NETSYN - A Connectionist Approach to Synthesis Knowledge Acquisition and Use

N. Ivezic, J. Garrett

EDRC 12-54-92

NETSYN — a Connectionist Approach to Synthesis Knowledge Acquisition and Use

a Technical Report by

Nenad Ivezic and James Garrett, Jr.

July 1992

This work has been supported by the Engineering Design Research Center, an NSF Engineering Research Center.

Abstract

The goal of machine learning for synthesis is the acquisition of the relationships between form, function, and behavior properties that satisfy design requirements. The proposed approach creates a function to estimate the probability of each possible value of each design property being used in a given design context. NETSYN uses a connectionist learning approach to acquire and represent this probability estimationfunction and exhibits good performance for aposed artificial design problem. The objective of future woik is to apply NETSYN to realistic design domains, specifically the domain of computer system design as practiced by Ml.

Acknowledgments

The authors gratefully acknowledge the help provided by Benoit Julien and Yoram Reich in evaluating the ILLS system and the ECOBWEB system for the purposes of comparisons presented in this report.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The objectives of this chapter are to describe a general framework for the synthesis processes considered in this research and to identify a role for machine learning in design domains with weak or nonexistent synthesis knowledge. In addition, this chapter includes an overview of, and the motivation for, connectionist learning approaches for synthesis knowledge acquisition and use (SKAU). Finally, an outline of the contents of the report is given.

1.1 SYNTHESIS AND THE ROLE FOR MACHINE LEARNING

1.1.1 Design Framework for Synthesis

The main objective of the synthesis process is to generate a description of the form of an artifact such that this description satisfies a collection of design requirements. Form here is used in the same sense as in [Hemming 92] to mean geometry, topology, material, etc. The synthesis process is the initial stage of a multi-stage artifact design process consisting of the following four stages: synthesis, analysis, evaluation, and redesign. In the synthesis stage, several possible forms of an artifact are generated based on a given set of specifications. In the analysis stage, the behavior of these possible forms are determined for the specified functional context. In the evaluation stage, the predicted behaviors are compared with those desired and the deficiencies of the design are identified. In the redesign stage, changes to original possible forms are made to address the deficiencies found during evaluation, or new realizations are generated.

If we consider the artifact resulting from a design process to be described by *its form* (e.g., topology, shape, material),/wrtcri0rt (e.g., load resistance, vibration isolation), and *behavior* properties (e.g., stress and strain states), then we can describe each of the four stages in the multi-stage process as follows [Flemming92]:

- Synthesis is the process of mapping from the design specification space (requirements on function, behavior, form) to the form space (Fig. 1).
- Analysis is the process of mapping from the form and specification spaces to the behavior space (Fig. 2).
- Evaluation is the process of comparing the behavior properties to the behavior requirements and the form properties to the form requirements (Fig. 3).
- Redesign is the process of mapping from the specification space to the form space, with the knowledge of subspaces that lead to infeasible designs (Fig. 4).

1.1.2 A Role for Machine Learning

The specification for an artifact used as input by the synthesis process may involve any number of required functionalities, form constraints, and behavior constraints. For problems where all of the constraints and objectives can be clearly articulated and expressed in algebraic form, mathematical opti-

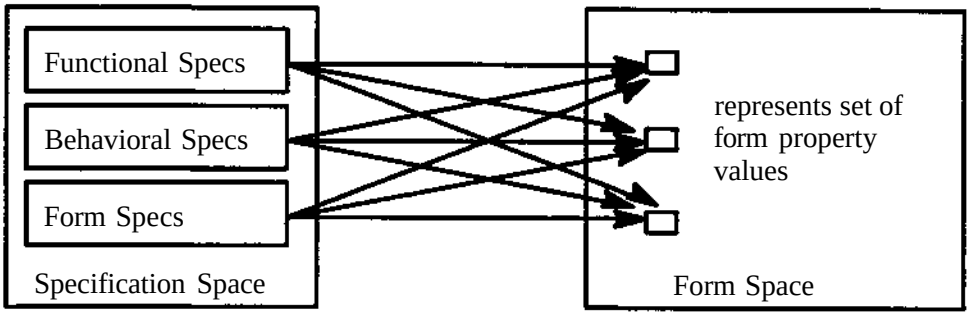


Figure 1 — Synthesis

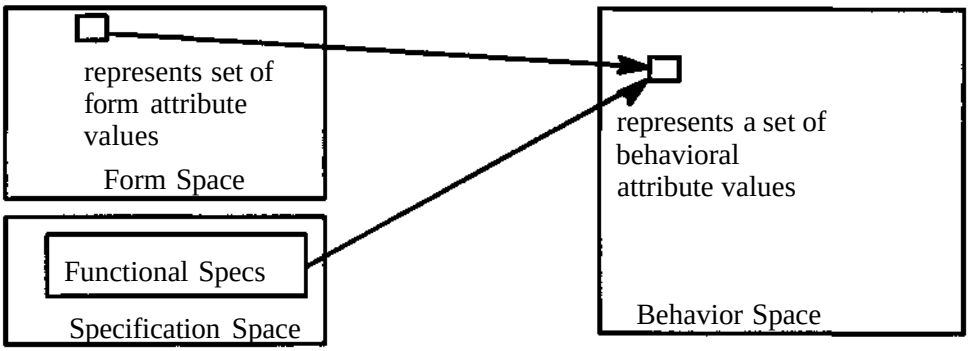


Figure 2 — Analysis

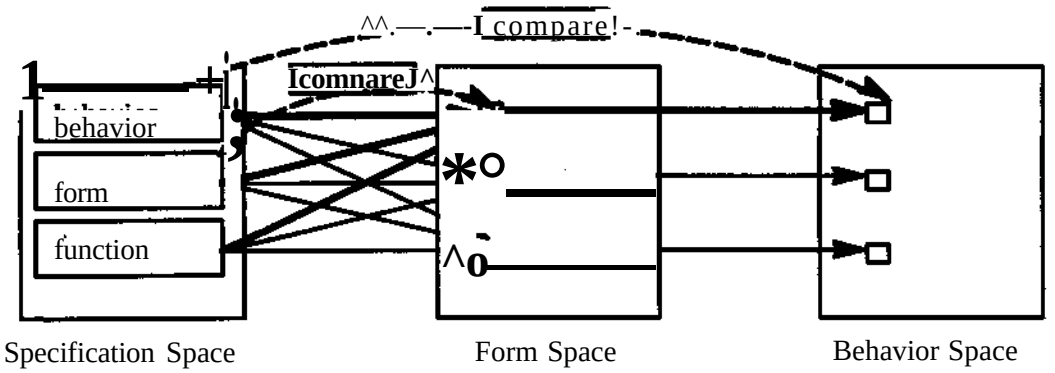


Figure 3 — Evaluation

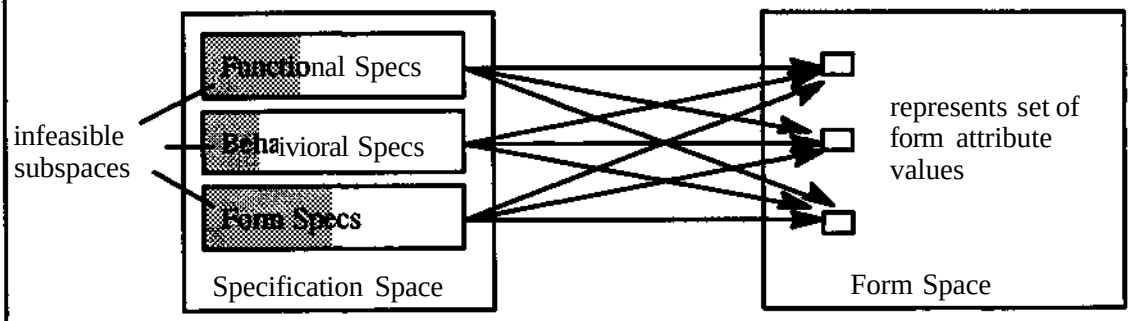


Figure 4 — Redesign

mization approaches can be employed to search over the solution space (i.e., the space of all possible solutions) and select the "optimal" solution to the set of constraints and defined objectives. However, for problems where there are few constraints and the relationships between form and performance are unknown, a mathematical optimization approach is not amenable. However, if weak theories or heuristic knowledge about synthesizing solutions to a problem exist, knowledge-based synthesis approaches can be used to assist (through the application of heuristic knowledge) in efficiently searching through the space of possible design solutions, eliminating those solutions that are not promising and focussing on those solutions that are. However, for those problems in which no significant domain theory or collection of heuristics exists, the only source of synthesis knowledge tends to be past design experience [Reich 91].

One role for machine learning, identified in [Reich 91], lies within those design domains possessing the following characteristics:

- The design space is identified (i.e., all properties and possible values are known a priori).
- Synthesis knowledge is weak or nonexistent
- There exists a collection of design experiences.

The role of machine learning in domains of this type is to acquire, from past design cases, the relationships between form, function, and behavior properties that satisfy specified design requirements. These relationships form a cone of useful synthesis knowledge. Having such synthesis knowledge facilitates *a more direct* mapping from the specification space to the desired locations within the form and behavior spaces. In other words, less search is needed to find a good solution to the specification (Fig. 5).

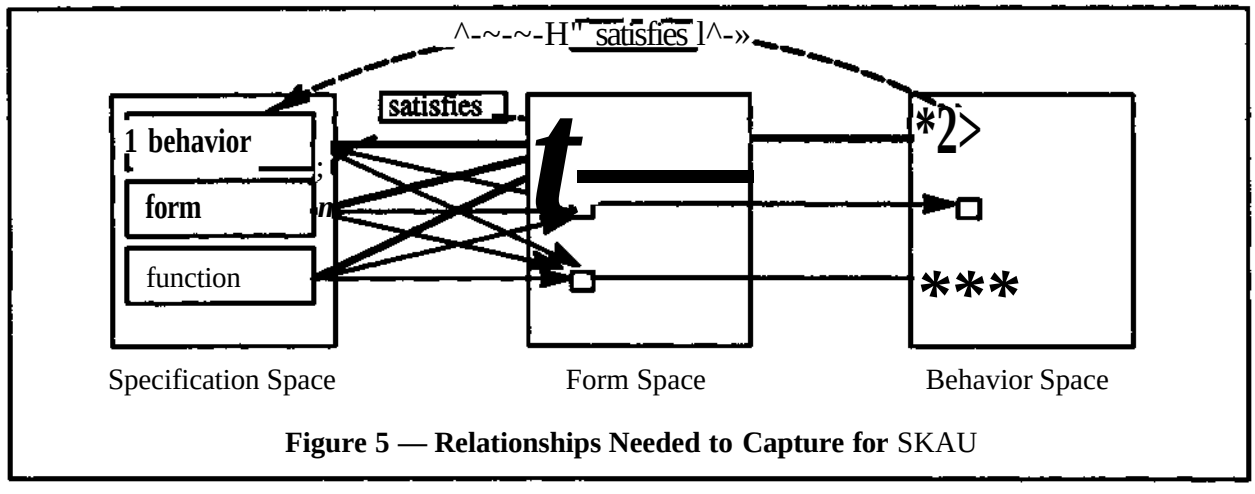


Figure 5 — Relationships Needed to Capture for SKAU

Hence, the goal of the research described in this report is to investigate and identify promising learning approaches which are capable of "playing" the above role of acquiring synthesis knowledge. Next, we outline the approach taken by this research in accomplishing this goal that will be presented in more detail later in this report.

12 OVERVIEW OF APPROACH

We propose a probabilistic approach to synthesis that is based on the following assumptions:

- Design can be sufficiently well represented as a collection of property-value pairs (i.e., a finite number of decisions). Consequently, the synthesis process is equivalent to a sequence of assignments of property values which ultimately lead to a complete design definition.

- Each property value assignment (decision) is made in some design context (i.e., partially defined specification and/or design description).
- It is possible to construct a sufficiently accurate estimation of the *probability of each value of each* ***design property being used in a given design context.***

Based on the above assumptions, the goal of identifying learning approaches capable of acquiring synthesis knowledge is transformed into two subgoals:

- Identify a useful probabilistic concept which provide an adequate measure of belief that a property value can be used in some design context.
- Identify an effective method for the construction of a probability estimation function that implements the selected probabilistic concept.

In this report we demonstrate that the solution to these two subgoals (and to the original goal) may be achieved by applying connectionist learning approaches. Our connectionist-based solution relies on the following:

- A recent fundamental theoretical research result demonstrates that a group of connectionist learning methods is capable of estimating Bayesian *a posteriori* probabilities.
- A representative method from the above group of connectionist learning methods may be successfully used to acquire and store the probability estimation function, thus, allowing inductive learning to be the method of construction for this probability estimation function.

This report discusses and illustrates the use of a connectionist learning approach to acquire and represent a probability estimation function that can be used to achieve our goal — acquire synthesis knowledge for its subsequent reuse.

1J MOTIVATIONS FOR USING CONNECTIONIST APPROACH

Building a synthesis system in problem domains where there is a weak, or nonexistent, domain theory, and where mathematical optimization methods are not amenable is a difficult task. For these types of problems, an inductive machine learning approach may possibly be used to capture, and consequently permit reuse of, the synthesis knowledge embodied in the past design cases.

While some very good research has been conducted on how symbolic machine learning techniques can be applied to the synthesis problem [Reich 91, Lu 92], little research has investigated how neural networks might be used to capture and subsequently use synthesis knowledge.

The research reported here has focuses on connectionist learning methods for SKAU. The following are the motivations for investigating the connectionist learning methods for SKAU:

- Connectionist systems are adaptive systems capable of capturing complex nonlinear relationships. Both the theoretical results concerning representational capabilities of connectionist systems [Cybenko 88, Hornik 89], and the results of applications of connectionist models to difficult learning problems in control [Pomerleau 89], signal prediction [Lapedes 88] and pattern recognition [Sejnowski 87, Qian 88], illustrate that connectionist systems are capable of learning complex relationships from a sample of input-output pairs representative of that relationship. The relationships between design variables can be very complex and it is felt that the power of neural networks can be employed to induce these relationships from a representative training sample of previous designs.
- The knowledge representation paradigm in connectionist systems offers attractive capabilities. The distributed knowledge representation found in connectionist systems has been shown to lead to the kinds of operations that are reminiscent of operations of the human brain: content addressable memory recall (i.e., reconstruction), noise resistant computation, and gracefully degrading computation [Hinton 86]. Obviously, a learning system for SKAU could beneficially employ these kinds of

operations, particularly in the context of design domains with weak synthesis knowledge. In addition, distributed knowledge representation allows automatic generalization [Hinton 86], which is one of the key requirements for inductive learning systems for SKAU.

- Connectionist learning systems have exhibited consistently good performance on tasks similar to SKAU. The application of connectionist systems that motivates our research is neural pattern recognition [Le Cun 89]. Since the domains of interest to this research are assumed to have weak formal knowledge of the synthesis process, one approach to acquire the actual synthesis knowledge is to search for meaningful patterns of features in past successful designs. Recognition of such patterns and their generalization to new design scenarios could assist the synthesis process in producing "good" designs. The existence of a long list of successful neural pattern recognition applications, illustrates the feasibility of a connectionist approach to SKAU [Le Cun 89, Tesauro 90, Pomerleau 89].
- Connectionist learning systems estimate Bayesian *a posteriori* probability. The result that a class of connectionist learning systems is capable of estimating Bayesian *a posteriori* probability provides justification for employing a neural pattern recognition system in performing SKAU. The fundamental difference between applying a pattern recognition system to the task of SKAU and other similar tasks is the nonunique nature of the mappings used in synthesis. The acquired synthesis knowledge has to preserve all of the relationships that hold between design properties for all of the multiple possible alternatives. This is fundamentally different from the requirement posed in other tasks that search for only one unique solution to the problem. By incorporating the concept of *a posteriori* probability estimation we are able to satisfy the need to represent both the relationships and the multiple alternatives for SKAU. In addition, this result provides a rigorous theoretical background for analysis of the performance of connectionist systems and for their use in practical applications.
- Connectionist systems allow the use of Bayesian learning method. Some of the recent research has focussed its attention on the analysis of learning in connectionist systems by following the Bayesian learning method [Buntine 91]. While we currently assume that the uniform convergence method applies (i.e., we assume the existence of a sufficiently large sample size and ignore factors that appear as priors in the learning model), the Bayesian learning method provides important results for the case of limited sample sizes. Although the Bayesian approach has not been applied in this research to date, we acknowledge the importance of these results for future work.

1.4 ORGANIZATION OF REPORT

The following is the organization of this technical report:

- Chapter 2 presents the synthesis process considered in this research and discusses one type of machine learning (i.e., inductive learning) approach that is capable of SKAU under the conditions specified in section 1.1. We present the characteristics of the synthesis process in general and of the synthesis process in domains with weak or nonexistent synthesis knowledge. From this discussion we derive basic requirements for a learning system capable of learning synthesis knowledge. We also describe two alternative inductive learning approaches which have been employed for SKAU.
- Chapter 3 develops a probabilistic approach to synthesis. The approach is based on the assumption that it is possible to construct a sufficiently accurate estimation of *the probability of each possible* ***value of each design property being used in a given design context.*** **The assumptions about the synthesis** process considered in this approach are presented and a probabilistic approach for synthesis knowledge acquisition and use (SKAU) is described. We discuss the minimal requirements on the accuracy of the probability estimation function to be useful for synthesis. The approach is illustrated on a number of examples taken from a realistic design domain.
- Chapter 4 provides a detailed account of NETSYN - a neural network approach for synthesis knowledge acquisition and use (SKAU). The basic idea of the approach is to use a connectionist learning

system to acquire and represent the probability estimation function introduced in Chapter 3. Theoretical results relevant to the expected performance of this approach are presented along with a discussion of how this learning approach fits into a more general machine learning framework. Connectionist representations are then discussed. Discussion of the architecture and the training of NETSYN is provided.

- Chapter 5 describes the test methodology used for evaluating the performance of NETSYN and its comparison to two symbolic inductive learning approaches: ILLS and ECOBWEB. A series of results is presented for the performance and comparison of NETSYN to these symbolic learning *approaches.*
- Chapter 6 gives directions for the future work concerning the overall goal of NETSYN: application to a realistic design synthesis task.
- Chapter 7 summarizes the results presented in this technical report.

Chapter 2

Background

In this Chapter, we describe in more detail the synthesis process considered in this research and discuss one a type of machine learning approach that is capable of SKAU under the conditions specified in the previous chapter.

First we present the characteristics of synthesis process in general and of the synthesis process in domains with weak or nonexistent synthesis knowledge. Then, we present basic requirements for a learning system capable of learning synthesis knowledge in such a domain. Finally, we describe two alternative inductive learning approaches which have been employed for SKAU.

2.1 CHARACTERISTICS OF THE SYNTHESIS PROCESS

The engineering synthesis process is often viewed as *an exploration of a space* of possible solutions for a posed design problem. However, the directions for performing this search are rarely given in an explicit form for any engineering design domain.

Design problems are *ill-structured* and the design problem formulation is itself a dynamic process where new design knowledge causes the reformulation of the design problem and, subsequently, reformulation of the synthesis search space [Coyne 90]. In certain design domains, it may be assumed that the dynamics of the design problem formulation are slower than the actual synthesis space traversal. This obviously holds in more mature design domains where the knowledge of the design synthesis space (i.e., design variables and the relationships between them) has been reinforced through the many successful outcomes of synthesizing designs for given design specifications.

The synthesis process in many design domains is characterized by generating alternatives which *"satisflee"* the given design requirements [Simon 81]. Therefore, the outcome of a synthesis process in such a design domain is a collection of candidate designs that satisfy initial design requirements. The nonunique nature of the synthesized designs poses a requirement on the designer to consider and evaluate more than one alternative solution.

Another characteristic of the synthesis process is the utilization of various pieces of information of different nature. Both *qualitative* and *quantitative* data are combined by a designer during the design synthesis process. The ways in which such data are combined to decide which part of the search space to traverse are often hard to express or analyze rigorously using classical mathematical approaches.

In the process of generating alternatives during the synthesis process, the designer will narrow the search space by considering only selected subintervals of values of design variables. Eventually, the design variables will be assigned specific values to give a more complete design context The decision to select a particular variable value will influence the feasible ranges of other design variables. The *relationships between design variables* are in general *complex* and *many-to-many:* constraining one design variable will affect the selection process of all other variables, while a particular design variable assignment may be inconsistent with previously constrained design variables and cause an infeasible design. These relationships are results of designer's experience and are reinforced throughout many successful synthesis processes. The designer's ability to capture and to *generalize* these relationships from a finite number

of previous design episodes and, subsequently, to use this knowledge in new design situations is a crucial capability for successful synthesis.

In the frameworic of a formulated synthesis problem, where the dimensionality of the search space (i.e., properties to be specified) is known but the synthesis knowledge (i.e., relationships between design properties) is weak or nonexistent, the only source of synthesis knowledge tends to be the past design experience. In such a design situation, where an experienced designer is faced with a new design problem, the first step is to perform some limited (to the extent to which the synthesis theory is available) analysis of the design problem and then to proceed with the synthesis process. The knowledge of first principles of behavior in the considered engineering domain is of limited utility in constructing the design alternatives. To solve the design problem, the designer relies on his or her past experience to recognize the similarities of the new situation to the design problems he or she has encountered in the past. The designer recognizes the important characteristics of the design problem and focuses his or her attention on these important considerations. This subjective, experience-based knowledge of the designer will govern the synthesis process of generating alternative design solutions.

The identified nature of the synthesis process in domains with nonexistent or weak domain theory forms the basis for deriving the requirements for systems that are to capable of capturing synthesis knowledge and facilitating its reuse in new design problems. These requirements are presented next.

2.2 REQUIREMENTS FOR A LEARNING SYSTEM

Three requirements are identified as necessary ingredients of any system capable of capturing synthesis knowledge and using this knowledge in novel design situations [Reich, 1991]:

- The ability to capture many-to-many mappings between form, function and behavior design properties. In general, one must consider all design properties for which values have been fixed or ranges constrained when making decisions about yet unbound design properties. Considering only the design specifications and disregarding values of design attributes fixed in the course of synthesis will in general lead to infeasible solutions.
- The ability to recall or generate multiple design alternatives satisfying the particular combination of specifications. As described earlier, the synthesis process is concerned with generating multiple alternatives for the same design specifications.
- The ability to generalize from the individual design experiences—i.e., generating acceptable solutions in design situations not seen before. A learning system is expected to behave well in the new design situations that are similar to, but not identical to, previous successful design episodes.

2.3 INDUCTIVE LEARNING APPROACHES FOR SKAU

Inductive learning approaches are candidate machine learning approaches for acquiring synthesis knowledge and allowing the reuse of this knowledge in new design situations. A brief discussion of two alternative inductive learning approaches for synthesis knowledge acquisition is presented next (for a more exhaustive treatment of learning methods for this task, see [Reich 91]).

Inductive learning approaches can be grouped in two main classes:

- The first is the class of supervised learning approaches. In supervised learning approaches, learning is based on the availability of a correct answer for any input description of a situation that needs to be classified or to which a value is to be associated. The learning system has to adjust itself according to the representative training cases. The goal of learning is that the system ultimately learns correct associations between input and output patterns.
- The second is the class of unsupervised learning approaches. In this case, the learning target is not specified explicitly in terms of correct results for a set of training cases. The only available informa-

tion is in the correlations of the input data. The system has to discover regularities in the training set and to create categories based on the discovered correlations. These categories are the basis for classification of the new input.

An example of an unsupervised learning approach to synthesis knowledge acquisition is the ECOBWEB learning algorithm used in the Bridger system [Reich 91]. The ECOBWEB learning algorithm is used to form meaningful clusters out of design training cases. These clusters can then be used to complete the synthesis process for given partial design descriptions. ECOBWEB utilizes a probabilistic approach and it uses a heuristic performance measure to compare alternative classifications with respect to their utility in the classification of designs. These classes are then used to classify a new, partially complete design specification. Once the input specification is classified, the whole classification hierarchy can be used to synthesize design property values consistent with members of that class.

The supervised learning approach has been criticized as being inappropriate for learning synthesis knowledge [Reich91]. Early learning systems used this approach in a straightforward manner which prevented them from learning "complete" relationships between design properties: only the mapping from the set of fixed specification properties to design description properties was captured. No attention was paid to the influence that fixing some design attribute values might have on the feasible ranges of other design properties. Hence, many-to-many relationships between design properties could not be captured and the use of the supervised learning approach was rightfully judged as inappropriate for learning synthesis knowledge.

In our research we have followed the supervised learning approach. However, the fundamental difference between our approach and the previous approaches using supervised learning is that our basis for classification is not just the subset of design properties declared to be specifications. Rather, all design properties known in a given design context are used to determine the values of the remaining, unknown design properties. The approach is basically one of predicting the *a posteriori* probabilities of each possible value of each unknown property for the given set of known property values. Using this approach, it is possible to account for the many-to-many relationships within synthesis knowledge. The next section gives a detailed description of this approach.

Chapter 3

A Probabilistic Approach To Synthesis

In this chapter we develop aprobabilistic approach to synthesis. The approach is based on the assumption that it is possible to construct a sufficiently accurate estimation of *the probability of each possible value* ***of each design property being used in a given design context.***

First, assumptions about the synthesis process considered in this research are presented. Then, a probabilistic approach for synthesis knowledge acquisition and use (SKAU) is described. Next, the minimal requirements on the accuracy of the probability estimation function to be useful for synthesis are given. Finally, the approach is illustrated on a number of examples taken from the domain of Ml [Gupta 91] - a knowledge based system for synthesizing single-board computer systems.

3.1 ASSUMPTIONS ABOUT SYNTHESIS

In this section we list several assumptions about the kind of synthesis processes we address in this research and for which we attempt to create a learning system.

We view a design as a collection of property-value pairs or, equivalently, collection of design decisions. The design synthesis process is then equivalent to a sequence of assignments of property values which ultimately leads to a complete design definition. Some property values are determined a priori, representing design specification values. All design decisions are made in some design context (i.e., a partial design description).

To further elaborate on our assumptions, consider a representation of a simplified design process shown in Fig. 6. On the left side of the figure, an initial design context is shown. To complete the design description, the values of three design properties need to be determined (one specification property and two design description properties). Each design property may take on one out of three distinct values (graphically represented by an rectangle, triangle, and an oval). In the initial context, only the design specification value is determined (denoted by a solid shape).

The first task in the design process is to consider all of the design properties for which values have not been determined and to select one and assign its value. In the most general case the designer would like to consider every unresolved design property and decide which values are feasible for the given design context. Then, based on some design strategy, the designer could select a property and its value which he believes will lead to a good design. In the figure, completing this task gives the intermediate state of design context (1) and the new, more complete, design contexts (2 or 3). The intermediate state (1), represents consideration of unbound design properties. The different shades of symbols representing values depict that each value may appear in a given design context (the darker shades indicate greater level of belief that the value applies for the given design context). Alternative design contexts (2 and 3) are results of different design decisions. Similarly, going from either of two partial design contexts (2 or 3), the designer considers the values that the remaining design property may assume. Intermediate states of the design context (2a) and (3a) represent a degree of belief that the designer may have that each unbound property could be assigned each of its possible values in design context (2) and (3), respectively. The final design contexts (2b and 3b) are representations of outcomes of two alternative decision sequences.

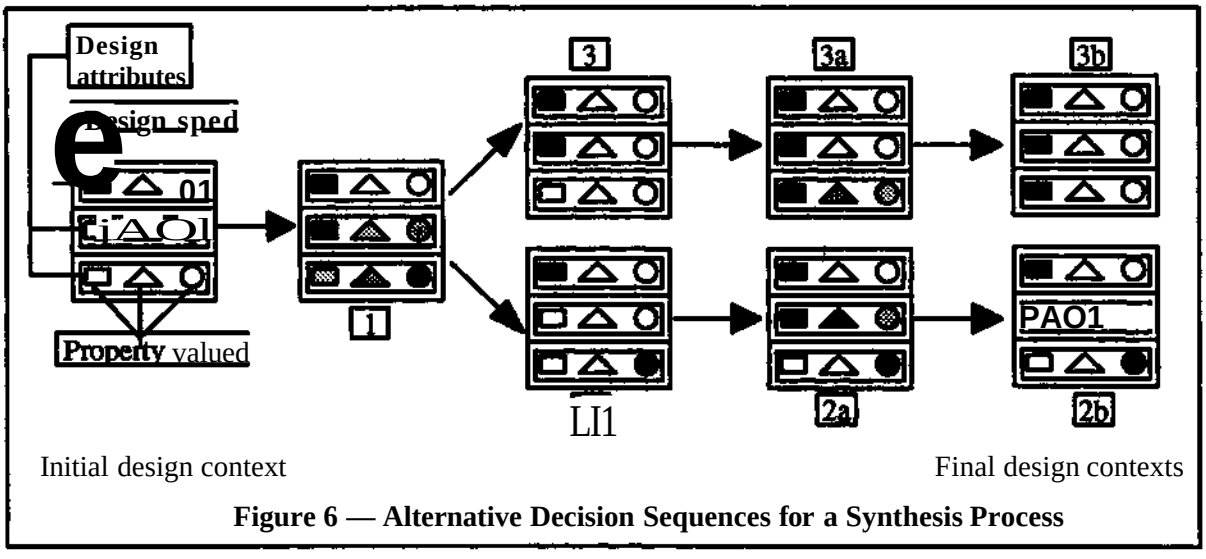


Figure 6 — Alternative Decision Sequences for a Synthesis Process

A particular point worth noting here is that a designer usually has a number of alternative choices for making synthesis decisions. The designer utilizes his or her subjective knowledge of the synthesis process (i.e., knowledge of relationships among property values) to determine which property value will be used in the given context (which changes as decisions are made); consequently, he or she decides which particular property value to determine next. We are specifically interested in acquiring this subjective knowledge that is used to select particular property values for given design contexts.

3*2* *a posteriori* PROBABILITY ESTIMATION FOR SYNTHESIS

3.2.1 General Idea

We cast the described synthesis process into a probabilistic framework. The probabilistic concept we select to employ in this research is Bayesian *a posteriori* probability. This concept has been long recognized as useful relative to engineering planning and design, as it incorporates engineering judgment and observational data in a formal framework [Ang 75].

To use the concept of *a posteriori* probability in a traditional way, one has to know a priori probabilities of classes of phenomena one deals with as well as likelihood functions (i.e., class conditional probability functions) of the measurements (i.e., dependent variables) that one can obtain for these phenomena. Measurements may be continuous or discrete values and they are represented in the form of a measurement vector. The Bayesian *a posteriori* probability p(Q IX) represents the conditional probability of class Q given the input measurement vector X. Use of Bayes rule allows it to be expressed as follows:

$$p(C_i \mid \vec{X}) = \frac{p\ (X\ I\ Q)\ p(C_t)}{p(X)}$$

where X is a measurement vector
p (XI Q) is the likelihood of producing the measurement X if the class is Q
p (Ci) is the a priori probability of class Q
p (X) is the unconditional probability of the measurement

In our synthesis framework, the phenomena in which we are interested are the design property value assignments for which we know in advance the classes (i.e., values) and the measurements are all bound

design properties that make a current design context (i.e., the measurement vector). The *a priori* probabilities can in general be estimated fairly well; however, the likelihood functions involve assumptions about the probability distributions of values of the phenomena (design properties) with respect to the measurements (design context). Instead of indirectly estimating *a posteriori* probabilities of each property value assignment by computing probabilities according to Bayes rule, we will consider an alternative approach of directly estimating *a posteriori* probabilities (hereafter, probabilities) using a connectionist learning model. This is a subject of Chapter 4.

One way to use the *a posteriori* probability concept is to use it as a *probability estimation of each value of each unbound design property* being used in a given design context Such probabilities will be very useful information during synthesis. Consider Fig. 7, which illustrates this idea. Each property value is denoted by a circle. The properties are divided into specifications and form and behavior properties. The assigned properties form the current design context which is operated upon by the probability estimation function. The result of this operation is a set of estimated probabilities of each value of each unbound design property being used in the given design context The shaded circles on the right hand side of the figure represent these probability estimations for each property value. Daiker shades represent higher probability of the value appearing in the given design context.

For a given set of design specifications, or any partial description of design, by using a probability estimation function we are in a position to do the following:

- Acquire the probabilities of the values appearing for each unbound form and behavior property.
- Reason about which values of unbound design properties are viable alternatives in a given design context

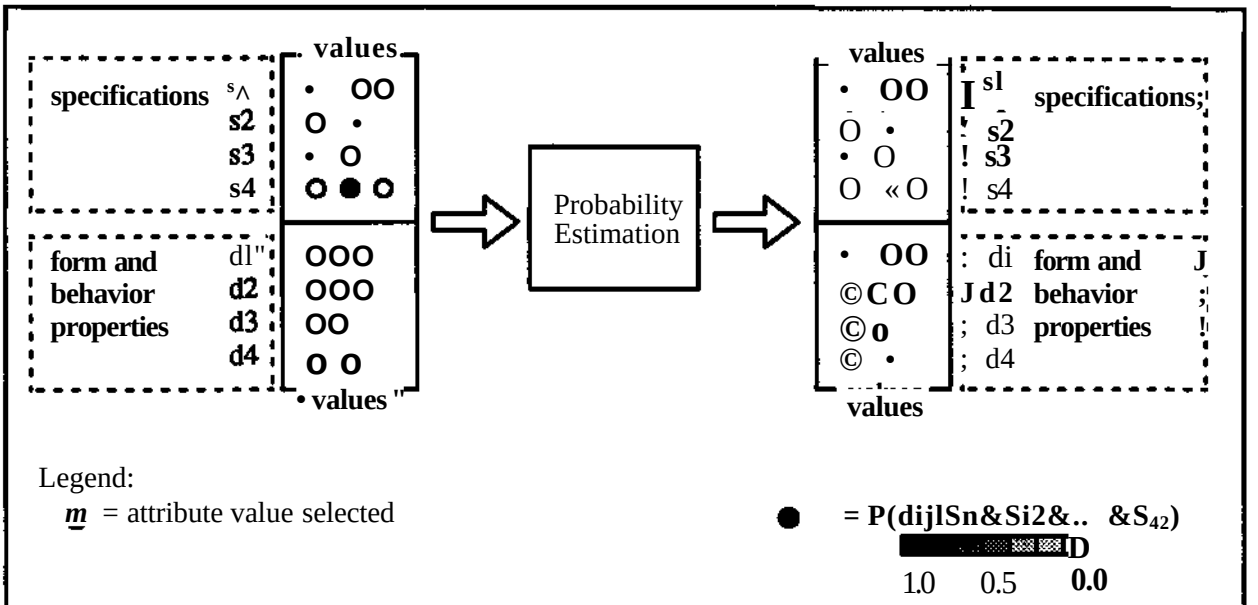


Figure 7 - The Probability Estimation Function

Given such a tool, one can imagine a synthesis process being completed by iteratively applying the following steps, starting from some design context (i.e., design specifications) and ending with the completion of the design (see Fig. 8):

- Apply the probability estimation function to acquire the probability estimation of each value of each unbound design property for the given design context.
- Consider probability distributions over the values for each unbound design property and decide to which property to assign a value next (this step may be subject to a specific strategy of selecting the **next property to bind).**

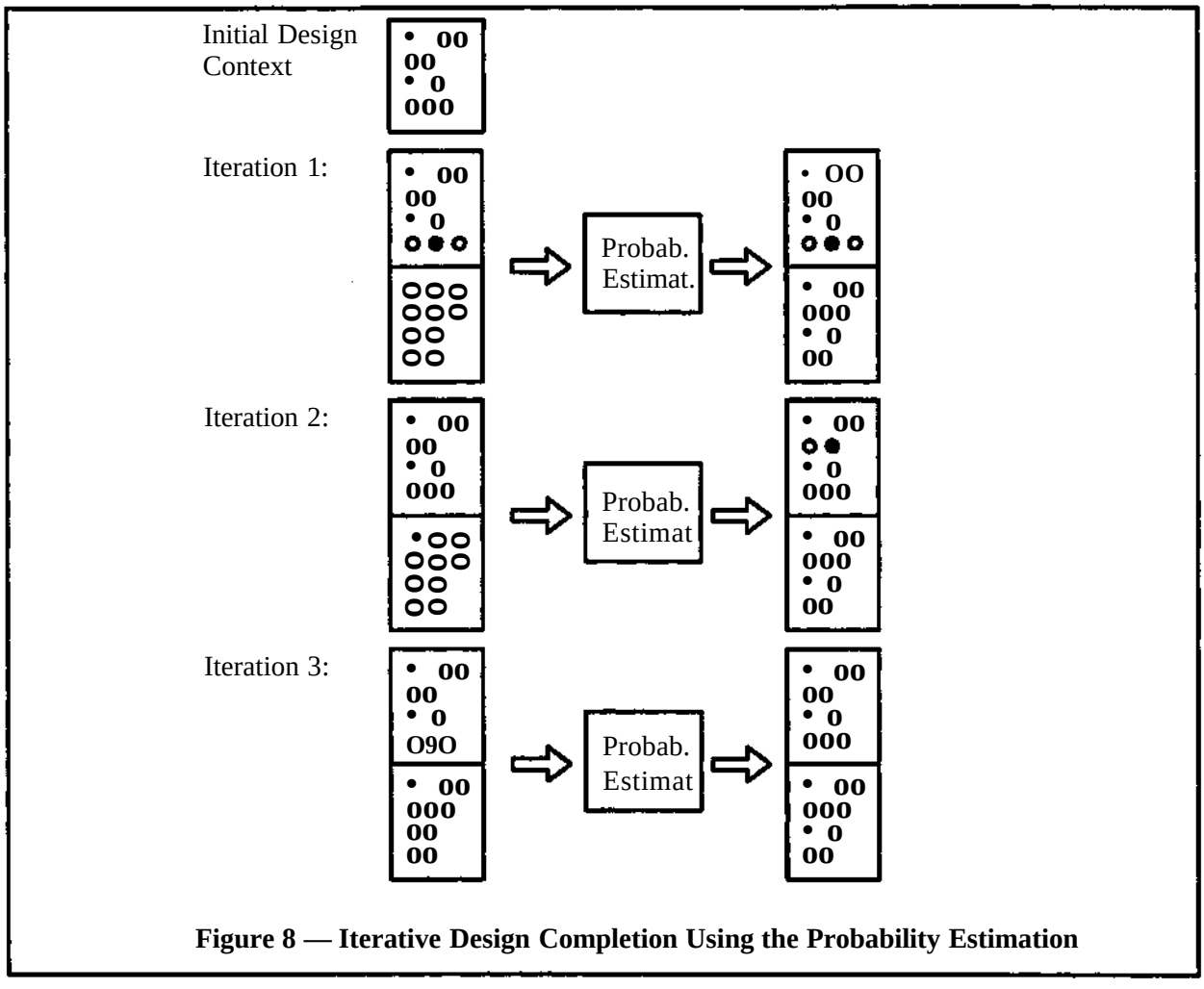


Figure 8 — Iterative Design Completion Using the Probability Estimation

- Consider the probabilities for values of the selected property and decide which value to assign to the property.
- Update the design context by fixing the property value.

3.2.2 **Required Accuracy for the Probability Estimation Function**

From the perspective of the usability of this approach, the probability estimation function should: (1) estimate non-zero probabilities for design property values which have appeared in a given design context; (2) estimate zero probabilities for the property values that have not been in that design context; and (3) correctly rank the most probable design property value. This minimalist accuracy requirement is necessary as the relationships between design properties in a realistic design domain may be very complex, and construction of a probability estimation of an arbitrarily high accuracy may be impossible.

Although the above requirement is sufficient to render the proposed approach useful, it still may be too hard for practical design problems. It is likely that for any problem of this complexity only an approximation of the the probability estimation function will be possible. We will return to this accuracy requirement again in Chapter 5, where we deal with evaluation of the proposed approach.

3.2.3 **Approaches to Probability Estimation**

One may employ different approaches to construct the probability estimating function. These approaches may in general be classified as exact probabilistic methods, approximations of exact probabilistic meth-

ods, or heuristic methods. By following an exact approach, one needs to follow axioms of probability theory and provide a rigorous proof that the constructed probability estimating function for the considered problem has the desired properties. While this may be possible for small problems, the difficulty of real synthesis problems prevents the actual usage of this approach. Methods for approximating these probabilistic methods may be a more viable approach to complex tasks such as synthesis. The quality of the performance of these approximating methods depends on the problem — typically, the size of the sample that is used to construct the probability estimation function and the dimensionality of the problem. Therefore, the usefulness of this approximating approach varies with the conditions that the actual synthesis problem may impose. Finally, the heuristic methods sacrifice exactness for feasibility and decreased complexity in constructing the probability estimating function. In addition, these methods are sometimes employed to introduce problem-dependent biases, which cause the estimation of the probabilities to follow some assumed probability distributions peculiar to that problem.

In the next chapter we present a connectionist approach to constructing the probability estimation function that actually estimates *a posteriori* probability.

3.3 EXAMPLE USAGE: MICON SYNTHESIZER VERSION 1 (Ml)

This section illustrates the use of the proposed probabilistic approach within the framework of a synthesis system for the design of computer systems. First, a description of the synthesis tool in MICON (Ml) and the system-level synthesis task performed by this tool are given (this part is based on [Gupta 91]).

3.3.1 MICON Synthesizer Version 1 (Ml)

Ml (MICON Synthesizer Version 1) is a part of the MICON system that designs single-board computer systems [Gupta 91]. Specifically, M1 is a knowledge-based synthesis tool that satisfies high level specifications by performing system-level synthesis of computer systems from a set of components. The objective of system-level synthesis is to create a complete and operational computer system capable of performing general-purpose or special-application computing. The components of design are integrated circuits (i.e., ICs) or application-specific IC libraries. Input specifications to Ml are based on a functional description of the elements; no details of components and their specifics are included in the specifications. Typically, the result of the system-level synthesis consists of a configuration of CPU (i.e., central processing unit), memory system, I/O (input output) component, bus interfaces, and other supporting circuitry.

Two characteristics of this synthesis task are:

- There exists no model or language to clearly define the function of an artifact being designed.
- There exists no complete and well-defined theory relating structure (i.e., form) and behavior of the artifact.

Being a realistic problem, the following are some properties that make the synthesis process in this domain hard:

- The design space is large as the parts can be used in a variety of ways to fulfill functional specifications. Combinatorial explosion is a result of an attempt to thoroughly search over the whole design space.
- There are interactions between the design sub-problems. Ml divides the synthesis problem into a set of sub-problems and cannot, in general, determine that the design for a subsystem will yield a satisfactory complete design.
- There are complex interactions among the components. The relationships between components depend on several factors, including the way in which a component is programmed.
- The knowledge base is rapidly evolving. New ICs are constantly being developed, leading to improved components and new design styles.

Based on the above description, this synthesis task belongs to the class of synthesis problems with well defined design spaces but weak synthesis knowledge.

In the following subsections, we illustrate several ways in which the proposed probabilistic approach for SKAU could be used by M1. We consider the least complex subdomain of M1—CGA controller design — for which the synthesis space description is given in the form of the synthesis hierarchy in Fig. 9. In general, Ml uses an "AND-OR" hierarchy to represent the design space. The synthesis hierarchy for CGA controller design represents a part of this "AND-OR" hierarchy. The state of the Mi's knowledge base for the CGA controller has dictated the shape of, and the size of, this hierarchy. The arch across the top level branches indicates the top node being "AND" node. The lower level nodes are "OR" nodes (indicated by heavier lines with ordinal numbers in circles in front of the alternatives) and "AND" nodes (indicated by lighter lines). The only decisions that need to be made correspond to "OR" nodes. Therefore, only those components corresponding to "OR" nodes are considered in the following examples. In addition, a subset of performances and specifications listed in the same figure are included in the illustrations: cost, RAM (Random Access Memory) size, ROM (Read Only Memory) size, and the speed of the board.

33.2 Example 1 — Completing a Design for CGA controller

The initial example illustrates the idea of completing a design for a given design specification (Fig. 10). Fòr the given specifications (cost range, RAM size, ROM size, and desired speed of the board), the task is to estimate in every step of the design completion process the probability of each value of each unbound design property (RAM chip, ROM chip, Address Decoder, and number of Address Decoder components) being used in the current design context The outcome of the represented design completion process can be read from the final iteration (4) in Fig. 13: RAM chip = 6264, ROM chip = 27512, Address Decoder = OR gate, number of Address Decoder components = 1. Also note that the synthesized solution is an outcome of developing only a single alternative within the synthesis space.

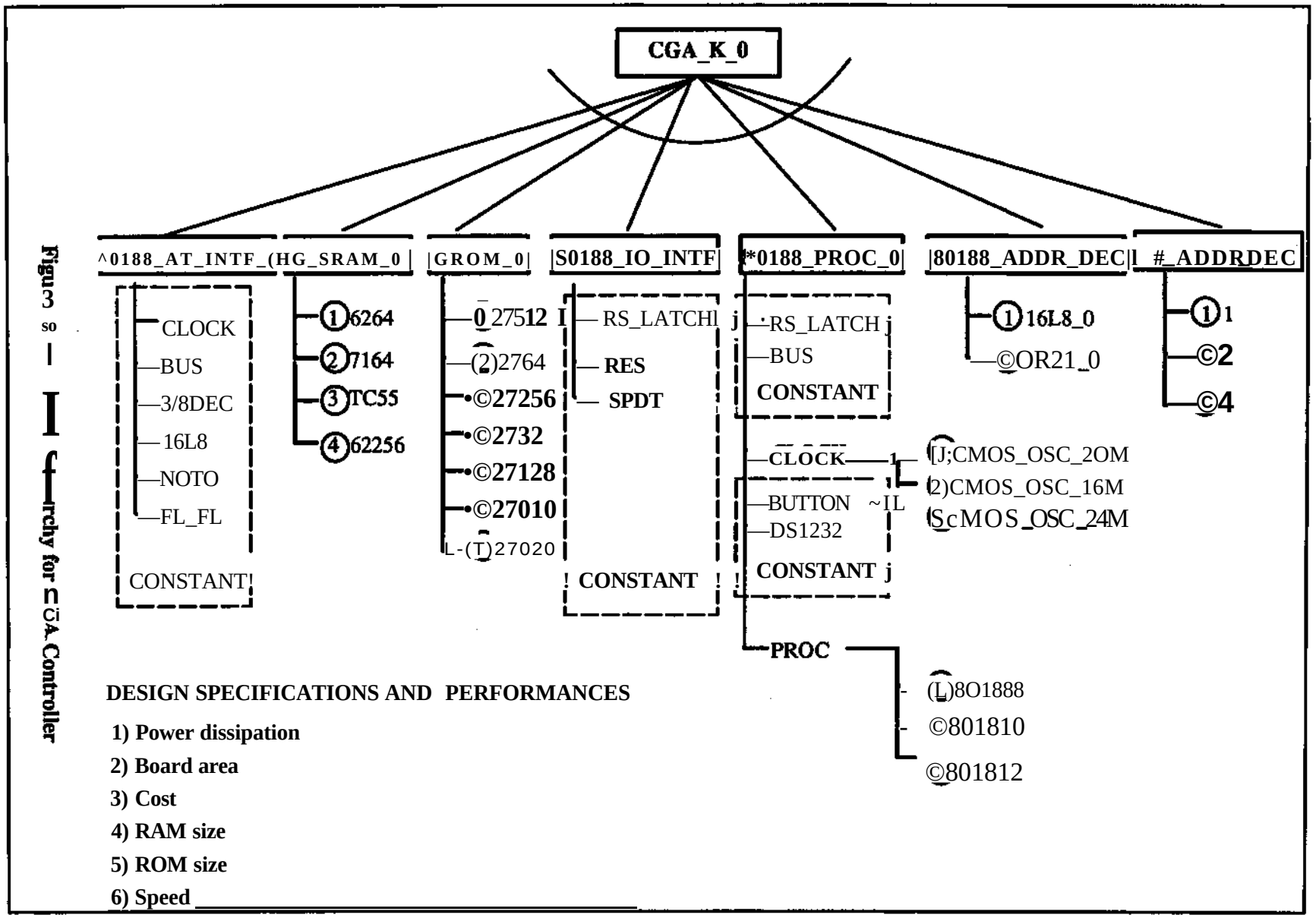
33.3 Example 2 — Estimating Behavior for Partial Designs

In the case where a behavior design property is included among the design properties for describing design context, it is possible to estimate the behavior of the partially completed designs in the early stages of the synthesis process. In Fig. 11, an example of this use of a probability estimation function is illustrated. The indicated input specification consists of desired RAM size, **ROM** size, board speed, and a constraint on the desired ROM chip type. By applying the probability estimation function, the probability of cost being above $150 was estimated to be the highest of all considered cost values, representing the prediction that the cost for this configuration will be above $150.

33.4 Example 3 — Recognizing Inconsistent Specifications

If we estimate the probabilities of all properties (including specification properties) based on all other properties, then it is possible to determine whether the given specifications are consistent. Fig. 12 illustrates this by showing how the output of the probability estimation function indicates different values for properties for which values are fixed as a part of specification (e.g., the cost was set to be in the range $5(>-$60 on the input, but the estimated probability indicates that this value for the cost is not consistent with the rest of the specifications).

To elaborate on this example use, consider a subpattern of the input formed by omitting one specification input This is exactly the input that enters the probability estimation function for this specification. The function then estimates the probability for each value of that specification of being a part of considered pattern (i.e., design context). Therefore, the estimation function is perfectly capable of estimating as nonzero those probabilities of design specification values which are not set on input (and vice versa, estimating as zero those probabilities of design specification values which are set on input) since it does not consider the input values for this design specification in its estimation process. The cause for this difference



Figu 3 so — I f rchy for C A Controller

DESIGN SPECIFICATIONS AND PERFORMANCES

1) Power dissipation

2) Board area

3) Cost

4) RAM size

5) ROM size

6) Speed

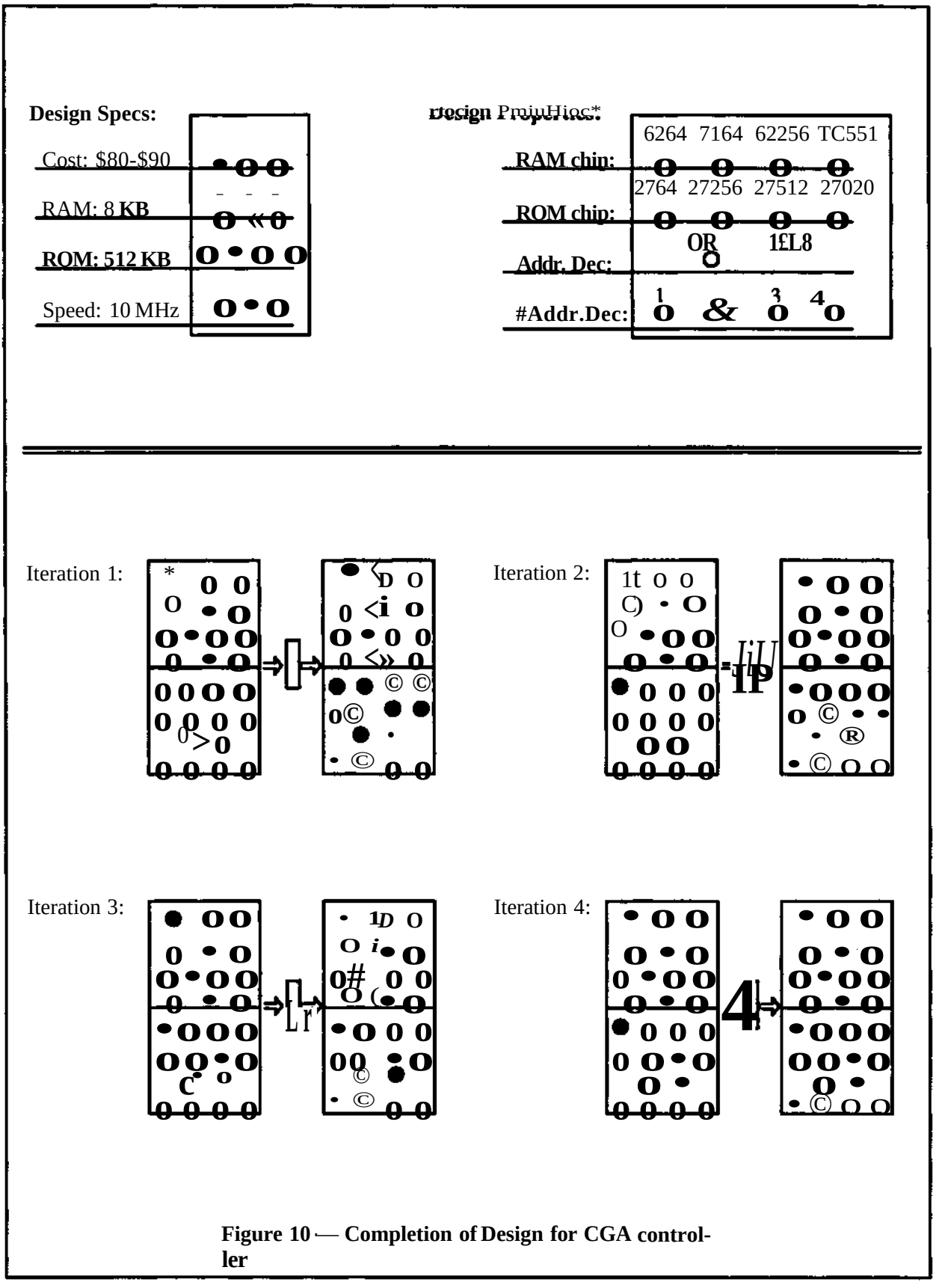


Figure 10 — Completion of Design for CGA controller

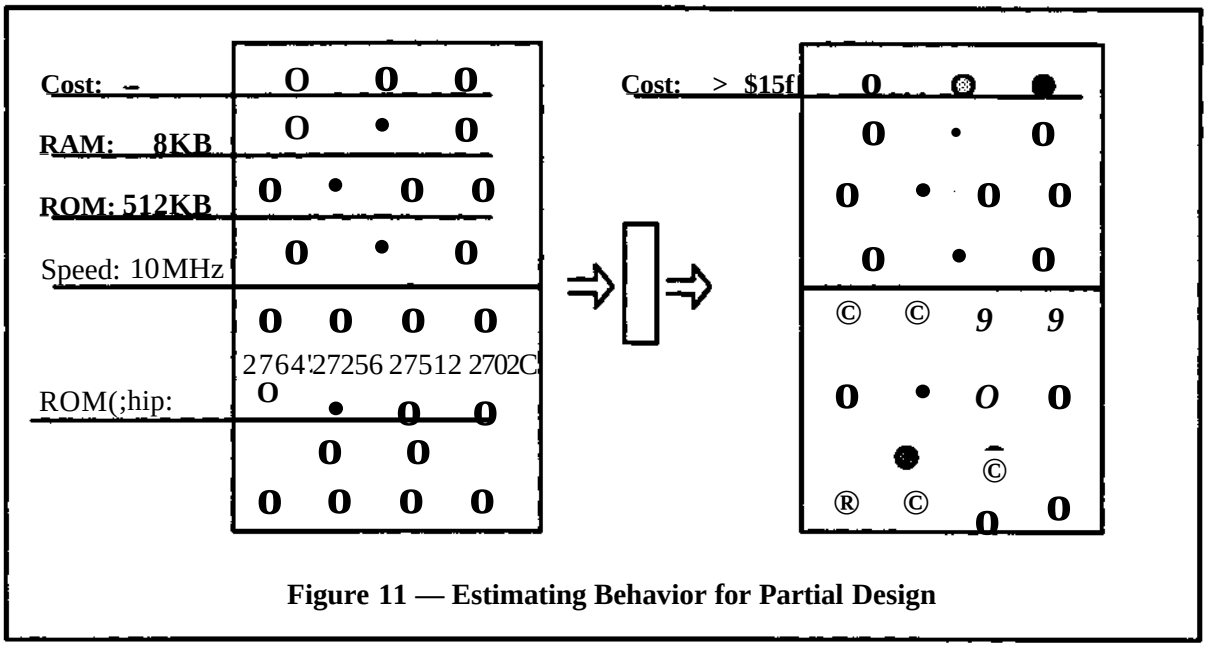


Figure 11 — Estimating Behavior for Partial Design

between the input value and the estimated probability of that value is that the selected design properties have created a design context that is inconsistent with the remaining selected property value. This inconsistency would not have been encountered if the decision making process (i.e., property assignment) had been conducted one decision at a time.

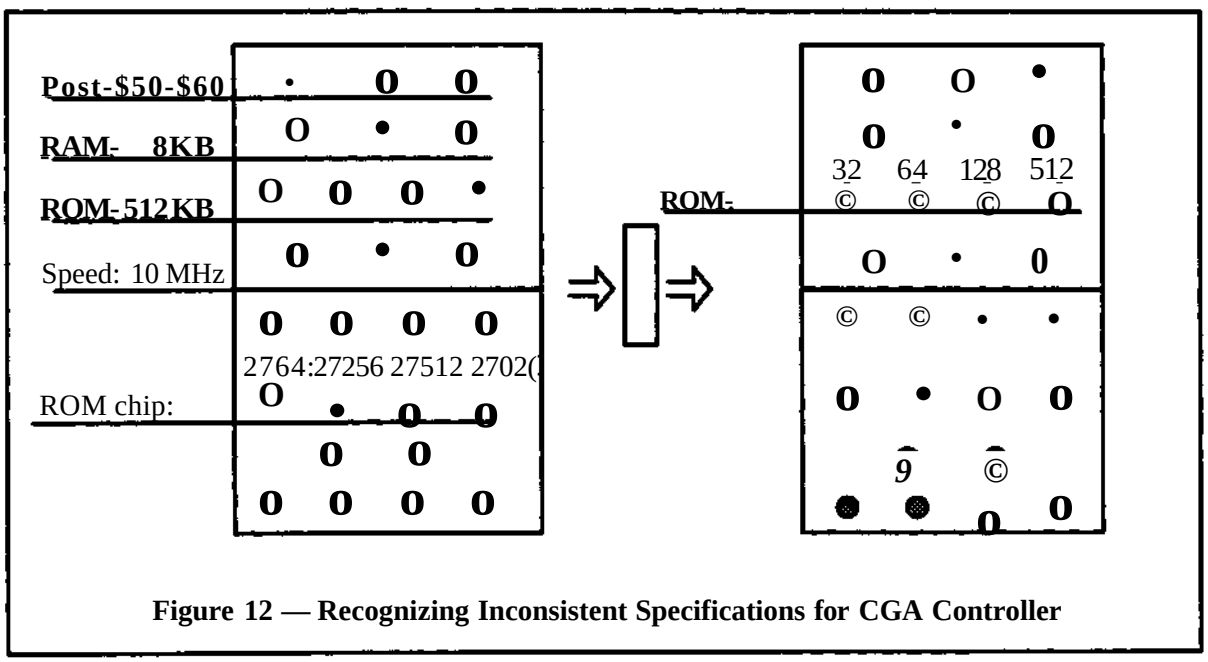


Figure 12 — Recognizing Inconsistent Specifications for CGA Controller

From the examples shown, this approach based on probability estimation could supply useful information during the synthesis process that can be used to guide the process more directly to the location in form and behavior space that satisfy the given requirements. The next chapter describes a connectionist-based approach that acquires and represents this probability estimation function.

Chapter 4

NETSYN — a Connectionist Approach to SKAU

This chapter provides a detailed account of NETSYN - a neural network approach for synthesis knowledge acquisition and use (SKAU).

First, the basic idea of the approach is given by mapping a connectionist learning system onto the probability estimation function introduced in the last section. Next, some theoretical results relevant to the expected performance of this approach are presented along with a discussion of how this learning approach fits into a more general machine learning framework. Connectionist representations are then discussed. Finally, a discussion of the architecture and the training of NETSYN is provided.

4.1 A CONNECTIONIST *a posteriori* PROBABILITY ESTIMATION

As discussed in subsection 3.2.3, construction of a probability estimation function is a difficult task for realistic synthesis problems. This difficulty is aconsequence oftwo properties of realistic synthesis problems:

- The dimensionality of the design space may be very high—there may be a large number of interacting design properties that need to be considered simultaneously in the synthesis process.
- Relationships between design properties may be highly complex.

Our approach to constructing the probability estimation function is illustrated in Fig. 13. We employ feed-forward neural networks (discussed below in more detail) as a mechanism by which to acquire and represent the probability estimation function. Two principal benefits of using a neural netwoik in the role of probability estimation function are:

- The probability estimation function is acquired through inductive learning. We use existing records of previous designs to train the neural netwoik to estimate the desired probabilities.
- The trained network, under assumptions presented below, estimates Bayesian *a posteriori* probabilities, thus allowing us for reliance the established theoretical background of Bayesian learning approaches.

42 THEORETICAL BASIS FOR NETSYN

This section presents some results of fundamental research on which the proposed connectionist approach to constructing the probability estimation function for SKAU are founded.

4.2.1 Computational Theory Background

In the paper by Richard and Lippmann [Richard 91], the following result has been presented, rigorously proven, and experimentally tested:

"For an M class problem, Bayesian probabilities are estimated when the network has one output for each pattern class, desired outputs are 1 ofM (one output unity corresponding to the correct class, all others zero), and an appropriate cost function is used. "

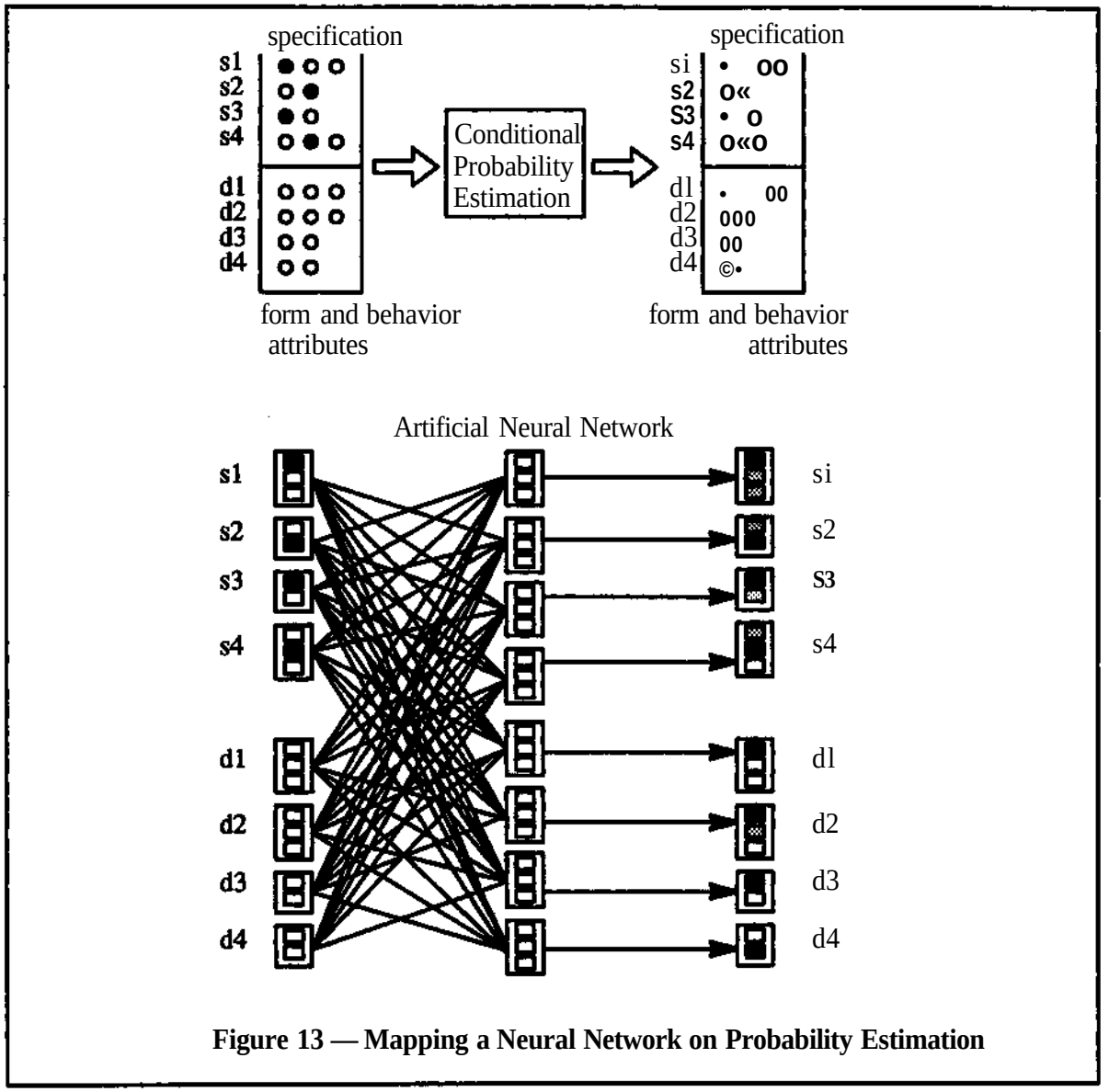


Figure 13 — Mapping a Neural Network on Probability Estimation

The consequence of the correct estimation is that the classification error rate will be minimized, outputs will sum to one, and outputs can be treated as probabilities. Hence, the common rule of thumb that "output values different from zero and one are indications that more training is required or that no classification decision should be made" are not necessarily true [Richard 91]. Such values may actually indicate that classes have overlapping distributions [Richard 91].

However, the estimation accuracy, as reported by Richard and Lippmann, is high only if the netwoik is sufficiently complex, there are adequate training data available, and the training data accurately reflect the actual likelihood distributions and the a priori class probabilities of the problem. In terms of the synthesis problem, these requirements are too stringent, and it is unlikely that any realistic synthesis process will be captured by this approach with high accuracy. In spite of this, as discussed previously in 3.2.2, the estimation of correctly ranked non-zero and zero *a posteriori* probabilities is sufficient to render the probability estimation approach useful for SKAU.

4.2.2 Learning Theory Background

The approach to learning (i.e., constructing the probability estimation function) in NETSYN is based on the assumption of uniform conveigence [Buntine 91]. Uniform convergence method approximates Bayesian method when the number of training cases is large [Buntine 91]. Uniform convergence method requires a sample size that is large enough that the prior term found in Bayesian method becomes insignificant; this method makes no assumptions, but instead have to ignore useful information in the training set. Therefore, it can only guarantee good performance in the worst case [Buntine].

Bayesian learning method, on the other hand, provides an approach for learning in the case of a limited sample size. Bayesian learning method in the case of the limited sample size is approximating a method that produces a classifier that will on average have equal or lower error than a classifier produced by any other method applied to the same training sample [Buntine 91]. This result holds for any size of the sample, not only for the "sufficiently large" sample as in the case of uniform conveigence method. Bayesian Learning Method is a relatively new area of research and were not employed in the research reported here. However, the Bayesian approach to learning is extremely important for a realistic engineering synthesis applications and will be addressed in future work for problems of limited size training sample.

43 CONNECTIONIST (NETSYN) REPRESENTATIONS

4J.I Design Representation

The representation of design in the NETSYN approach is influenced by the neural netwoik-based *approach.* To provide input to a neural network, one has to present an input signal to the input processing units of the network; similarly, to read output from a neural network, one has to read the signals from the output processing units of the netwoik.

Because of the relative structural simplicity of neural networks, the neural network representational capabilities are limited to manipulating feature-vector representations of objects or situations. A straight-forward mapping of a design onto a feature-vector representation is achieved by identifying design properties and their values and performing a one-to-one mapping of these properties and values onto the network input and output units. This a current design representation approach taken in NETSYN.

A need for different design representations may arise in the case where a natural design description is given in the form of some structure of properties that are indicative of the relationship between these design properties (e.g., hierarchical property relationships). The possibility of representing these kind of relationships is muchmore limited in aneural netwoik context than in the symbolic "articulated" representations [Barto 90]. In spite of this limitation, approaches to representation of the hierarchical property relationships exist, and will be addressed in the future work.

43.2 Design Process Representation

The design process representation in NETSYN is in the form of *incremental completion* of design based on the estimated probabilities of property values appearing in a given design context.

The following are some of the characteristics of this design process representation:

- There exists & *feedback* during the design completion process. Each new property value assignment affects all other previous decisions (i.e., property value assignments). This feedback appears as an update of probability estimations for values of previously determined properties.
- The *constraints* on the property value assignments are "soft". The constraints resulting from the above feedback are not enforced; rather, the constraints are simply a comparison between property value assignments and the estimated probabilities of these property value assignments.
- Violating the constraints leads to new designs in the considered design space. By going against the estimated probabilities, one breaks the constraints imposed by the probability estimation function.

Since the function is based on the previous design experiences, the decision to assign a value to the property for which the estimated probability is low or zero, corresponds to the exploration of yet to be considered design configurations inside of the same design space.

4A NETSYN ARCHITECTURE

The approach that we are taking in this research is to use a feed-forward network architecture of the form shown in Fig. 14 and map each property value to a distinct input and a distinct output unit. A signal applied to an input unit encodes the presence of a corresponding property value in the current design context. The transfer function for both hidden and output units was the sigmoid function.

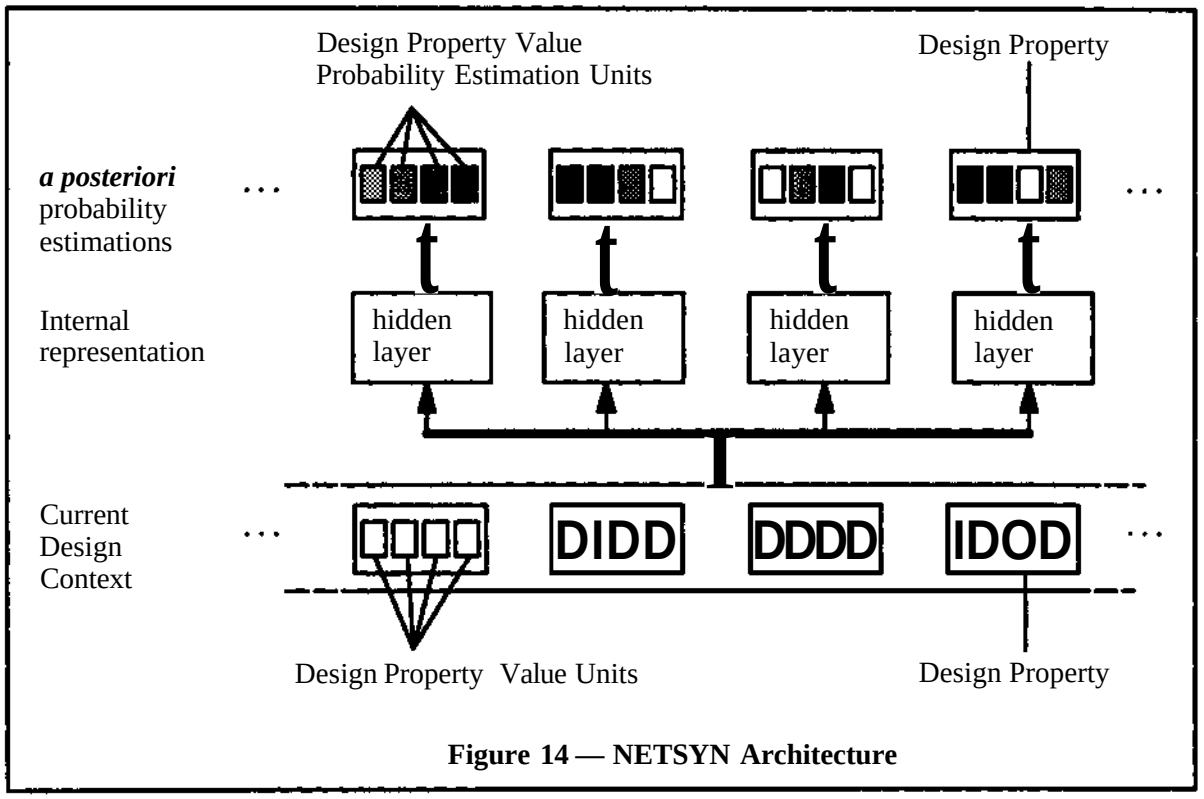


Figure 14 — NETSYN Architecture

The construction of this type of netwoik is modular (Fig. 15): for each design property we create a neural network structure which will act as a probability estimation function for that property. Therefore, our original probability estimation function is composed of as many feed-forward neural networks as there are properties considered in the synthesis task. In addition, we do not provide connections between a property's set of values on the input layer of the netwoik and this property's set of values on network output layer. In other words, fixing the value of a property does not affect the estimation of probabilities for the values of that property (see Fig. 13).

4.5 NETSYN LEARNING

The netwoik learning algorithm used for NETSYN is the standard back-propagation learning algorithm [Rumelhart 86]. Inorderto properly capture the underlying statistical properties of the given training set, the usual Least Mean Squares (LMS) error criterion was substituted with the error criterion similar to the Kullback-Leibler information measure [El-Jaroudi 90]. This error criterion is also known under different names: relative entropy and cross entropy. The advantage of this error measure is that it diverges if the output of one unit saturates at the wrong extreme [Hertz 91]. It was shown that the new error function allows better approximation of *a posteriori* probabilities across the whole range of a property values:

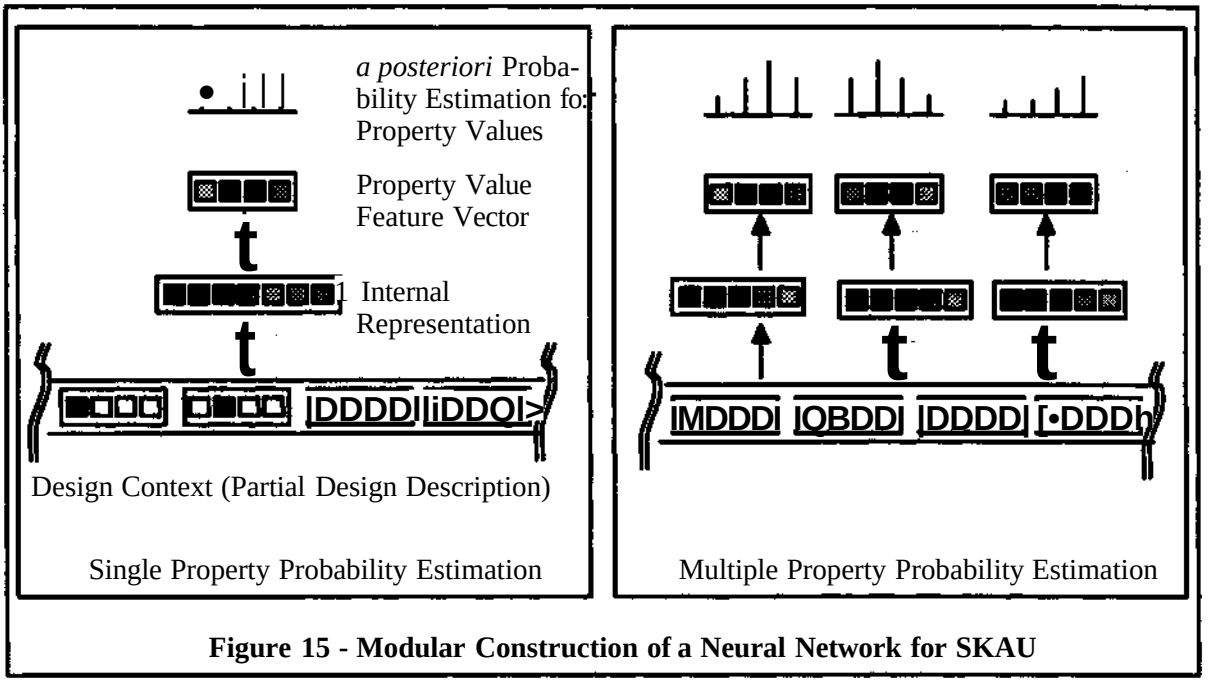


Figure 15 - Modular Construction of a Neural Network for SKAU

both high and low probabilities are preserved in this approximation, which is not the case with LMS error function [El Jaroudi 90].

The strategy of the presentation of cases to the netwoik during training is somewhat unusual from the standard way of the training a neural netwoik in a supervised learning mode. Fig. 16 illustrates the strategy for one netwoik module (i.e., probability estimation for a single design property). During the training process, a training record of a past design is selected randomly from the training set. Then, a number of the properties are selected randomly from the record and omitted, creating a partially complete design description which is presented to the netwoik on the input layer. The same record in its entirety is shown to the netwoik at its output, which provides the correct output for each network module. The forward propagation of input signal, error computation, and backward propagation of the error followed by updating of the weights in the whole network takes place as in any application of the back-propagation learning algorithm.

The random omission of a part of the design record was experimented with to determine the optimal strategy of the training record presentation. Although it was definitely determined that the netwoik performed much better with this approach to training then when only the complete records are used for training, no conclusive results could be obtained as to which strategy of partial presentation is optimal: many different strategies (e.g., leaving different number of properties from the netwoik, gradually increasing number of properties presented or omitted from the record) yielded the same quality of results. We will address this issue in more detail in future research.

The learning algorithm was the standard Generalized Delta (i.e., gradient descent) algorithm. The learning parameters that were adjusted for the learning experiments are the step coefficient and the momentum term. It was found that the step coefficient affected the quality of the solution if it was set to a relatively high value. The values that we used were in the range [0.01,0.005], The momentum term was kept at 0.4.

The criterion for successful learning is not as obvious for NETSYN as it is for the usual pattern recognition tasks. For example, it makes no sense to use the Sum of Squared Errors criterion over output unit activations for the reason that the individual errors will always exist due to the different target outputs for the same netwoik input. An error criterion that was employed in NETSYN learning was based on the

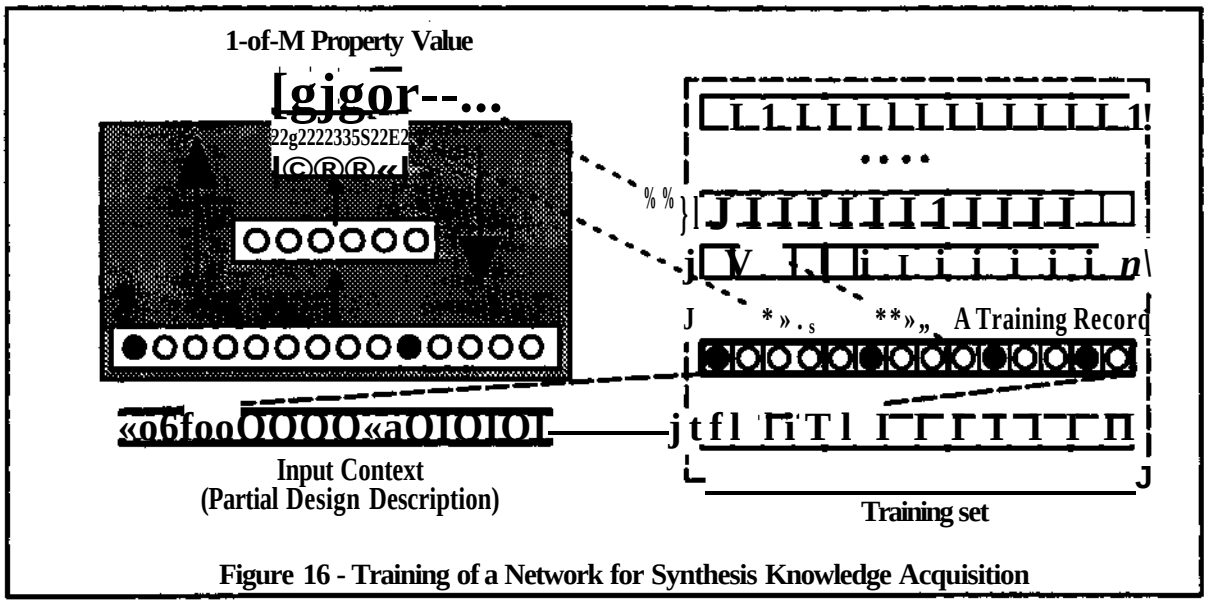


Figure 16 - Training of a Network for Synthesis Knowledge Acquisition

sum of the activations of output units. Ideally, the sum of the output unit activations would converge to unity as the netwoik approaches a correct estimate of the probabilities. However, because of the difficult nature of the learning task, the sum of these outputs would oscillate around unity and never converge. When a steady pattern of these oscillations have occurred, the training would be stopped and the netwoik tested.

Chapter 5

Evaluation of NETSYN

This chapter first describes the test methodology used for evaluating the performance of NETSYN and its comparison to two symbolic inductive learning approaches: ILLS and ECOBWEB. Then, the following are presented:

- The performance of NETSYN on two sets of tests. In the first test, a very large number of training cases which covered almost the whole synthesis space were used during training, while a relatively small number of cases were kept aside for testing purposes. The second test then investigated the performance of NETSYN on a series of learning tasks having relatively different numbers of training cases. The number of training cases and the portion of the synthesis space covered by these cases were considerably smaller than in the first test.
- The comparison of NETSYN and the ILLS symbolic learning system on a learning task with a large number of training cases covering nearly the whole synthesis space.
- The comparison of NETSYN and the ECOBWEB symbolic learning system on a series of learning tasks using several sizes of training sets. The number of training cases and the portion of synthesis space covered by these cases were relatively small with respect to the complete space.

5.1 TEST METHODOLOGY - AN ARTIFICIAL DESIGN PROBLEM (ADP)

The intent behind creating an artificial synthesis problem was: (1) to create a task that emulates the complexity of a synthesis knowledge learning task in a real application; (2) to clearly demonstrate computational capabilities of the proposed neural netwoik approach on this learning task; and (3) to allow clear measurement of the performance of the network and a clear statement about the capability of the proposed approach.

5.1.1 Synthesis Space

The synthesis space of the problem is defined by eight design properties: A, B, C, D, E, F, G, and H. Each of these properties may take on one out of five corresponding property values. For example, property A may take on a value from the collection {ai, a2, a3,34, as}, and property H may be assigned a value from the collection {hi, I12,113,114,115}. One may think about these properties in terms of the properties found in Ml as discussed in 3.3 (e.g., RAM size, RAM chip, board cost, board speed).

For this problem, we treat the initial four properties (A, B, C, D) as design specifications and the latter four (E, F, G, H) as design descriptions to be determined by the synthesis process. Moreover, we have reduced the dimensionality of the search space such that only the design specifications span their complete domains to produce valid design specifications. Thereafter, the design descriptions are assigned values depending on the design specifications and, for some, on one or more design descriptions.

5.1.2 Synthesis Knowledge

The synthesis knowledge which defines the valid relationships between design properties are given next. This knowledge is presented in the form of rules. The general forms of these rules are:

if A G { a_x } A B G { b_y } A C G { c_k } => E_m

if A G { a, } A B G { b_j } A C G { c_k } A £ G { e, } => F_m

if C G { c_i } A D G { d,) A E G (e , } A F G { /* } => G_m

if C G { c, } A £> G { d, } A F G { /* } => H_m

where each rule gives the range of those design properties that govern the value assignment for each design description.

Tables 1,2, 3, and 4 give the complete set of rules that describe relationships that may hold at any time between design property values of a feasibly synthesized alternative. The rules are exhaustive: for any combination of design specification values, there is at least one rule that applies for determining each design description value. However, the rules are not exclusive: for the same design specification, more than one rule may be applied to give the value of a design description. The later characteristic of this set of rules becomes obvious when the rules are expressed in terms of the specifications only.

Each row in the table represents one synthesis rule applicable for the value assignment of the corresponding design description. The ranges of values of design specifications and, eventually, design descriptions, for which the synthesis rule applies, are given in the columns to the left of the last column. The last column gives the values that the corresponding design description may be assigned if the left-hand side is satisfied. The rules are either deterministic or probabilistic. In the case of probabilistic rule, the assumed frequencies of the assignments of each value are given in parentheses next to that value. One out of a collection of values may be assigned to the design attribute.

5.13 Training and Test Set Generation

The above definitions of the problem solution space and the synthesis knowledge were used to implement a generator of valid designs for the synthesis problem. The purpose of the design generator is to create the designs that are subsequently used for training and testing of the neural network.

The generator was run ten times on a complete set of specification combinations (i.e., each specification taking every value from its domain) giving a total of 6250 valid design cases. This collection of 6250 cases necessarily consisted of repeated design cases as the generator was run with the goal of capturing the probabilistic nature of the underlying design synthesis process. Next, the complete design set was partitioned into five train-test collection pairs: (1) 6014 training cases and 50 test cases; (2) 500 training cases and 500 test cases; (3) 1000 training cases and 1000 test cases; (4) 4000 training cases and 2000 test cases; and (5) 5000 training cases and 1000 test cases. Only in the first partition were the test cases completely unique (i.e., not present in the training set), while in the latter four partitions the test and training cases were selected randomly from the original collection of 6250 cases with the chance of having a number of identical training and test cases.

Two different evaluation methodologies have been employed in these tests. The following two subsections describe each of these evaluation methodologies.

A	B	C	E
1	—	—	1 (50%), 2 (50%)
2	–	1,2,3	2
2	–	4,5	1
3,4	1,2	–	3
3	3	–	3
3,4	4,5	–	5
4	3	–	5
5	–	–	4

Table 1 - Synthesis Rules for Attribute E

A	B	C	E	F
1	—	—	1	1
1	-	-	2	2
2,3	1,2	1,2	1,3	3
2,3	1,2	1,2	2,5	4
2,3	3,4,5	1,2	1,2	3
2,3	3,4,5	1,2	5	4
2,3	-	3,4,5	-	5
4,5	-	1,2	3,4	3 (70%),4 (30%)
4,5	—	1,2	5	5
4,5	-	3,4	5	3 (30%),4 (70%)
4,5	—	3,4	3,4	5
4,5	-	5	-	3 (30%),4 (30%),5 (40%)

Table 2 - Synthesis Rules for Attribute F

C	D	E	F	G
1,2	1	1,2,3	-	1 (50%),2 (50%)
1,2	1	4,5		1 (50%),3 (50%)
1,2	2, 3	-	1,2, 3	1 (50%),3 (50%)
1,2	2, 3	—	4,5	2 (50%),4(50%)
1,2	4, 5	-	1,2, 3	1 (3O%),3 (70%)
1,2	4, 5	—	4,5	2 (30%),4(70%)
3,4,5	1	-	1,2	1 (100%)
3,4,5	1	—	3,4, 5	1 (80%),5 (20%)
3	2, 3,4, 5	—	1,2, 3,4	2 (50%),3 (50%)
3	2, 3,4, 5	-	5	2 (8O%),3 (20%)
4,5	2, 3,4, 5	-	1,2, 3	3 (50%),4 (50%)
4,5	2, 3,4, 5	-	4,5	3 (80%),4 (20%)

Table 3 - Synthesis Rules for Attribute G

C	D	F	H
1	-	1,2	1 (50%), 2(50%)
1	-	3,4,5	3 (50%), 4 (50%)
2,3,4,5	1,2	-	1 (30%), 2 (50%), 5(20%)
2,3,4,5	3,4,5	1, 2, 3, 4	4 (20%), 5 (80%)
2,3,4,5	3,4,5	5	5

Table 4 - Synthesis Rules for Attribute H

5.1.4 Performance Evaluation Methodology 1

For the test runs of NETSYN on the ADP and for the comparison of NETSYN with ILLS, the following performance evaluation methodology was used:

- A collection of identified test cases was used to create the partial design contexts to be used as input: a part of the design description was omitted to simulate an intermediate design stage and corresponding design context

- For selected design descriptions (i.e., properties E,F,G,H), *aposteriori* probabilities for each value for the test design context were estimated and compared to the frequency of appearance of these property values in the same design context in the complete collection of design cases (i.e., original 6250 design cases). The latter relative frequencies of appearances of each property value were considered to be an accurate *a posteriori* probabilities — they implicitly represent the underlying synthesis knowledge.
- Two metrics were employed to measure and compare the performance of NETSYN:
    - "max-correct" - The percentage of correctly predicted maximum probabilities for each design description for all of the test cases in a test run.
    - "set-coverage" - The number of property values that were correctly predicted to have non-zero and zero probabilities (true/false sense) for the whole range of each property. The probability prediction is assumed to be correct in a binary sense when one of the following is true: (1) the network prediction is below a threshold value when the real frequency of appearance of the property value in the design context is zero, or (2) the network prediction is above that threshold value when the real frequency of appearance of the property value in the design context is greater than zero. The zero threshold is assumed to be *5%* throughout this and the remaining experiments.

In terms of the synthesis process, the first metric measures the capability of the network to correctly learn the most frequent assignments of design property values for given design contexts and therefore to direct the search for alternatives to those solutions that have occurred most frequently.

The second metric measures the capability of the netwoik to reliably learn to retrieve all feasible alternatives and, at the same time, it measures its capability to prune the infeasible search directions. All zero values for this measure indicate that the network has learned to correctly retrieve all and only the feasible alternatives for property values. All positive values of this metric indicate that the network has learned to recall all feasible solutions; however, the netwoik would direct the search process in a number of cases in infeasible directions (the term infeasible direction denotes here a value assignment which is not specified in the artificial synthesis problem). The number of infeasible searches is proportional to the absolute value of the metric. Finally, all negative values of the metric indicate that the netwoik has learned to retrieve only the feasible solutions, but some of the feasible solutions were not learned. The number of cases which are not learned is proportional to the absolute value of the metric.

5.1*5 Performance Evaluation Methodology 2

In the test runs for comparison of NETSYN with ECOBWEB on the ADP, the following performance evaluation methodology was used:

- A collection of identified test cases was used to create the partial design contexts: a part of design description was omitted to simulate this design stage described by the design context.
- For selected design descriptions (i.e., properties E, F, G, H), the value with the highest probability of being used in the test design context was predicted and compared to the actual value of the design description that appears in the given test case.

This evaluation methodology gives a relative measure of performance of two methods: only the most probable value for a property was predicted and compared to the property value of the specific test case which supplied the input design context This methodology did not take advantage of the well-defined nature of the problem to assess the accuracy of the learning approaches with respect to:

- the true, most probable value and
- the correctness of the method in refusing the infeasible values and selecting only feasible values for the design context.

Nevertheless, the tests provide a useful comparison of two approaches with respect to their capability to learn a particular kind of synthesis knowledge.

The following subsections report on the performance results of NETSYN and comparison results with ILLS and ECOBWEB.

5*2* NETSYN PERFORMANCE ON **ADP**

To determine the performance of NETSYN on the artificial synthesis problem, two types of tests were run:

- TEST 1: The partition with 6014 training cases and 50 test cases was used to perform three different kinds of test runs.
- TEST 2: All remaining partitions of the original collection of cases (500/500, 1000/1000, 4000/2000,5000/1000) were used to run the hardest of the three test runs (determined to be the case when all 4 specifications are presented as input).

Evaluation method 1 was used in both tests to determine the performance of the netwoik.

52.1 TEST 1 **Results**

The following test runs have been performed in this test:

- 2-of-8 run. Two out of four design specifications were given as a partial design context. All 300 partial design contexts generated from the 50 test cases were used, resulting in 300 estimations.
- 4-of-8 run. All four design specifications were given. All 50 test cases were used for this test run.
- 7-of-8 run. Complete test design cases were used to predict each remaining design description value in turn. All 50 cases were used in this test run.

Figs. 17 and 18 present the results for each of the test runs with respect to the set-coverage and max-correct metrics, respectively. Each test run (2-of-8,4-of-8,7-of-8) was performed for each design description (E, F, G, H) giving a total of twelve test instances.

The performance of the network with respect to the set-coverage metric was centered around the perfect value (i.e., zero) in each of the test instances. In the worst test instance (run 4-of-8, property G), 24% of the estimations were out of the (0, +1) interval, and 100% of estimations were inside (0, +2) interval. In other words, in this worst test instance, 76% of the estimations of probability were perfect or included single incorrect probability estimation (in the binary sense) while in only 24% of the estimations the network had two incorrect probability estimations. In nine out of twelve test instances 80% of estimations were perfect (i.e., estimations with zero set coverage).

The performance of the network with respect to the max-correct metric was above 75% in all estimations, while in most test cases (nine out of twelve), 80% of the estimations were correct.

These results show respectable performance of NETSYN on the ADP. Construction of a probability estimation function is possible for this artificial design problem where we have a representative training set covering a wide range of the synthesis space. The following test illustrates that we can maintain this performance to a large extent when having much smaller training samples.

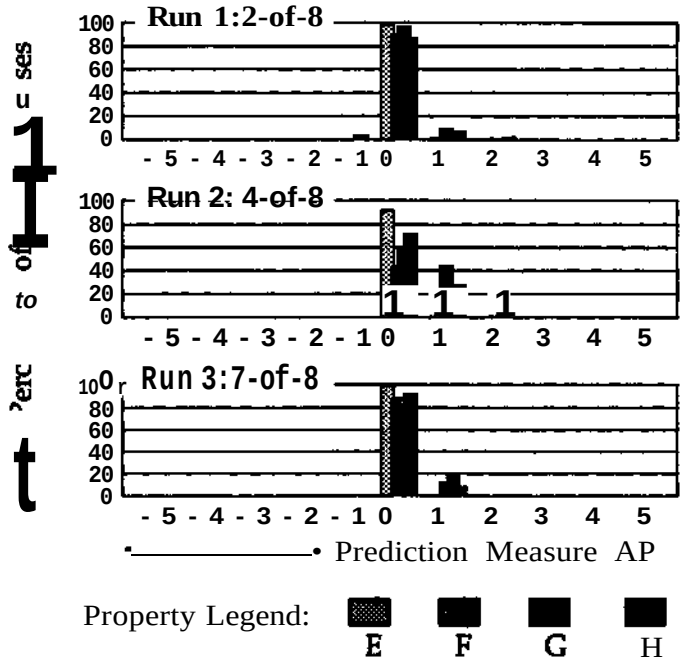


Figure 17 — NETSYN Performance with Respect to "set-coverage" (TEST 1)

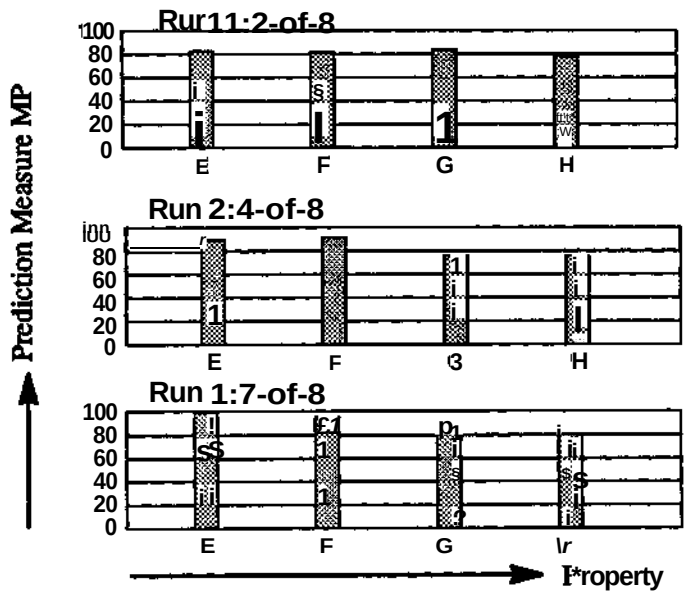


Figure 18 — NETSYN Performance with Respect to "max-correct" (TEST 1)

522 TEST 2 Results

In this second test, the hardest test strategy as indicated by the previous test was used (4-of-8 specifications presented as input). The following test runs have been performed in this test:

- 500/500 run. Network was trained on 500 randomly selected cases and tested on another collection of 500 randomly selected test cases.
- 1000/1000 run. Network was trained on 1000 randomly selected cases and tested on another collection of 1000 randomly selected test cases.
- 4000/2000 run. Network was trained on 4000 randomly selected cases and tested on another collection of 2000 randomly selected test cases.
- 5000/1000 run. Network was trained on 5000 randomly selected cases and tested on another collection of 1000 randomly selected test cases.

Figs. 19 and 20 represent the results for each of the test runs with respect to the max-correct and set-coverage metrics, respectively. Each test run (2-of-8,4-of-8,7-of-8) was performed for each design description (E, F, G, H) giving a total of twelve test instances.

The results represented by these diagrams show that NETSYN could achieve very good performance when significantly smaller training sets were used. The performance with respect to the set-coverage metric, shows that with the training size of 500, the network succeeded in estimating the probabilities in the binary sense so that for 70% of the estimations the result was in the interval (0, +1). Similarly, for the max-comect metric, the performance of the network was above 60% in the worst case when trained on 500 cases. This performance went over 70% for all test instances when the network was trained on 1000 cases. The performance of the network improved with the increase of the training size and with respect to both metrics, but the improvement was relatively minor. This indicates that the approach may be useful in the cases with relatively small number of training cases.

5J COMPARATIVE ANALYSIS: ILLS AND NETSYN

To compare the performance of NETSYN and the ILLS symbolic learning system [Julien, 92] on the artificial design problem (ADP), a single test was run. The partition with 6014 training cases and 50 test cases was used to perform three different kinds of test runs: 2-of-8 run, 4-of-8 run, and 7-of-8 run.

The estimated probabilities of values for a single property (F) are shown in the diagram in Fig. 21. Two shades of the bars represent two approaches applied to the same test: NETSYN and ILLS. Evaluation method 1 was used in the test to determine the performance of both networks.

Apparently, ILLS has difficulty with capturing the relationships between design properties and it performs considerably worse with respect to the set-coverage metric than NETSYN. This is particularly obvious in the hardest test run (i.e., 4-of-8) where the bar diagram is shifted away from the ideal value (i.e., zero) indicating that the performance substantially deteriorated. In [Reich 91], a similar conclusion was made with respect to the performance of methods such as ILLS for synthesis tasks where many-to-many relationships between design properties exist and where multiple solutions exist for the same input specifications. Therefore, this result comes as an experimental confirmation of an earlier finding.

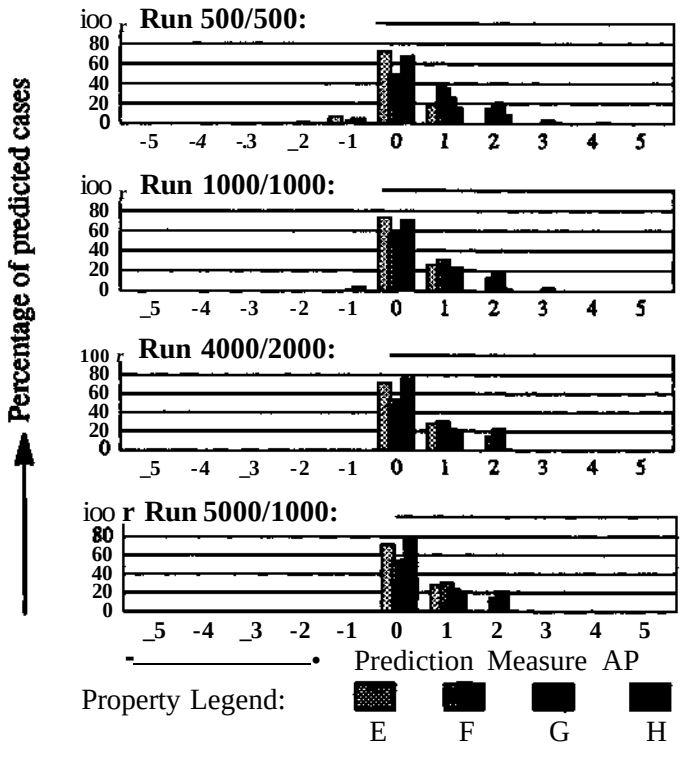


Figure 19 — NETSYN Performance with Respect to "set-coverage" (TEST 2)

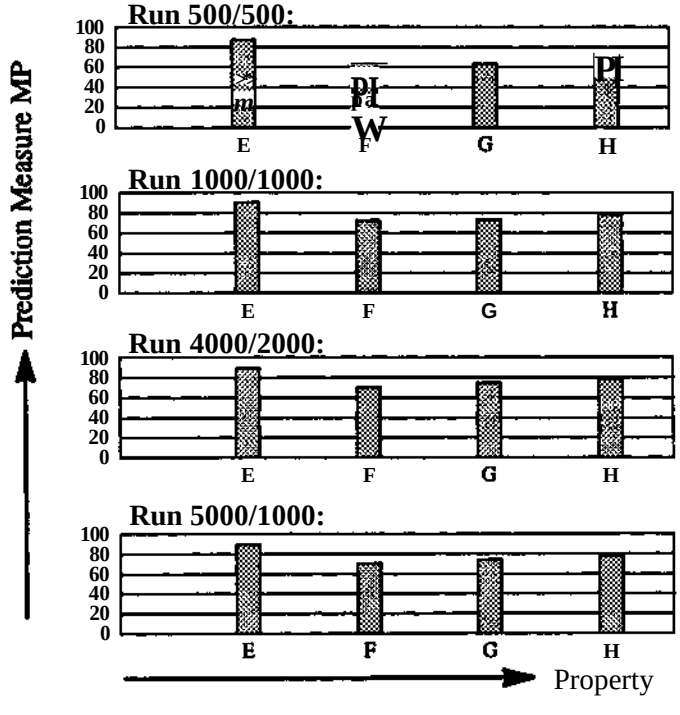


Figure 20 — NETSYN Performance with Respect to "max-correct" (TEST 2)

5.4 COMPARATIVE ANALYSIS: ECOBWEB AND NETSYN

To compare the performance of NETSYN and the ECOBWEB symbolic learning system [Reich 91] on ADP, a single test was performed again using four of the training/test partitions of original collection of 6250 (500/500,1000/1000,4000/2000,5000/1000). Only the 4-of-8 test presentation strategy was implemented to compare the performance of the methods.

Fig. 22 presents the results for each of the test runs. Evaluation method 2 was used in the test to determine the performance of both networks. The results show consistently better performance of NETSYN relative to that of ECOBWEB. The relative increase in performance ranged from 6% to 27%. Due to the use of evaluation methodology 2, it is not possible to definitely say anything about the performance of ECOBWEB on ADP with respect to either max-correct or set-coverage metrics. We plan to address this issue in all future comparisons of NETSYN.

5.5 DISCUSSION

We have constructed an Artificial Design Eroblem (ADP) in order to evaluate the proposed approach and to compare it to other relevant machine learning approaches. The rationale behind building ADP was to impose a synthesis problem in a relatively small synthesis space (i.e., synthesis problem described by relatively small number of design properties) with relationships between design properties of relatively high complexity. These relationships are meant to emulate the complexity of the knowledge that may be found in a real synthesis problem. The complexity was achieved by providing (1) nondeterministic relationships, (2) many-to-many relationships between design properties, (3) relatively involved dependencies between design properties in which a considerable number of properties influence the feasible ranges of other properties.

By testing the machine learning approaches on only a single learning problem (such as ADP), one cannot draw a complete picture about capabilities and relative performances of these methods (NETSYN, ECOBWEB, and ILLS). Nevertheless, indications of capabilities and incapabilities of these tested methods for SKAU are present. We hope that more comparative test problems will be devised to help portray the behavior of these different learning methods.

NETSYN performs well both when a large portion of synthesis space was covered by training cases and when considerably smaller numbers of training cases were used. This result provides an initial indication of the merit of the NETSYN approach with respect to the size of training sample. More experimental results are required for stronger statements to be made concerning performance of NETSYN in situations having limited sample size.

The performance of two symbolic inductive learning methods have been compared to the performance of NETSYN and the following are the basic findings:

- The performance of the ILLS system was shown to be insufficiently accurate for the task involving many-to-many mappings and multiple alternatives for same design specifications.
- The performance of NETSYN was consistently better than the performance of ECOBWEB on a series of learning tasks derived from the ADP (ranging from 7% to 24%).

Therefore, the conclusions that may be drawn from the above evaluations are the following:

- NETSYN appears to be a promising approach to SKAU.
- NETSYN appears to have a more appropriate learning bias than the two symbolic learning approaches to which it was compared for problem types of which ADP is a representative example.
- More experimental work is required to test the approach thoroughly on similar and other synthesis problems for which a larger number of parameters of the learning task will be varied.

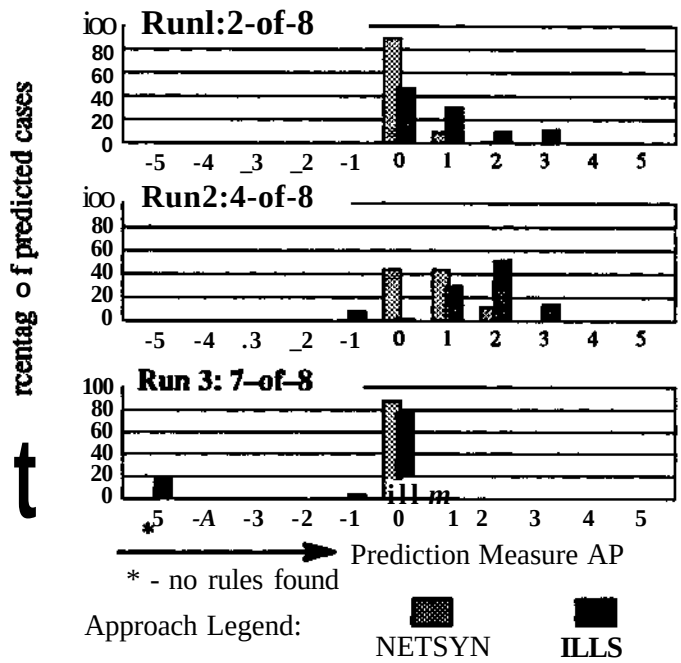


Figure 21 — Comparison of Performance: NETSYN and ILLS

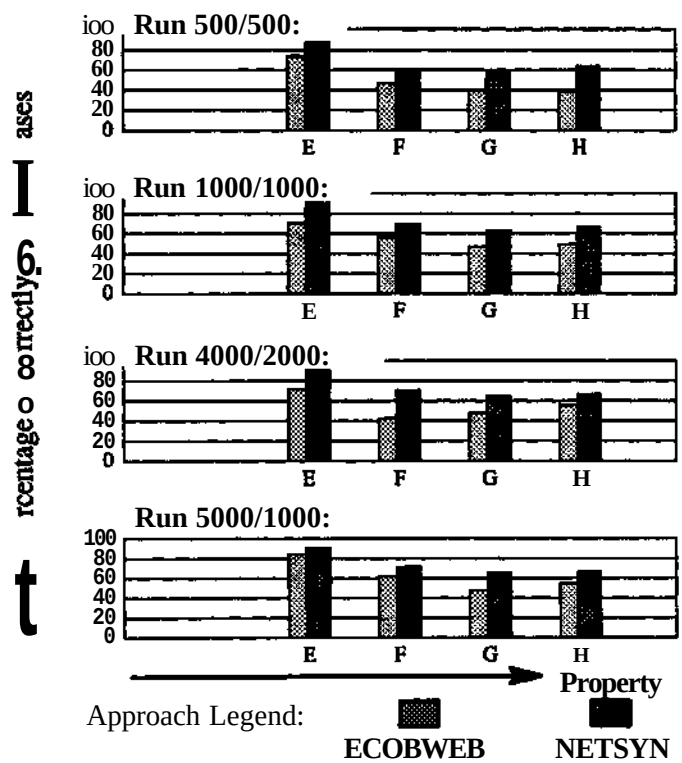


Figure 22 — Comparison of Performance: NETSYN and ECOBWEB

Chapter 6

Future Work

This report gives initial results in applying connectionist learning methods to the generic task of synthesis knowledge acquisition and use (SKAU). The research provides a starting point from which several directions may be followed in extending and refining the proposed approach. The following is a collection of such directions that collectively aim at the goal of bringing this approach closer to realistic design domains and their synthesis tasks:

- Extend NETSYN representational capabilities. Presently, the representational capability of NETSYN is limited to mapping the discretized design property values onto a collection of binary processing elements both at the input and at the output layers of the network. While the approach is based on the discrete binary output representation of property values (i.e., 1-of-M classification problem), the representation of the input may vary widely. Representations of continuous-valued properties may readily be included as input To provide the capability of handling continuous values at the output layer of the network, the separate modules of the network may be modified to estimate useful statistical properties for continuous design property values (e.g., mean and variance). In addition, the schemes for representation of the structured design properties (e.g., hierarchical properties) need to be investigated.

- Investigate approaches for incremental learning. The capability of those connectionist models that allow incremental learning needs to be investigated, since the current connectionist model described in this report does not have this capability. An example of such a model is the Cascade-Correlation Network [Fahlman 90]. The principal question is whether this or any other alternative model that can learn incrementally is capable of estimating probabilities in the sense required for this approach.

- Investigate approaches for scaling up. One of the critical issues for connectionist approaches is scaling up to large problem spaces. Several factors influence this problem, two of which are representation and use of the domain knowledge. By selecting appropriate representations for design, it is possible to considerably reduce the complexity of the computations involved in constructing (learning) the probability estimation function. Similarly, by using any domain knowledge that may exist, one can make informed decisions about the connectivity between different parts of the input layer and different modules of the network (each module estimates probabilities for single property). These and other approaches will be investigated as the evaluation and the use of NETSYN is taken to larger design spaces.

- Apply NETSYN to realistic synthesis tasks. Currently, NETSYN is being tested on a subdomain of Ml [Gupta 91]. This subdomain is design of CGA controllers (described in 3.3). The initial experiments bring an initial flavor of a real synthesis task to NETSYN. New requirements, that have not been identified earlier in our research, are shaping the NETSYN approach further. The future work will address the extensibility, applicability, and quality of the NETSYN approach to acquiring synthesis knowledge in this and other more complex subdomains of Ml. Additionally, further comparative studies on performance of NETSYN and other approaches for SKAU are planned.

Chapter 7

Summary

This document presents results of research that has investigated connectionist learning approaches for synthesis knowledge acquisition and use (SKAU).

Synthesis is a process of mapping from the specification space to the form space. The goals for machine learning in support of synthesis is to acquire the relationships between form, function, and behavior properties that satisfy specified design requirements and, thereafter, to facilitate a more direct mapping from the specification space to the desired location within form and behavior spaces. The basic requirements for a system capable of acquiring synthesis knowledge are 1) the ability to capture many-to-many relationships between design properties, 2) the ability to produce multiple solutions satisfying the same design specifications, and 3) the ability to generalize from a finite number of representative design cases.

The nature of the synthesis process allows a probabilistic approach. The probabilistic approach followed in this research is founded upon the idea of using a probability estimation function for SKAU. The goal of the probability estimation function is to estimate the probability of each possible value of each design property being used in a given design context The minimal requirements for such a probability estimation function to be useful for synthesis are: 1) estimate non-zero probabilities for feasible design property values; 2) to estimate zero probabilities for infeasible design property values; and 3) to correctly rank the most probable design property value.

The NETSYN connectionist approach to SKAU constructs a probability estimation function by using a connectionist learning system to acquire and represent the probability estimation function. Theoretical results provide an interpretation of the NETSYN approach in terms of estimating Bayesian *a posteriori* probabilities. NETSYN presently follows the uniform convergence learning approach and it may be extended to the Bayesian learning approach for the treatment of synthesis learning tasks that have access to only small training samples.

Performance of a learning system on a well defined learning task is one of the necessary ingredients for evaluation of learning systems for SKAU. The artificial design problem (ADP) presented in the report is one such well defined learning task that emulates a real synthesis learning task. Based on the proposed evaluation methodology for performance of learning systems on ADP, the NETSYN approach is promising with respect to both absolute performance and relative performance to two symbolic learning approaches (DLLS and ECOBWEB) on the ADP.

The goals of the future work are the application, evaluation, and adaptation of the NETSYN approach for real design domains. Design of single-board computer systems (MICON) is the domain in which NETSYN is being tested presently. This domain clearly meets the necessary requirements of a realistic design domain for evaluation of a learning system.

REFERENCES

[Ang 75] Ang H-S. A. and W. H. Tang, *Probability Concepts in Engineering Planning and Design*, John Wiley & Sons Inc, 1975.

[Barto 90] Barto A. G., "Connectionist Learning for Control: An Overview", Technical Report COINS 89-89, September 1989, Department of Computer and Information Science, University of Massachusetts Amherst.

[Buntine 91] Buntine W. L. and A. S. Weigend, "Bayesian Back-Propagation", Complex Systems, Vol. 5,1991, pp. 603-43.

[Coyne 90] Coyne R. D. et al, *Knowledge-Based Design Systems*, Reading Mass, Addison-Wesley, 1990.

[Cybenko 88] Cybenko, G., "Continuous Valued Neural Networks with Two Hidden Layers Are Sufficient", Technical Report, Department of Computer Science, Tufts University, Medford, MA.

[El-Jaroudi 90] El-Jaroudi, A. and J. Makhoul, "A New Error Criterion For Posterior Probability Estimation With Neural Nets", IJCNN International Joint Conference on Neural Netwoiks, San Diego, 1990, pp 185-92.

[Fahlman 90] Fahlman, S. E. and C. Lebiere, "The Cascade-Correlation Learning Architecture", in D. S. Touretzky (Ed), Advances in Neural Information Processing Systems II, Denver 1989, San Mateo: Moigan Kaufmann, pp. 524-32.

[Flemming 92] Flemming, U., J. Adams, C. Carlson, R. Coyne, S. Fenves, S. Finger, R. Ganeshan, J. Gamett, A. Gupta, Y. Reich, S. Siewiorek, R. Sturges, D. Thomas, R. Woodbury, "Computational Models for Form-Function Synthesis in Engineering Design," EDRC Technical Report No. 48-25-92, Engineering Design Research Center, Carnegie Mellon University, March 1992.

[Gupta 91] Gupta A. P., W. P. Birmingham, and D. P. Siewiorek, "Automating the Design of Computer Systems", Technical Report CSE-TR-104-91, Department of Electrical Engineering and Computer Science, University of Michigan.

[Hertz 91] Hertz J. A., R. G. Palmer, and A. S. Krogh, *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.

[Hinton 86] Hinton G. E., J. L. McClelland, and D. E. Rumellhart, "Distributed Representations", in D. E. Rumelhart and J. L. McClelland (Eds), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1., MIT Press, Cambridge, Massachusets Press, 1986.

[Hornik 89] Homik K., M. Stinchcombe, and H. White "MultilayerFeedforward Networks are Universal Approximators", Neural Networks 2, 1989, pp. 359-66.

[Julien 92] Julien, B. "Experience with Four-Based Induction Methods", AI Applications, Vol. 6, No. 2, 1992, pp. 51-56.

[Lapedes 88] Lapedes, A. and R. Farber, "How Neural Nets Work", in D. Z. Anderson (Ed), Neural Information Processing Systems, Denver 1987, New York: American Institute of Physics, pp. 442-56.

[Le Cun 89] Le Cun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition", Neural Computation 1, 1989, pp. 541-51.

[Lu 92] Lu S. C-Y., *Knowledge-Based Engineering Systems Research Laboratory Annual Report,* Department of Mechanical and Industrial Engineering, Colege of Engineering, University of Illinois at Urbana-Champaign, 1992.

[Pomerleau 89] Pomerleau D. A.," ALVINN: An Autonomous Land Vehicle in a Neural Network", in D. S. Touretzky (Ed), Advances in Neural Informations Processing Systems I, Denver 1988, San Mateo: Morgan Kaufinann, pp 305-13.

[Qian 88] Qian, N and T. J. Sejnowski, "Predicting the Secondary Structure of Globular Proteins Using Neural Network Models", Journal of Molecular Biology 202,1988, pp. 865-84.

[Reich 91] Reich, Y., "Building and Improving Design Systems: A Machine Learning Approach", Technical Report EDRC 02-16-91, Engineering Design Research Center, Carnegie Mellon University.

[Richard 91] Richard, M. D. and R. P. Lippmann, "Neural Netwoik Classifiers Estimate Bayesian *aposteriori* Probabilities", Neural Computation, Vol. 3, No 4, Winter 1991, pp. 461-83.

[Rumelhart 86] Rumelhart D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation", in D. E. Rumelhart and J. L. McClelland (Eds), *Parallel Distributed Processing: Explorations in the Microstructures ofCognition,* Vol. 1., MIT Press, Cambridge, Massachusets Press, 1986.

[Sejnowski 87] Sejnowski T. J., and C. R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text", Complex Systems 1,1987, pp. 145-68.

[Simon 81] Simon H. E., *The Sciences of the Artificial,* MIT Press, Cambridge, Mass, 1981.

[Tesauro 90] Tesauro G., "Neurogammon Wins Computer Olympiad", Neural Computation 1,1990, pp 321-23.