) (children equivalent-model)) )

(equivalent-model
    (AT (node-type (value specialization)) )
    (PR (em-r1: if   (< signal-rise-time (* 3 time-of-flight-delay))
                then (prune RC-Line) and
                     (prune High-Capacitance-Line) )
        (em-r2: if   (> signal-rise-time (* 3 time-of-flight-delay))
                then (prune Transmission-Line)
        (em-r3: if   (> line-resistance Transistor-On-Resistance)
                then (prune High-Capacitance-Line)) )
    (LK (parent VLSI-Interconnection)
         (children High-Capacitance-Line RC-Line Transmission-Line)))

(RC-Line
    (AT (node-type (value entity))
        (line-resistance (if-needed (eval
             (/ (* (data_query conductor-material resistivity)
                   line-length) (* line-width line-thick)))) (unit ohm)) )
    (LK (parent equivalent-model) (children rc-module-dec)))

(rc-module-dec
     (AT (node-type (value decomposition)))
     (PR (rc-r1: if (equal? RC-Driver/Receiver Repeaters)
                 then (set! RC-Segments.number Repeaters.number))
         (rc-r2: if (equal? RC-Driver/Receiver Cascaded-Inverters)
                 then (set! RC-Segments.number 1))
         (rc-r3: if (equal? operation-phase coupling) and
                    (set! RC-Driver/Receiver Repeaters)
               then (make-coupling (RC-Line.in RC-Segments[1].in)) and
                    (eval
                      (loop ((= i 1) (< i (+ Repeaters.number 1)))
                         (make-coupling
                           (RC-Segments[i].out Repeaters[i].in)
                           (Repeaters[i].out RC-Segments[i+1].in)))
                      (make-coupling RC-Segments[RC-Segments.number].out
                                     RC-Line.out)))
         (rc-r2: if (equal? operation-phase coupling) and
                    (equal? RC-Drivers/Receivers Cascaded-Inverters)
                then (make-coupling
                      (RC-Line.in  RC-Segments[1].in)
```

FIGURE 6. Entity Information Frames for VLSI interconnection.

```
                    (RC-Segments[1].out  Cascaded-Inverters.in)
                    (Cascaded-Inverters.out RC-Line.out) )))
   (LK (parent RC-Line) (children RC-Segments RC-Driver/Receiver)) )

(rc-multi-dec
   (AT (node-type (value multiple-decomposition)))
   (PR rcm-r1:
        if (equal? operation-phase coupling)
        then (eval (loop ((= i 1) (<= i RC-Segments.number))
                (make-coupling (RC-Segments[i].in RC-Segment[i].in)
                    (RC-Segments[i].out RC-Segment[i].out)  ))))
   (LK (parent RC-Segments) (children RC-Segment)))

(rc-driving-scheme
   (AT (node-type (value "spec")))
   (PR (rcd-r1:
         if (> line-resistance
            (* 7 (data_query minimum-size-buffer on-resistance)))
         then (prune Cascaded-Inverters)))
   (LK (parent RC-Drivers/Receivers)
         (children Repeaters Cascaded-Inverters)))

(Cascaded-Inverters
   (AT (node-type (value entity))
       (number (if-needed (eval
          (ln (/ (+ RC-Line.line-capacitance RC-Line.load-capacitance)
               (data_query minimum-size-buffer input-capacitance))))))
   (LK (parent c-driving-scheme) (children nil)))

(Repeaters
   (AT (node-type (value entity))
       (number
        (if-needed (eval (sqrt
          (/ (* 0.4 RC-Line.line-resistance RC-Line.line-capacitance)
             (* 0.7 (data_query minimum-size-buffer input-capacitance)
               (data_query minimum-size-buffer output-resistance))))))
        (boundary (> 2)) ))
   (LK (parent rc-driving-scheme) (children repeater-multi-dec)))

(repeater-multi-dec
   (AT (node-type (value multiple-decomposition)))
   (PR (rpm-r1:
         if (equal? operation-phase coupling)
         then (eval (loop ((= i 1) (<= Repeaters.number))
                    (make-coupling (Repeaters[i].in Repeater[i].in)
                       (Repeaters[i].out Repeater[i].out)  )))))
   (LK (parent Repeaters) (children Repeater)))

(inv-multi-dec
   (AT (node-type (value multiple-decomposition)))
   (PR (ivm-r1:
         if (equal? operation-phase coupling)
         then (eval (loop ((= i 1) (< i number))
               (make-coupling (Inverter[i].out Inverter[i+1].in))))))
   (LK (parent Cascaded-Inverters) (children inverters)))

(Repeater
   (AT (node-type (value entity))
       (50%-delay (if-needed (eval (* 2.5 (sqrt
          (* (data_query minimum-size-buffer output-resistance)
             (data_query minimum-size-buffer input-capacitance)
             RC-Line.line-resistance
             RC-Line.line-capacitance))))) )
   (LK (parent repeater-multi-dec) (children repeater-type)))
```

FIGURE 6. (continued).

```
(Inverter
   (MD (value inverter))
   (AT (node-type (value entity))
       (50%-delay (if-needed (eval
        (+ (* 0.4 VLSI-Interconnection.line-resistance
                VLSI-Interconnection.line-capacitance)
        (* 0.7 VLSI-Interconnection.line-resistance
                VLSI-Interconnection.load-capacitance)
        (* 1.9 (data_query minimum-size-buffer output-resistance)
                (data_query minimum-size-buffer input-capacitance)
                (ln (/ (+ VLSI-Interconnection.line-capacitance
                           VLSI-Interconnection.load-capacitance)
                (data_query minimum-size-buffer input-capacitance))))))
      (unit micro-sec)))
(LK (parent inv-multi-dec) (children nil)))
```

FIGURE 6. (continued).

experience. We seek to derive an automatic knowledge refinement scheme by incorporating methodologies of machine learning (Fisher, 1986; Michalski, Carbonell, & Mitchell, 1986; Mostow, 1989).

The user-interface should incorporate methods offered by cognitive psychology (Barnard & Harrison, 1989; Harrison & Thimbleby, 1990; Long & Whitefield, 1989). At the current stage of implementation (i.e., LISP on VAX-II/780), the human-computer interaction in KAR is carried out in the question/answer format. No graphical or audible aids are used to support the acquisition process. Confusion may result if query and verification rules are not clear to the user. To improve this drawback, a high quality user interface utilizing cognitive ergonomics is needed.

REFERENCES

Barnard P. & Harrison M. (1989). Integrating cognitive and system models in human computer interaction,"In A. Sutcliffe & L. Macaulay, (Eds), People and Computers V, Cambridge University Press.

Boose, J.H. (1988). Uses of repertory grid-centered knowledge acquisition tools for knowledge-based systems. *Ant. J. Man-Machine Studies,* **29**(3), 287-310.

Chang, C.L. & Lee, R.C. (1973). *Symbolic logic and mechanical theorem proving.* New York: Academic Press.

Fisher, E.L. (1986). An AI-based methodology for factory design. *AI Magazine,* 7(4), 72-85.

Gaines, B.R. (1987). An overview of knowledge acquisition and transfer. *Ant. J. Man-Machine Studies,* **26**, 453-472.

Gaines, B.R. (1988). Knowledge acquisition-systems for rapid prototyping of expert systems. *Inform.,* **26**(4), 256-285.

Harrison, M. & Thimbleby, H. (1990). Formal methods in human-computer interaction. New York: Cambridge University Press.

Hart, A. (1985). Knowledge elicitation: Issues and methods. *Computer-Aided Design,* **17**(9), 455-462.

Hart, A. (1985). The role of induction in knowledge elicitation. *Expert Systems,* **2**, 24-28.

Hu, J., Huang, Y.M., & Rozenblit, J.W. (1989). FRASES—A knowledge representation scheme for engineering design. *Advances in AI and Simulation,* **20**(4), 141-146.

Hu, J. (1989). *Towards an integrated knowledge-based design support environment for design automation and performance evaluation.* Doctoral dissertation, University of Arizona, Tucson.

Kelly, G.A. (1955). *The psychology of personal constructs.* New York: Norton.

Kessel, K.L. (1986). Methodological tools for knowledge acquisition and transfer. *Proceedings of the* 1986 *IEEE International Conference on System, Man, and Cybernetics,* Atlanta, GA.

Long, J. & Whitefield, A. (1989). Cognitive ergonomics and human-computer interactions. New York: Cambridge University Press.

Mesarovic, M.D. & Takahara, Y. (1975). *General system theory: Mathematical foundation.* New York: Academic Press.

Michalski, R.S., Carbonell, J.G., & Mitchell, T.M. (1986). *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann.

Minsky, M. (1975). A framework for representing knowledge. In P.H. Winston (Ed.), *The psychology of computer vision* (pp. 211-277). New York: McGraw-Hill.

Mostow, J. (1989). Design by derivational analogy: Issues in the automated replay of design plans. *Artificial Intelligence,* **40**, 119-184.

Newell, A. & Simon, H.A. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice-Hall.

Nilsson, N.J. (1988). Principles of artificial intelligence. Palo Alto, CA: Tioga Press.

Olson, J.R. & Rueter, H.H. (1987). Extracting expertise from experts: Methods for knowledge acquisition. *Expert System,* **4**(3), 152-168.

Quillian, M.R. (1968). Semantic memory. In M. Minsky (ed.), *Semantic information Processing* (pp. 27-70). Cambridge, MA MIT Press.

Ritchie, I.C. (1984). Knowledge acquisition by computer induction. In *Proceedings of UNICOM Seminar.*

Rozenblit, J.W., Hu, J., & Huang, Y.M. (1989). An integrated, entity-based knowledge representation scheme for system design. *The* 1989 *NSF Engineering Design Research Conference.*

Rozenblit, J.W. & Zeigler, B.P. (1988). Design and modeling concepts. *International Encyclopedia of Robotics Application and Automation.* New York: John Wiley and Sons.

Rozenblit, J.W., Hu, J., Kim, T.G., & Zeigler, B.P. (1990). Knowledge-based design and simulation environment (KBDSE): Foundational concepts and implementation. *Journal of Operational Research Society,* **41**(2).

Rozenblit, J.W. (1985). *A conceptual basis for integrated, model-based system design.* Doctoral Thesis, Department of Computer Science, Wayne State University, Detroit, MI.

Rozenblit, J.W., & Zeigler, B.P. (1986). Entity-based structures for model and experimental frame construction. In: *Modelling and* *simulation in artificial intelligence era,* M.S. Elzas, T.I. Ören, & B.P. Zeigler, (Eds.), Amsterdam: North-Holland. (pp. 78–100).

Rozenblit, J.W. & Zeigler, B.P. (1988). Design and modelling concepts. In *International Encyclopedia of Robotics, Applications and Automation* (pp. 308–322). (R. Dorf (ed), New York: John Wiley and Sons.

Schank, R.C. & Abelson, R.P. (1977). *Scripts, plans, goals, and understanding.* Hillsdale, NJ: Erlbaum.

Shastri, L. (1988). Semantic networks: an evidential formalization and its connectionist realization. *Research notes in artificial intelligence.* San Mateo, CA: Morgan Kaufman Publishers.

Waterman, D.A. & Newell, A. (1971). Protocol analysis as a task for artificial intelligence. *Artificial Intelligence,* **2,** 285.

Weste, N. & Eshraghian, K. (1985). *Principles of CMOS VLSI design—A system perspective.* Reading, MA: Addison-Wesley Publishing Company.

Wymore, A.W. (1967). *A mathematical theory of systems engineering: The elements.* New York: John Wiley.

Zeigler, B.P. (1984). *Multifaceted modelling and discrete event simulation.* London: Academic Press.

Zeigler, B.P. (1986). Knowledge representation from Newton to Minsky and beyond. *Applied Artificial Intelligence,* **I,** 87–107.

Zeigler, B.P. (1987). Hierarchical, modular discrete-event models in an object-oriented environment. *Simulation,* **50**(5), 219–230.

Zeigler, B.P. (1990). *Object-oriented simulation with hierarchical modular models: Intelligent agents and endomorphic systems.* New York: Academic Press.