Computer Assisted Reasoning On Principles And Properties of Medical Physiology*

B. Zupan[1], J. A. Halter [2], M. Bohanec[1]

[1] J. Stefan Institute, Dept. of Intelligent Systems, Ljubljana, Slovenia

[2] Baylor College of Medicine, Div. of Restorative Neurology and Human Neurobiology, Houston, TX, USA

Abstract

Developing practical tools to aid in understanding medical physiological systems is a formidable undertaking. This paper presents a method that uses a property structure for the domain being investigated. Furthermore, it employs realistic models to present examples of the behavior of the system. From these examples the principles that relate the properties are inferred through the use of genetic algorithm-based machine learning. The principles are expressed as qualitative rules that derive the values of the properties. The structured approach and the qualitative representation of principles provide a simplified means to reason about the roles of properties and meaning of principles of the physiological systems being investigated.

1 Introduction

Developing practical tools to aid in understanding medical physiological systems is a formidable undertaking. Recent advances in computational technology allow the use of complex realistic models to assist in the discovery of principles and properties within different physiological systems. Such models map a set of input properties (input parameters of a model) to a set of output properties (description of model behavior). However, as these models grow in sophistication, it becomes more difficult to manipulate the growing number of model parameters and equations efficiently in order to reason about the roles of the properties and hypothesize about the principles of the modeled system. Moreover, the simulation of realistic models usually requires substantial computational resources. The reasoning about principles which is then based only on model simulation can be a potentially time-consuming and tiresome process.

The principles of the modeled system determine *how* the input properties are mapped to output properties. In order to understand the process that is being modeled, it should be feasible to derive *why* such mapping occurs and to be able to derive and understand the principles that guide the mapping.

The realistic model itself is usually based on several known mechanisms that can be viewed as atomic principles. The reason that such models were built is not only to reproduce the behavior of the realistic system, but moreover to discover how these mechanisms interact, how they are influenced by a selection of input properties, and what the relationship is between properties. For example, a realistic computational neuromodel [13, 9] may model the properties of sodium channels, but a sole inspection of the model equations does not reveal how sodium channel activation

*Support for this effort was generously provided by the Ministry of Science and Technology of the Republic of Slovenia (BZ and MB), the W.M. Keck Center for Computational Biology, the Whitaker Foundation (JAH), and Developmental Support from Information Technology Program at Baylor College of Medicine (JAH and BZ).

voltage changes the sodium ion concentration dynamics, and more, how this dynamics influences the conduction of the sodium ion action potential.

This example presents a need for properties intermediate to input and output properties and a need for the relational structure that would outline the possible dependencies among the properties. By intermediate properties we are referring to those that are not explicitly present within the model, but are considered to be useful when reasoning about model behavior. One might assume the existence of intermediate properties from medical science or can solely hypothesize it.

In this paper we propose a fuzzy rule-based framework that uses a hierarchical dependency of properties and encodes the principles that relate the properties through fuzzy rules. We show that these principles can be automatically extracted by learning the rules from examples of model behavior. The hierarchical property structure can then be used for either predicting the value of output properties from a given set of input properties or for the analysis of principles and property dependencies.

The paper is organized as follows. The next section presents several frameworks that use a structured approach to reasoning and data analysis. The method proposed in this paper is then outlined in Sec. 3. An example of using this method to derive the principles for the model from the domain of computational neuroscience is given in Sec. 4. Sec. 5 concludes the paper and outlines the future work.

2 Related Work

An approach presented in this paper is based on a qualitative multi-attribute decision making method DECMAK [3]. DECMAK uses decomposition of a decision problem into smaller, less complex problems. The representational structure of the problem is a tree of attributes. Attributes can take a finite set of values, each value being a distinct descriptor for that attribute. The tree hierarchically orders the attributes from inputs (tree leaves) to a single output (the root of the tree). DECMAK uses a utility functions in a form of rules that define the aggregation of lower level attributes into corresponding higher level attribute. DECMAK assumes the structure and the utility functions to be given by an expert. Both are then used to evaluate the options which are given as a set of input attributes. The overall utility of the option (output attribute) is then derived in a bottom-up fashion from the input attributes. DECMAK provides an extensive support for option evaluation, comparison and ranking. Its successor DEX [4] has been used in over 50 complex decision-making problems.

A similar tree-based hierarchical approach called "signature table schema" is presented in [1]. The problem representation is again a tree of attributes, this time restricting the attributes to two-valued (boolean) domains, i.e., each attribute has only two descriptors, "0" and "1". An aggregation of lower-level to higher-level attributes is done through the use of signature tables, which are equivalent to a rule representation of DECMAK, where a single table entry is represented as a single rule. In [1, 2] signature table schema was used to represent a two-valued function. In contrast with DECMAK, the signature tables where not elicited from expert but rather derived from the example inputs and outputs of a function the system has to represent. It has been shown [1], though, that the learning algorithm has an exponential complexity with respect to the number of function's attributes being modeled and that the chances of finding an efficient heuristic approach are minor.

Both DECMAK and signature table schema deal with qualitative attributes only. Recently, and to a large extend in this decade, methods that use fuzzy logic have been used for approximation and reasoning in domains with quantitative data. For example, [15] describes a hybrid approach that combines both hierarchical structure and fuzzy reasoning. The structure is represented with the directed knowledge graph which picks out the influences (dependencies) of the attributes (properties). Again, the input attributes appear as leaves and the output attributes as roots of the knowledge graph. Similarly, as in DECMAK, attributes can be fuzzy variables and can be described by descriptors and corresponding degrees of belief. Different from DECMAK and signature table schema, the influences of attributes are not encoded as rules but rather as neural networks. Thus, each non-input attribute requires a single neural network, where descriptors of that attribute are output nodes of the network, and descriptors of the lower-level attributes are input nodes. Each neural network has also a hidden layer which is constructed automatically. The weights of the connections in neural network are derived from a set of examples of input and output attributes using a punishment and reward algorithm.

The approach in [15] can derive output attributes from a set of input attributes that take either qualitative or quantitative values. In later case, a system would perform a fuzzification that would assign degrees of beliefs to every descriptor of input attributes.

Common to all of the methods mentioned above is the attempt not only to approximate the decision process or some function being modeled, but to reveal its hierarchical components and their interrelation. The major differences between DECMAK and signature tables schema and the approach from [15] is though in complexity and expressional power of utility functions. We might observe that the approach that uses neural networks might be more accurate in predicting the value of output attribute, while within DECMAK and signature tables it might be easier to explain why such a value was derived. This accuracy – simplicity tradeoff and the fact that our interest is more in the transparency of the derivation process than in its accuracy influenced the basic structure and the derivation methods of the framework presented to be derived from the ones used in DECMAK.

As for the signature table schema, we base our approach on learning the rules from examples of input and corresponding output properties. The method employed uses a genetic algorithm approach. Similar methods have been used to solve several different optimization problems for fuzzy rule-based systems, ranging from rule derivation and selection to improve accuracy of classifier [11], to derivation of fuzzy membership functions [12] and a combination of these two approaches [5].

3 Method

This section outlines the proposed method. The method is based on a property structure that hierarchically orders the properties and depicts their possible dependencies. The values of properties are derived from the properties they directly depend on. The derivation technique uses fuzzy rules, which can be either explicitly given or derived through the use of machine learning. The later assumes to be given a set of examples with input properties and corresponding output properties.

Property structure, rules, and derivation of property values

The input, intermediate, and output properties and their dependencies constitute a hierarchical property structure which is directed and required to be acyclic. Within the property structure, properties are represented as nodes: input properties as leaves, intermediate as internal nodes, and output properties as root nodes of the structure. Vertices in the property structure indicate property dependence. An example of property structure is given in Fig. 3.

Property descriptions and aggregation functions

Each property has a corresponding set of descriptors. The value of the property is fuzzy and is given by degrees of beliefs for each of its descriptors.

Formally, if x_i is a property, $\{v_{i1}, v_{i2}, \ldots\}$ is a set of its descriptors and $d(v_{ij})$ is a degree of belief for j-th descriptor of property x_i, then the value of x_i can be written as $(v_{i1}/d(v_{i1}), v_{i2}/d(v_{i2}), \ldots)$.

To derive a value for each intermediate and output property x_i, an aggregate function is used. As in DECMAK [3], an aggregate function is given as a set of rules that map the values of the properties x_ij that directly influence x_i to a value of x_i, i.e.,

$$f_i : x_i = f_i(x_{i1}, x_{i2}, \ldots) \tag{1}$$

A valid rule would thus be

$$r_i : (v_{i1}, v_{i2}, \ldots) \rightarrow v_i \tag{2}$$

where $v_{i1}, v_{i2}, \ldots$ are descriptors of the properties x_{ij} and v_i is a descriptor of x_i.

During derivation of a property value, a degree of belief for every rule associated with that property is determined as

$$d(r_i) = \min(d(v_{i1}), d(v_{i2}), \ldots) \tag{3}$$

Let R_{v_i} be a set of rules that map the set of parameter descriptions to a descriptor v_i, i.e.,

$$R_{v_i} = \{r_i, r_i : (v_{i1}, v_{i2}, \ldots) \rightarrow v_i\} \tag{4}$$

The degree of belief for v_i is computed as:

$$d(v_i) = \frac{\Pi_{r \in R_{v_i}} d(r)}{\sum_i d(v_i)} \tag{5}$$

Normalization is required in order to keep the degrees of belief in the range $[0, 1]$.

Derivation of the value of output properties from the input properties values is then a bottom-up process that derives the intermediate properties that are directly dependent on input properties first, and derives output properties last.

In order to deal with possible quantitative values of input and output properties, for each of these a corresponding set of membership functions are given. An example membership functions that map a quantitative value $q(x_i)$ of x_i to degrees of beliefs of descriptors v_{i1}, v_{i2}, and v_{i3} is given in Fig. 1.

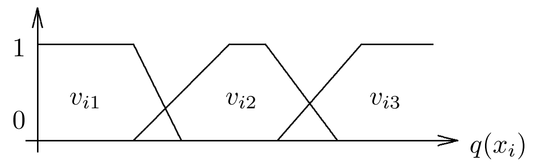

Figure 1: An example membership functions for property x_i and its descriptors v_{i1}, v_{i2}, and v_{i3}.

Fuzzification (assignment of degrees of belief to descriptors from a quantitative value) of input properties takes place prior to derivation of intermediate and output properties values. Once a qualitative values of output properties with corresponding degrees of beliefs are derived, a defuzzification can convert this value to a quantitative one.

Learning rules and membership functions from examples

The entities that are required for the proposed framework to be functional are:

1. a property structure,
2. a set of descriptors of each property,
3. a set of rules to derive a value for every intermediate and output property,
4. a membership function for every descriptor of input and/or output properties that require quantitative interpretation.

While (1) and (2) are required to be given a-priori, (3) and (4) can be specified only partially or even not given at all. The idea is that one can present a set of examples E of the form

$$E = \{e_i; e_i = (iv_{i1}, iv_{i2}, \ldots, ov_{i1}, ov_{i2}, \ldots)\} \tag{6}$$

where iv_{ij} are values of the input properties and ov_{ij} are the expected values of output properties. In the case of realistic models, these examples are obtained through simulation. The system is then required to learn a complete representation of (3) and (4), so as to minimize the error when trying to predict the output properties from input properties of examples E. If, for every example e_i, the system derives an estimated value of output properties $(ev_{i1}, ev_{i2}, \ldots, ev_{in})$, the cumulative relative error over all the examples can be derived in one of the following ways:

$$e_{rel} = \sum_j \sum_i \frac{|ev_{ij} - ov_{ij}|}{ov_{ij}} \tag{7}$$

$$e_{abs} = \sum_j \sum_i |ev_{ij} - ov_{ij}| \tag{8}$$

$$e_{norm} = \sum_j \sum_i \left(\left| \frac{\mathrm{MAX}(ev_{ij}, ov_{ij})}{\mathrm{MIN}(ev_{ij}, ov_{ij})} \right| - 1 \right) \tag{9}$$

The decision which error function to use might be of ultimate importance to the performance of genetic optimization. For the example provided in this paper we have chosen e_{norm} for the reasons discussed in Sec. 4.

The learning method employed is based on genetic algorithm. The rules and membership functions are encoded in integer-typed genes. A population of genes, initially set to arbitrary values, then undergoes a standard genetic algorithm procedure [7] which includes iterative selection, mutation, and crossover of genes. The fitness of each gene is associated with the error estimate as above, i.e., the smaller the error the bigger the fitness. The learning terminates when the gene with fitness that is estimated to be close to optimal is found.

The probabilities of crossover and mutation for a rules' part and membership functions' part of gene are set separately. While the mutation and crossover for a part of the gene that represents the rules are standard, the corresponding ones for membership functions are not and had to be designed to accommodate the constraints the membership functions have to comply with. For example, we assume descriptors of properties to be ordered and allow only a specific type of such functions (see Fig. 2).

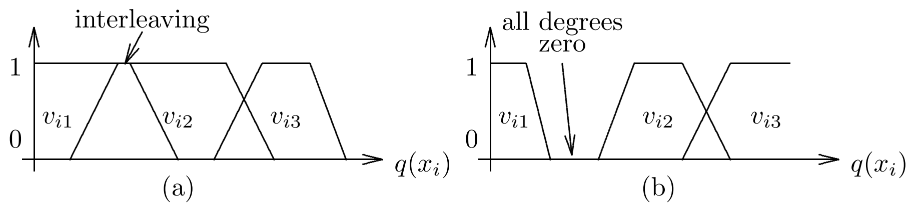

Figure 2: The membership functions are constrainted. For example, a membership functions (a) are not allowed since the upper segments of two functions should not interleave. Also, there should be no quantitative value of property that has all the degrees of beliefs of descriptors 0. This makes memberships functions at (b) illegal.

Utilization

The simplest use of the property structure with rules and fuzzy mappings is prediction. Given a set of input properties' values, these are mapped to qualitative descriptions and then propagated through the property structure up to the output properties. Such predictions can be orders of magnitude faster than the ones that require simulation. But what really differentiates the property structure approach from realistic models is the means of its exploitation. Here we list some possible ways a property structure with its rules can be used:

- The discovery of principles: the principles that relate the properties can be revealed by the inspection of qualitative rules. The rules can be either viewed textually or graphically, or can be transformed to a decision tree. All of these and other rule inspection methods have been introduced in DEX and DECMAK [4, 3].
- Testing and discovery of the constraints that may exist between input and output parameters (hypothesis testing). The constraints that we are testing for are simple monotonicity and local minima/maxima constraints of types defined in [6]. The structured approach reduces the complexity of constraint testing. For example, when a constraint between input property x_i and output property x_o is tested, it is only necessary to inspect the rules that apply for properties that are on the path from x_i to x_o (see Fig. 3).
- Explanation of the derivation of output properties for a given set of input properties through the use of intermediate properties and the rules that were used to derive them.
- The estimation of influence of the parameters to a certain output parameter using the known methods that derive informativity of parameters from rules.
- Analysis of the differences of two (or more) input parameter sets that resulted in different output sets.

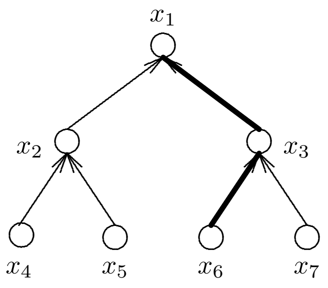

Figure 3: An example property graph for x_1 as output property, x_2 and x_3 intermediate properties and $x_4 \ldots x_7$ input properties. The figure also shows that to test for the relationship between input property x_6 and output property x_1, it is only required to inspect the principles (rules) that derive the values of x_3 and x_1.

Implementation

The system described in this Section was implemented in a C programming language under HP and SGI UNIX workstations. The system uses a PGApack genetic algorithm library [14] and employs Geomview 3D visualization tool [16] for graphical display of rules.

4 Example

This section is to illustrate the use of the proposed method and its capability to derive the principles that define the relationships among properties. In this example, a realistic nerve fiber model is used. The sophisticated distributed parameter model was developed at Baylor College of Medicine in Houston and is used to predict the functional implications of neuronal structural and biophysical properties [10, 8]. The model is given in the form of a system of multiple cross-coupled parabolic partial differential equations that are solved by an implicit numerical integration method.

The problem addressed in this paper is to observe the influences of sodium ionic channel properties (voltage activation `av` and inactivation `iv`) and the influence of sodium permeability (naperm) to the peak sodium current (ina) in a node segment of the neuron. Fig. 4 shows the property structure used.

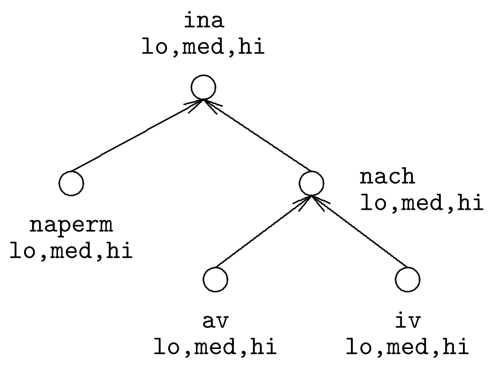

Figure 4: Property structure for the peak sodium current (output property) as a function of sodium permeability, inactivation and inactivation voltage (input properties).

Let us assume that ina monotonically increases with naperm. Let us also define an intermediate property nach that, when increases, should increase the value of ina. nach is the property of the sodium channel that is influenced by iv and av and identifies a level of channel conductance for sodium current. We can state this assumptions for nach and naperm by the following rules:

naperm	nach	ina
lo	lo	lo
med	med	med
hi	hi	hi
lo	hi	med
hi	lo	med

The problem is now to discover how iv and av influence nach and ina. To derive this principles, 100 examples of the form (naperm, iv, av) where presented to the system. Input properties were expressed as the offsets to corresponding parameters of normal mammalian mielinated nerve fiber (for the values and units see [9]). For each example they where selected arbitrary from ranges [0.5,1.5] for naperm, [-10,10] for av, and [-20,20] for iv. The experimental results show that there is high non-linearity because of two different states the neuron might be: non-firing and firing. High negative values of ina appear when the neuron fired and the values close to 0 are in the cases when neuron did not fire.

Next, the system was requested to learn the remaining rules (the ones that are still missing for derivation of ina and all of the rules for nach). For learning, the information about rules and membership functions was coded into genes of 53 alleles. The population size was 300 genes. The crossover probability was 0.85, and probability for mutation was 0.9% and 2.0% for rule and membership function part of gene respectively. Error function used was e_{norm} (Eq. 9). The problem with other two error functions where that the use of the e_{abs} (Eq.8) lead to a poor performance for examples with low values of ina. The e_{rel} (Eq.7) lead genetic algorithm to a local minima with estimated ina=0 (the error equal 1.0) for early generations. The algorithm could then not escape this local minima at later generations and would not find a global optima.

The learning took 1000 iteration steps which accounted for app. 5 min of CPU time on IRIS SGI 4400 Workstation. Membership functions derived for input and output properties are shown in Fig.5. The additional rules for ina, i.e., other than those preset above, are:

naperm	nach	ina
med	lo	lo
lo	med	med
hi	med	med
med	hi	hi

The rules for `nach` are graphically presented in Fig. 6. To no surprise, the graph reveals that `nach` increases with `av` and decreases with `iv`. Because of the monotonical dependence of `ina` from `nach`, `ina` increases with `av` and decreases with `iv`. This discovered principles qualitatively match with a known physiological effect an activation and inactivation voltage on a peak sodium current [8].

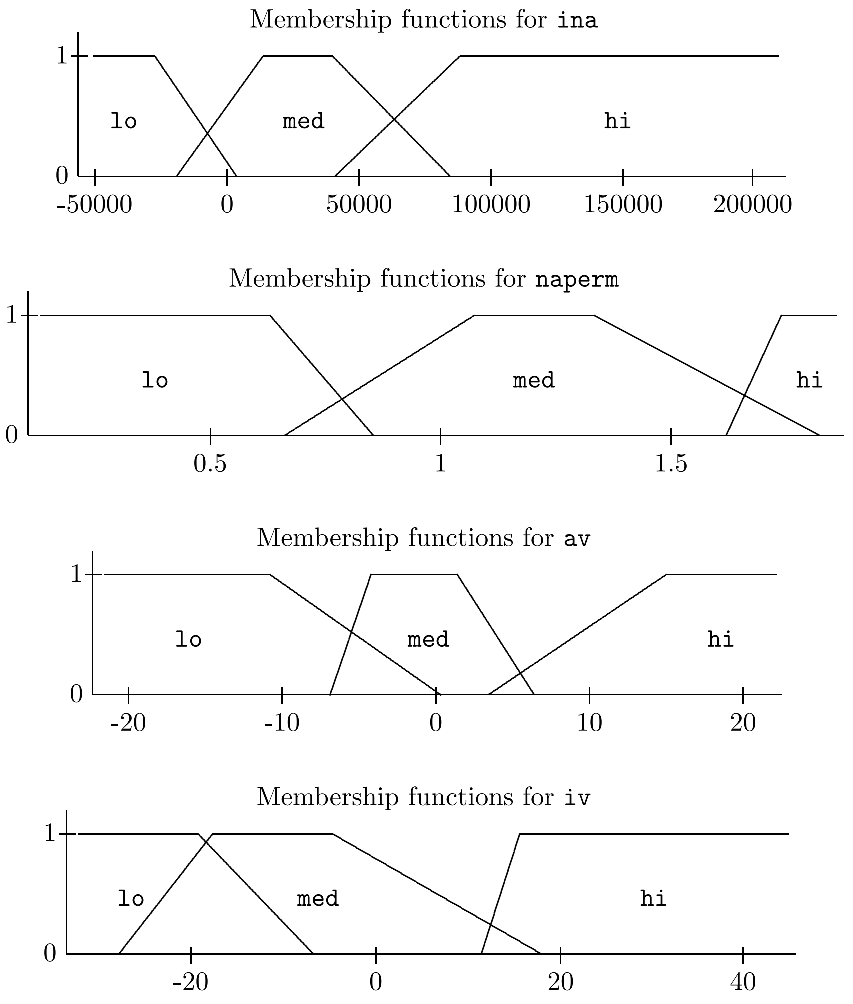

Figure 5: Membership functions as derived from the set of examples by the learning algorithm.

The genetic algorithm derived a solution (rules and membership functions) that minimizes e_{norm}. After 1000 generations the value settled at 17.63. This error is relatively high and can be contributed to the simplicity and low number of descriptors of our system. We performed an experiment where every property had 5 descriptors instead of 3. The error of the solution found was 0.01. But, most importantly, the principles found expressed the same influence of activation and inactivation voltages to peak sodium current as for presented, less complex system.

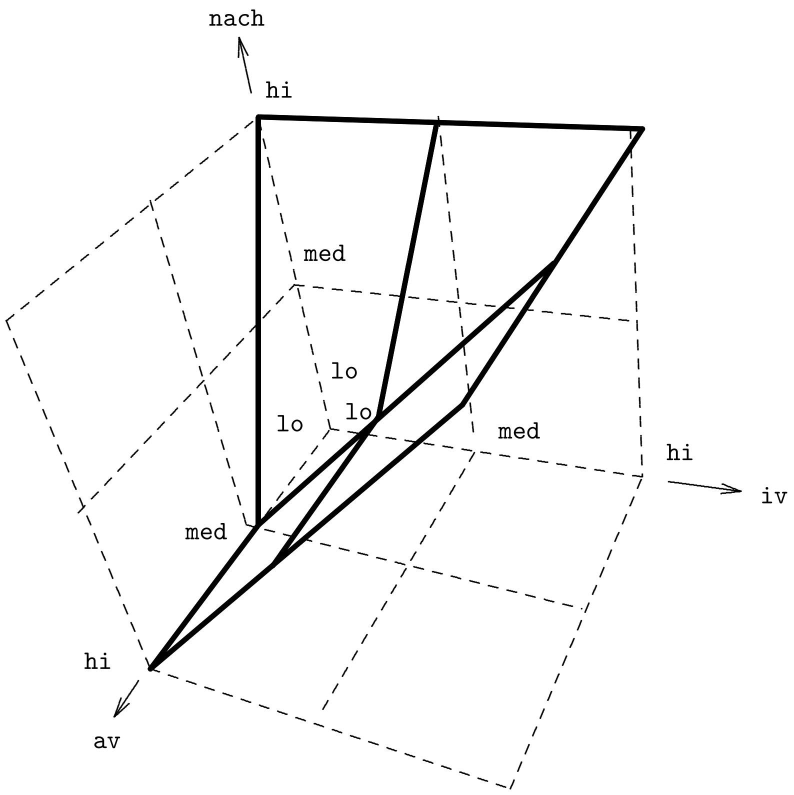

Figure 6: Rules for derivation of `nach` from `iv` and `av`.

5 Conclusion

We have presented a method that is to assist in discovery of the principles of the medical physiological systems. The method uses a structure of properties. An additional information used for automatic derivation of principles is supplied in form of the examples that map input properties to output and are, in the case of realistic models, derived through simulation.

An example given in Sec. 4 shows that although the predictive accuracy of the method might be low for the systems of low complexity, the same principles as with using more complex systems might be discovered. The advantage of the less complex systems are its higher transparency and lower computational requirements.

We are currently exploring several issues of the method. These include the selection of appropriate error estimate, impacts of the genetic representation scheme to a learning time, and the automatic derivation of property structure. Future work includes also the estimation of prediction accuracy of the method and its dependency on complexity of the systems.

Although the outlined approach is targeted for computer assisted reasoning with a general class of realistic models, the idea for its development and the first tests where done on a specific model. This realistic nerve fiber model was developed at Baylor College of Medicine in Houston and is used to predict the functional implications of neuronal structural and biophysical properties. The idea we are pursuing is to embed both realistic model and methods presented in this paper in a framework that would be efficiently used in the exploration of properties and principles in neurobiology.

References

[1] A. W. Bierman, J. Fairfield, and T. Beres. Signature table systems and learning. *IEEE Trans. Syst. Man Cybern.*, 12(5):635–648, 1982.

[2] A. W. Bierman, K. C. Gilbert, A. F. Fahmy, and B. Koser. On the errors that learning machines will make. *International Journal of Intelligent Systems*, 9:269–302, 1994.

[3] M. Bohanec, I. Bratko, and V. Rajkovič. An expert system for decision making. In H. G. Sol, editor, *Processes and Tools for Decision Support*. North-Holland, 1983.

[4] M. Bohanec and V. Rajkovič. DEX: An expert system shell for decision support. *Sistemica*, 1(1):145–157, 1990.

[5] M. G. Cooper and J. J. Vidal. Genetic design of fuzzy controllers: the cart and jointed-pole problem. In *Third IEEE Int'l. Conf. of Fuzzy Systems*, Orlando, FL, June 1994.

[6] K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[8] J. A. Halter, J. S. Carp, and J. W. Wolpaw. Operantly conditioned motoneuron plasticity: possible role of sodium channels. *J. Neurophysiology*, 73(2):867–871, 1995.

[9] J. A. Halter and J. W. Clark. A distributed-parameter model of the myelinated nerve fiber. *J. Theo. Biol.*, 148:345–382, 1991.

[10] J. A. Halter and J. W. Clark. The influence of nodal constriction on conduction velocity in myelinated nerve fibers. *Neuro Report*, 4:89–92, 1993.

[11] H. Ishibushi, K. Nozaki, N. Yamamoto, and H. Tanaka. Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(3):260–270, 1995.

[12] C. L. Karr and E. J. Gentry. Fuzzy control of pH using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 1(1):46–53, 1993.

[13] C. Koch and I. Segev, editors. *Methods in neuronal modeling.* MIT Press, 1989.

[14] D. Levine. User's guide to PGAPack parallel genetic algorithm library. Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, June 1995.

[15] R. J. Machado and A. F. de Rocha. A hybrid architecture for fuzzy connectionist expert systems. In A. Kandel and G. Langholz, editors, *Hybrid architectures for intelligent systems.* CRC Press, 1992.

[16] M. Phillips. Geomview manual. The Geometry Center, University of Minnesota, October 1994.