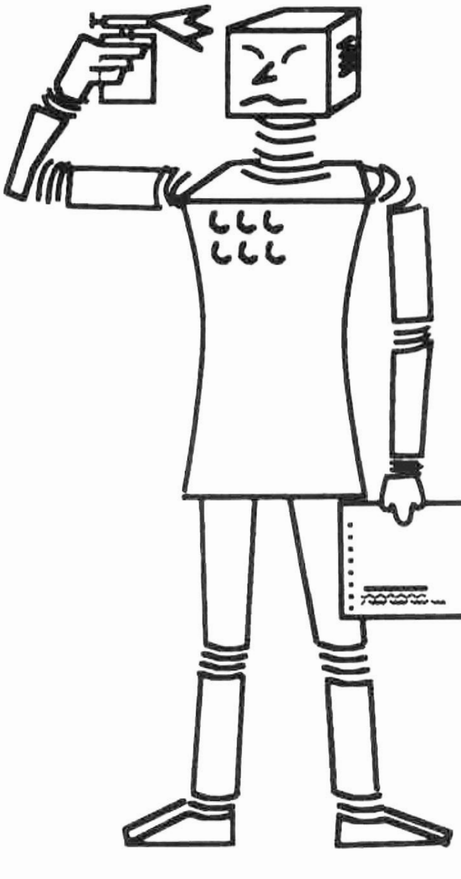
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT



Automated Data Analysis and Discovery in Neurophysiological Simulation Experiments using a Combination of Numerical and Symbolic Methods

Stefan Schrödel, Oliver Wendel

SEKI Report SR-92-19

Automated Data Analysis and Discovery in Neurophysiological Simulation Experiments using a Combination of Numerical and Symbolic Methods[*]

Stefan Schrödl[†]
Dept. of Computer Science
University of Kaiserslautern
W–6750 Kaiserslautern, Germany

Oliver Wendel [‡]
Dept. of Computer Science
University of Kaiserslautern
W–6750 Kaiserslautern, Germany

Abstract

Raw data derived from experiments or numerical simulations in a certain domain is at an abstraction level far too low to be used for the interpretation by a computer. In order to detect similarities, relations or dependencies within different data sets it is often advisable to construct a qualitative description using various transformational steps. Ideally this process should lead to the same high–level symbolic form that human researchers are used to dealing with themselves. Enabling a system to perform this transformation and to make inferences based on the symbolic description represents a first step towards automatic discovery. In this paper, we report on the above–mentioned aspects of the *MOBIS* (Modelling of Biological Systems) project, a case–based, interactive simulation environment which is designed to assist neurophysiologists to step through the experiment life–cycle of design, simulation, and analysis of neurophysiological simulation experiments. We are going to present one component of the intelligent assistant whose purpose is to automatically simplify, analyze, and interpret complex numerical neurophysiological data derived from real experiments or — as in our case — from the results of computerized simulation.

1 INTRODUCTION

Although this paper presents an AI application in neurophysiology, we omit an in-depth introduction to biological neuronal networks, the electro-chemical processes in neurons and synapses that are modelled in the simulator and the like. Instead we assume a basic understanding of these processes and, if necessary, we include sufficient detail along the following sections so that the non–biologist can understand the rest of the paper.

Typically, simulations of biological neural networks produce only raw data, as e.g. in the simulation system *GENESIS* (Wilson, Bhalla, Uhley, Bower 90). A variety of existing software tools already allow some (statistical) analysis of this data, but for a neurophysiologist, it turns out that certain ***qualitative features*** of the simulation (e.g., the presence of ***spikes***, or the fact that a neuron remains inactive throughout a certain period of time whilst another neuron shows activity) represent the main results of a numerical simulation. In this case we would like a computer program to provide (and understand) a representation of the results that includes these qualitative features. Simply graphing the results is helpful but not sufficient for these purposes: a plotting routine does serve to summarize data for the user, but it fails to provide that summarized data in a more abstract and symbolic form that may then be further examined by the computer itself (c.f.(Eisenberg 90)).

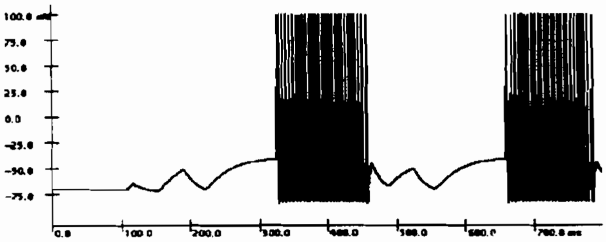


Figure 1: A typical membrane voltage plot.

Fig. 1 shows a typical plot of neuronal activity within a network. The plot is the graphical representation of the raw data derived from numerical simulation. We developed algorithms and a data structure (called ***episode–structure***) enabling us to represent an overall

[*] Appeared in: J. M. Zytkow (Ed.): Proceedings of the ML–92 Workshop on Machine Discovery (MD–92), July 4th, 1992, Aberdeen, Scotland, UK, pp. 131–136

[†] email: schroedl@informatik.uni-kl.de

[‡] email: wendel@informatik.uni-kl.de

qualitative description of the results of a simulation or of real (digitized in-vivo) experiment recordings. This data structure is coarse-grained enough to provide a substantial compaction of the numerical data while still retaining enough fine-grained detail to capture those features of the simulation that are of interest to the researcher. Additionally, this data structure is in a form that can be examined, classified, and manipulated in interesting ways by the computer program itself, allowing for intra- and inter-experiment discoveries. Since we tried to use domain independent algorithms, our system does not contain any knowledge about the simulation process, thus accounting for a shallow coupling of simulator and analysis component. Domain knowledge is needed only for *feature detection.*

The following section gives an in-depth discussion of the various transformational steps in experiment analysis that have to be accomplished in order to extract any meaningful information buried in the simulation data. Data reduction, gain of information, and the possibility of further inferences based on this symbolic representation are the main advantages of these transformational steps. We finally show how the system detects qualitative relationships in an example taken from our problem domain.

2 THE TRANSFORMATIONAL STEPS IN EXPERIMENT ANALYSIS

Neurophysiologists can observe the result of a simulation and discern a sequence of significant events. These events may be interpreted as part of a larger repertoire of qualitative behavior of neural networks. When addressing the problem of automated experiment analysis in neurophysiology, the key issue is to construct a qualitative history of membrane voltage plots. The elements of this qualitative history are called *episodes.* Fig. 1 shows a typical membrane voltage plot. In analogy to human perception, several steps of interpretation can be distinguished. Initially, the function is recognized in a hierarchical manner: a primal sketch of the rough outline is established, skipping irrelevant details, but being fine-grained enough to maintain important phenomena. The function is segmented into *intervals.* Next, sequences of successive intervals form typical patterns or *features* subject to a direct interpretation. Finally, sequences can be grouped into *repetitions.* In the following sections, these transformation steps will be discussed in more detail.

2.1 SEGMENTATION INTO INTERVALS

For segmenting functions into meaningful intervals, various kinds of points (such as extrema of a function and its derivatives) can serve as boundaries. Fig. 2 depicts the interval between 90 and 190 ms from the example voltage plot of Fig. 1. Each vertical line indicates a local extremum. Out of this set of candidates, "significant" segmentation points are to be selected, generally by application of a digital filter. Thus in Fig. 3 the (possibly irrelevant) events between 115 and 150 ms may be neglected.

An automated function segmentation is supposed to comply with the following requirements:

- Noise and unimportant details (e.g. oscillations without domain-dependent meaning) should be omitted while preserving characteristic phenomena.
- A hierarchical form would be desirable in order to examine functions on different resolution levels. E.g., in Fig. 2 the minimal modifications between 115 and 150 ms could be regarded as subintervals of the decreasing interval and, if necessary, be neglected.
- The generation of a description ought to be independent of parameter input by the user.
- Contraction or extension of the axes should not affect the interval structure.
- The segmentation process should be capable of treating arbitrarily shaped functions.
- The significance of interval boundaries should be measured not by a filter derived from the overall function, but rather by comparison with the local neighborhood.

These demands introduce the problem of *scale.* Digital filters principally smooth a curve by computing some kind of weighted mean value over a neighbor environment. The size of this environment (which can be regarded as a window) can be determined by modifying the distribution parameter σ. Finding a suitable σ proved to be nearly impossible in two respects: a chosen value resulting in an adequate description of one example function yielded unacceptable segmentations of the other (and vice versa); moreover, it didn't suffice to apply one fixed filter size to the entire potential function, but it had to be adapted to the local environment. In order to gain an interval segmentation which fullfils the above-listed requirements, it seems inevitable to exploit the idea of *scale space.*

2.2 THE SCALE SPACE

Function segmentations on different scales (i.e., resolution levels) can be obtained by continuous smoothing using convolution with some filter operator. The parameter of standard deviation σ indicates the scale of the filtered signal. With x denoting the signal axis, the (x, σ)-plane is called *scale space* (Shapiro 88). Extrema on different scales but similar positions now appear as the same event seen through different filters. Events consist of trajectories of single points along the

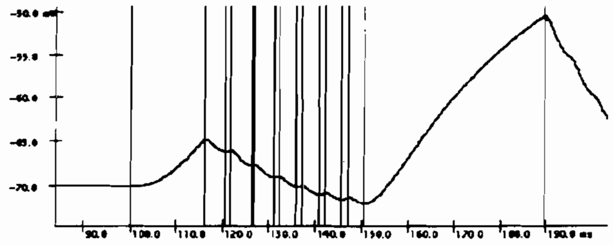


Figure 2: Section from Fig. 1.

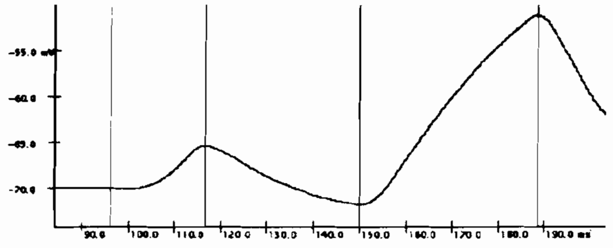


Figure 3: Filtered function.

σ-axis. By increasing σ, they can move along the x-axis and finally disappear in pairs (e.g., a minimum merges with the neighbor maximum). This vanishing scale can be seen as a measure of significance.

2.3 THE INTERVAL TREE

The scale space extrema whose scales exceed a given threshold σ_T partition the x-axis into intervals. As σ_T is decreased, starting from a coarse scale, new events appear in pairs (each associated with an "apex" singularity in the scale space image), dividing the enclosing interval into a triple of subintervals. As σ_T is decreased further, these new intervals in turn subdivide, down to the finest observable scale.

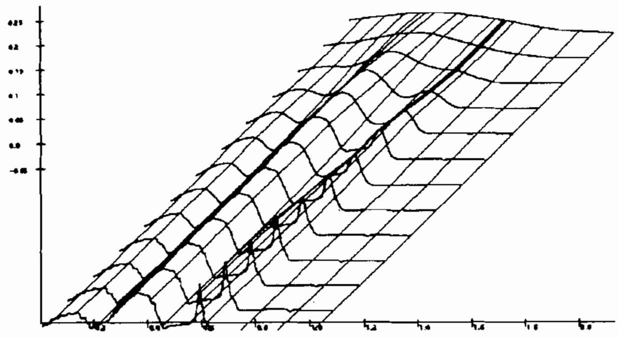


Figure 4: The scale space. The z-axis represents $\log \sigma$. By decreasing the resolution level of a signal function, extrema become flat and extend until they finally disappear. This end point indicates the scale and can be regarded as a measure of significance.

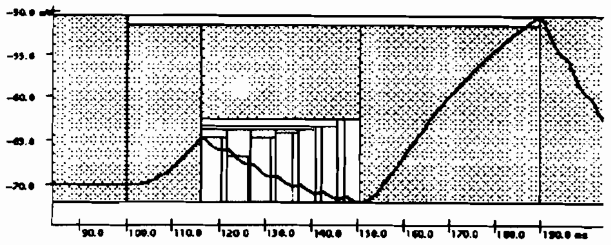


Figure 5: Interval tree of the example function. Shaded areas mark the interval nodes chosen by the stability criterion.

The intervals correspond to nodes in a ternary-branching tree: an interval's parent is the larger interval from which it emerged, and its offsprings are the subintervals into which it divides. Fig. 5 shows the interval tree of the example.

2.4 GAINING A FUNCTION DESCRIPTION

All sets of intervals from the interval tree satisfying the condition that every point on the x-axis is covered by exactly one node represents a valid segmentation of the function. From any starting point, one may generate a new segmentation either by splitting an interval into its offspring, or merging some intervals into their common parent. The space of descriptions can be thus explored. Note that senseless descriptions (i.e., an interval contains an extremum of larger scale than either of the two bounding ones) are excluded from the interval tree.

The final description can be derived from the interval tree by specification of a so-called ***stability criterion***. An interval appears to be the more "intuitive", the larger the ratio (scale of bounding extrema/scale of offspring) is. Such an interval should be preferred in the choice of interval nodes.

2.5 IMPLEMENTATION OF THE SCALE SPACE METHODS

A complete computation of the scale space or the correct gaussian filtering of one function would be too time-consuming. Thus a filter operator was developed which can be computed in constant time for one value, but imposes the disadvantage of being inexact in some cases (especially in functions with a relatively high density of extrema). The determination of the vanish scales of the extrema avoids filtering the whole function; extrema are traced through scale space by successive steps until they can't be observed any more. These steps comprise of little increments of σ and subsequent tests to find out the extrema's change in position.

The interval tree of our example in Fig. 2 is shown in Fig. 5. The modulations have been classified as subintervals, providing the desired hierarchical description.

2.6 FEATURE CLASSIFICATION

We aim at obtaining a function description in terms of "meaningful units". Such *features* represent typically shaped regions within a function where a domain-specific interpretation can be directly associated with. They consist of a number of subsequent function segments gained from the interval tree. Figures 6 — 10 show examples of such membrane potentials, taken from actual simulation runs. Feature classification is the transformational step where domain-dependent knowledge is introduced for the first time.

Features are detected by a simple rule interpreter, which classifies sequences of function segments according to some properties (e.g. length, slope, curvature etc.). The rules are applied with a priority defined by their ordering. All function segments are classified, at least as dummy "unknown"-features.

For different types of functions, separate rule sets are applied:

- Neuron potentials: spike, epsp, near miss (epsp just below spike threshold), ipsp, constant.
- Stimulus functions (applied to neurons): rising, falling, constant, zero.

2.7 GROUPING INTO REPETITIONS

Some phenomena as e.g. spikes often appear in packets (this phenomenon is called *burst*, see Fig. 10). Especially repetitions can be analytically exploited by asking "how does a property of a feature change from one occurrence in a repetition to the next?". Thus it makes sense to think of repetitions of features (or combinations of them) as episodes rather than of single features themselves.

Our system finds the shortest possible description in terms of repetitions; these repetitions also can be nested. For example, if A, B, C are features, then the descriptions of the sequences

$$ABABCABC, \quad AAABCAAABC$$

become

$$AB(ABC)^2, \quad (A^3BC)^2$$

respectively.

Within our system, the recognition of repetitions represents the final step towards a "symbolic" function description. As shown in Fig. 12, the description can be visualized in textual form and integrated in an automatically generated analysis report.

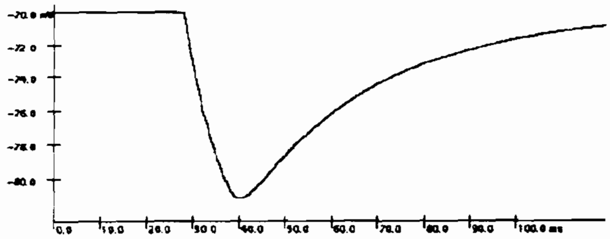


Figure 6: Ipsp: inhibitory postsynaptic potential.

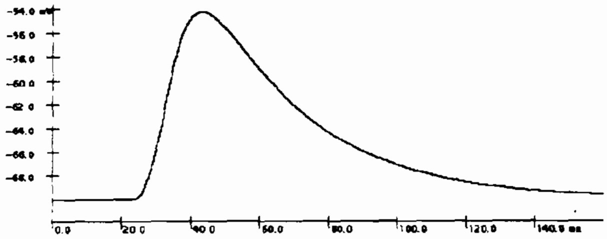


Figure 7: Epsp: excitatory postsynaptic potential.

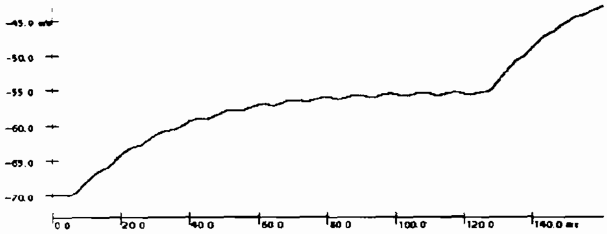


Figure 8: Superposition of epsps.

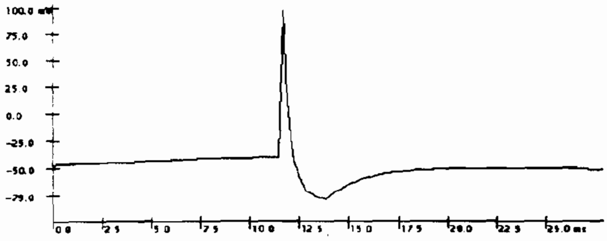


Figure 9: Spike.

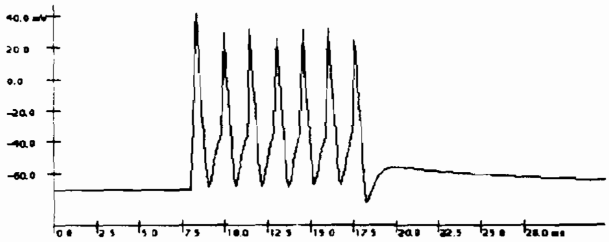
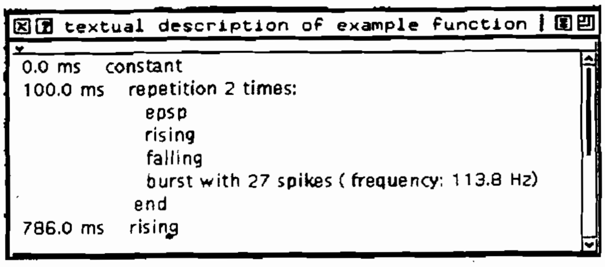


Figure 10: Burst.

```
self makeRule:
  [:each :following :
    (each firstValue lessThan: (each lastValue))
    & (following firstValue greaterThan: (following lastValue))
    & (each firstValue lessThan: (following lastValue))
    & (each slope abs greaterThan: (following slope abs))]
  is: Epsp.
```

Figure 11: Example of a rule.



```
0.0 ms    constant
100.0 ms   repetition 2 times:
             epsp
             rising
             falling
             burst with 27 spikes ( frequency: 113.8 Hz)
           end
786.0 ms   rising
```

Figure 12: Textual description of Fig. 1.

3 USING SYMBOLIC DESCRIPTIONS

3.1 FUNCTION MATCHING

Experiments frequently are performed in series with slight variation of parameters or conditions. From one simulation to another, potential plots of involved neurons look very similar. Thus the episode sequences can be mapped onto each other, and analogous episodes can be identified.

We implemented a matching algorithm for episode structures, which finds a relation with maximum total time of overlapping similar episodes. Two episodes are considered similar, if both are features of the same type or both are repetitions of similar patterns. Thus, e.g. bursts with 5 or 8 spikes can be matched. The differences between identified episodes (e.g. a change in the average frequency of a burst, the strength of a repetition of epsps and so on) are particularly relevant for experiment analysis.

Thus the matching algorithm can be used to discover dependencies between experiment parameters and neuron behavior. The user may define formulae constructed of episode parameters. Similar episodes within the experiment series are matched, consequently the variation of the specified formula can be traced automatically, hence supporting the neurophysiologist's analysis. The generated dependency function could be submitted to the same transformation process and be described symbolically. Moreover, it can be used for predicting experiment results by means of correlation analysis.

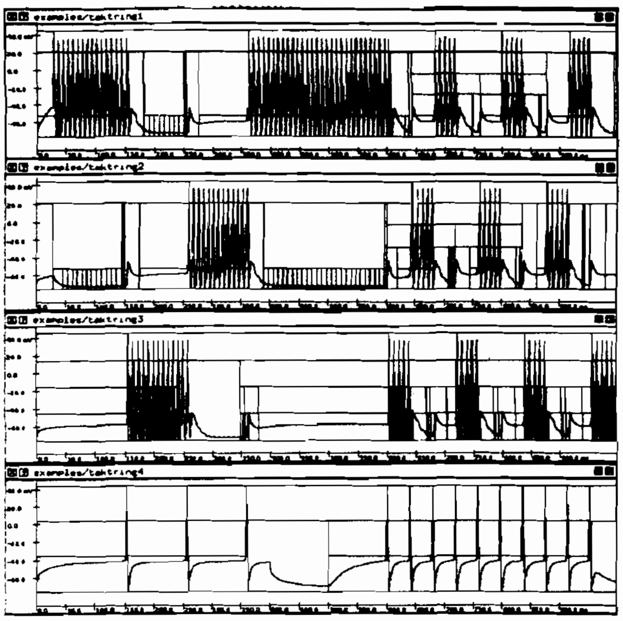


Figure 13: Four neuron potentials.

3.2 NEURON POTENTIALS SEEN AS PATTERN LANGUAGE

Another issue we are currently investigating is the interpretation of the symbolic representation of transformed neuron potentials as a sentence of a pattern language "spoken" by the neuron. An interesting question thus could be: "What is the underlying grammar of a neuron's language?"

The overall behavior of the entire network can be expressed using tokens of vectors consisting of episodes that occur simultaneously in different neurons.

In the example of Fig. 13, which shows the neuronal activity of a 4–neuron network, the behavior can be described by the pattern

$$\begin{bmatrix} s* \\ i* \\ r \\ r \end{bmatrix} \begin{bmatrix} i \\ e \\ s \\ s \end{bmatrix} \begin{bmatrix} i* \\ r \\ s* \\ r \end{bmatrix} \begin{bmatrix} e \\ s \\ i \\ s \end{bmatrix} \begin{bmatrix} r \\ s* \\ i* \\ r \end{bmatrix} \begin{bmatrix} s \\ i \\ e \\ s \end{bmatrix} \begin{bmatrix} s* \\ i* \\ r \\ f \end{bmatrix}$$

$$\begin{bmatrix} s* \\ i* \\ r \\ r \end{bmatrix} \begin{bmatrix} i \\ e \\ s \\ s \end{bmatrix} \begin{bmatrix} i* \\ f \\ s* \\ r \end{bmatrix} \begin{bmatrix} e \\ s \\ i \\ s \end{bmatrix} \begin{bmatrix} f \\ s* \\ i* \\ r \end{bmatrix} \begin{bmatrix} s \\ i \\ e \\ s \end{bmatrix} \begin{bmatrix} s* \\ i* \\ f \\ r \end{bmatrix}$$

$$\begin{bmatrix} i \\ e \\ s \\ s \end{bmatrix} \begin{bmatrix} i* \\ f \\ s* \\ r \end{bmatrix} \begin{bmatrix} e \\ s \\ i \\ s \end{bmatrix} \begin{bmatrix} f \\ s* \\ i* \\ r \end{bmatrix} \begin{bmatrix} s \\ i \\ e \\ s \end{bmatrix} \begin{bmatrix} s* \\ i* \\ f \\ r \end{bmatrix} \begin{bmatrix} i \\ e \\ s \\ s \end{bmatrix}$$

$$\begin{bmatrix} i* \\ f \\ s* \\ r \end{bmatrix} \begin{bmatrix} e \\ s \\ i \\ s \end{bmatrix} \begin{bmatrix} f \\ s* \\ i* \\ r \end{bmatrix} \begin{bmatrix} s \\ i \\ e \\ s \end{bmatrix} \begin{bmatrix} s* \\ i* \\ f \\ r \end{bmatrix} \begin{bmatrix} i \\ e \\ s \\ s \end{bmatrix} \begin{bmatrix} i* \\ f \\ s* \\ f \end{bmatrix}$$

where each 4–tupel of simultaneous episodes is regarded as attributed character. Each character is interpreted as follows: s – spike, i – ipsp, e – epsp, f – falling, r – rising, * – repetition.

Several observations can be made using this representation: for example, in the first tuple a spike in neuron 1 ($s*$) occurs with an ipsp in neuron 2 ($i*$). Neurons 1 and 2 exhibit a similar behavior five tuples later, inducing the hypothesis of inhibitory coupling between these two neurons. The characteristic property of this type of network is the existence of three distinct states, where only one of the neurons can fire.

The following subpattern can be identified within the above–listed token sequence where it is repeated three times:

$$\begin{bmatrix} i \\ e \\ s \\ s \end{bmatrix} \begin{bmatrix} i* \\ f \\ s* \\ r \end{bmatrix} \begin{bmatrix} e \\ s \\ i \\ s \end{bmatrix} \begin{bmatrix} f \\ s* \\ i* \\ r \end{bmatrix} \begin{bmatrix} s \\ i \\ e \\ s \end{bmatrix} \begin{bmatrix} s* \\ i* \\ f \\ r \end{bmatrix}$$

It reflects exactly the cyclic state transitions of the network.

Another promising approach using the presented symbolic description will involve the applicability of Allen's temporal logic (Allen 83) to describe temporal relationships between episodes. Investigations in this direction have just begun.

4 SUMMARY AND CONCLUSION

In this paper we described the analysis component of MOBIS, a case–based system that is capable of carrying out a complex series of neurophysiological simulation experiments. The analysis phase performed after each simulation run transforms the computed raw data into an abstract symbolic description which then is used to detect correlating features in the behavior of different neurons, between neurons and stimulus functions or even between the behavior of neurons across different experiments. Establishing matches between different episodes and the application of statistical analysis tools induce hypotheses about the structure of the neuronal network that exhibits the observed behavior.

Analysis of the system's performance on a number of different problems taken from various simulation results, different simulators and even digitized recordings from in–vivo experiments has shown its stability and usefulness. A major drawback of the approach is the fact that functions consisting of many tiny superimposed signals cannot be processed adequately.

MOBIS refers to ongoing efforts rather than to the ultimate system. The simulator with the underlying mathematical model of neuronal activity is fully implemented and can be used as a stand–alone system without the knowledge–based intelligent assistant. It is written in C and runs under OSF/MOTIF on Unix workstations. The experiment analyzer as one part of the intelligent assistant is fully implemented as well. The implementation language is Smalltalk–80 and it runs on any platform for which Objectworks Smalltalk–80 Release 4 by ParcPlace Systems is available.

The whole system is developed and used in conjunction with a neurophysiology project in the Kaiserslautern Department of Biology investigating the neuronal grounds of the *femur–tibia* junction in the stick insect *carausius morosus.*

We believe our work contributes to the state of current research on scientific discovery by proposing a computational approach that can — due to its primarily data driven nature — be applied to a wide variety of domains, such as interpretation of electroencephalograms or seismographic data.

Acknowledgements

We wish to thank all people in the MOBIS project group. Special thanks are due to Prof. M. M. Richter for his comments and various discussions on our project.

References

Allen J. F.: *Maintaining knowledge about temvoral intervals.* In: Comm. ACM 26(11), Nov. 1983.

Eisenberg M.: *Descriptive simulation: combining symbolic and numerical methods in the analysis of chemical reaction mechanisms.* In: Artificial Intelligence in Engineering, 1990, Vol. 5, No. 3, 161–171.

Shapiro S. C. (ed.): *Encyclopedia of Artificial Intelligence* (Vol. 1, 2). Wiley, 1988.

Wilson M. A., Bhalla U. S., Uhley J. D., Bower J. M.: *GENESIS: A system for simulating neural networks.* Technical report, California Institute of Technology, 1990.