A System for Using Time Dependent Data in Patient Management

Thomas A. Russ

MIT Laboratory for Computer Science, Cambridge, MA 02139 U.S.A.

Traditional artificial intelligence (AI) in medicine systems do not operate in an environment in which continuously changing data is encountered. The standard approach of using a rule-based system to address a domain with time-varying data does not work. This is because rules assume that the environment in which they operate contains only universally true assertions. This paper presents a control system that retains the clarity and modularity of rule-based systems while adding a capability for using data which changes over time. The system supports temporal reasoning by providing a framework for generating steady state abstractions from incoming data streams, and for automatically running relevant reasoning modules when required. The resulting data-driven system can support clinical patient management systems which use time dependent data.

1. INTRODUCTION

Many medical AI systems address the problem of diagnosis, most often in the context of a single consultation. While this single visit approach may be successfully used for (some) diagnosis problems, it is inadequate for therapy management. Patient management requires the tracking of the treatment being used and the patient's response over time. More than one session with the same patient is an inherent part of the management task. A management system must be able to deal with data that changes over time. In certain applications, it is also important to be able to deal with data that is not immediately available.

The course of this research into temporal control systems is motivated by the difficulties encountered in using existing AI technology for the construction of a ventricular arrhythmia advisor. This program would be used in a cardiac intensive care unit. The very nature of critical care medicine often requires therapeutic intervention before all tests can be performed and their results reported. Furthermore, the patient's condition itself can change quickly, thus making it necessary to interpret the results of tests in light of the changes that have taking place in the patient's state since the time to which those test results refer.

In a typical single-session consultation with a computer advisor, if a test result is requested, but not available, then the user has one of two choices. Either the test can be performed and the results awaited before continuing with the session or the consultation can continue without the data. If data that is relevant to the decision-making process arrives later, the consultation must either be redone with the newly available data, or that the data will be ignored. Treating the *newly reported* data as if it were *new data* is inappropriate. For example, treatment may have been administered which would have changed the patient parameters that the newly available test data measured. Even in the absence of therapy, the disease process itself could have evolved, thus changing the situation relative to the time the test was conducted. A new consultation based on the current state of the therapy, but using the old data is not correct.

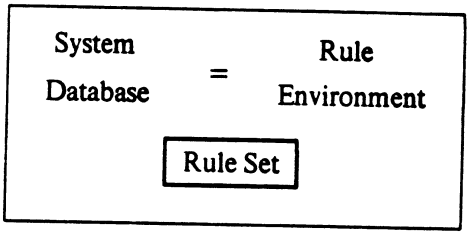

Figure 1: Standard Rule Environment.

The correct solution would be to update the execution history of the consultation system by re-executing those parts of the reasoning structure affected by the newly available data. This re-execution must take place in the historical context.

2. PROBLEMS WITH CURRENT SYSTEMS

Current rule-based expert systems do not deal with time at all. The prototypical system has a large universal data base which contains all of the valid data and conclusions reached by the system. There is no provision for limiting the validity of data to a particular period of time. This architecture is shown in Figure 1.

* This research was supported by National Institutes of Health Grant No. R24 RR01320 from the Division of Research Resources.

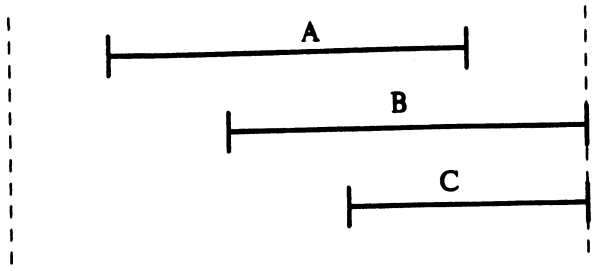

Figure 2: Duration of Premise Validity.

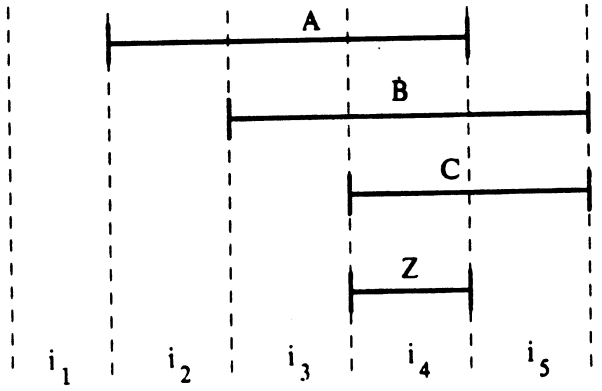

Figure 3: Database After Rule Execution.

To see where this architecture can lead to problems, consider the following prototypical rule

IF *A* and *B* and *C*
THEN *Z*.

which asserts that the conclusion *Z* is true (added to the data base), whenever the premises *A*, *B* and *C* are in the data base. Now consider the situation shown in Figure 2, where the time period over which *A*, *B* and *C* are valid is different. The ideal situation would be to have the rule produce the result shown in Figure 3, where *Z* is asserted only in interval i_4 when all of *A*, *B* and *C* are true.

It would, of course, be possible to attach some measure of the extent over which the data in the data base was valid and program the appropriate checking for concurrency in the rules themselves. Unfortunately, this would destroy one of the advantages of the rule-based paradigm, namely, that the action of the rule is a straightforward if-then statement. Cluttering the rule statement with additional tests would defeat this purpose.

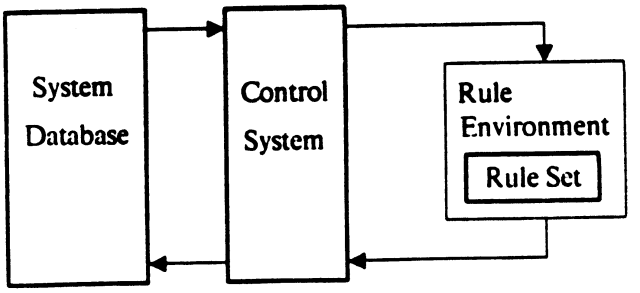

Figure 4: Rule Environment with Temporal Control.

The separation of the inference engine from the rule base was done so that the mechanical details of matching the information in the data base to the premises of the rules would not need to be explicitly programmed by the applications writer. Similarly, the mechanistic job of maintaining the temporal relationships of rule premises and rule conclusions should not be the responsibility of the applications writer, but rather it should be a service provided by the programming environment.

3. USING A STATE ABSTRACTION

The key to maintaining the advantage of the rule's perspicuity while at the same time allowing for events to change with times lies in the use of the state abstraction. This is something that is commonly done by humans in medicine. The division of a disease course into phases or the staging used in oncology are examples of imposing the abstraction of discrete states with a duration in time upon what are essentially continuous processes or a continuum of extent of disease progress. The utility of this process forms the basis of rule-based systems attractiveness for the construction of expert systems.

To accomplish this goal, a control system is introduced between the system data base and the environment in which the rules are interpreted. This new architecture is shown in Figure 4. This control system will break the time-space into intervals in which each of the premises is either valid or not. (This trivially generalizes to the case where a variable can have any number of values.) In each of these intervals, the control system invokes the rule. Referring again to Figure 3, the rule would be used once in each of the intervals i_1–i_5 and the results of the rule invocation would be combined into intervals of validity or invalidity by the control system.

Although it would be possible to restrict the rule invocation to only three intervals, i_1–i_3, i_4 and i_5, this would require the control structure to analyze the internal logic of the rule, which is impractical. Instead, the same external effect

is achieved by combining adjacent process intervals with the same value.

In summary, the control system allows time dependent conclusions by setting up a stable environment in which the variables referred to by the rule (module) have only one value. This is the provision of the state abstraction.

By using a state abstraction, similarities in cases become more easily apparent and the use of simpler decision procedures becomes possible. A rule in the production rule system is based on the assumption that the rule is operating in a static environment. This makes the decision process clearer and permits an easier explanation.

Associated with this use of a state abstraction is the need to make the abstraction based on the underlying data about the problem. The system that is described in this paper supports both the state abstraction and the process of forming abstract states from individual data points (samples). The main point of this paper is to show how the use of state abstractions allows the use of rule-like reasoning in an environment with changing data. The process of forming the state abstraction is beyond the scope of this paper.

4. OTHER WORK

Most other work done in temporal reasoning has been concerned with developing general representations for time or reasoning about temporal relationships among data that appears at different times. (See, for example, [1], [2], [3], [5] and [7].) Little research has focused on the problem of handling data that arrives over time, or on how to update existing conclusions when past data changes.

Early expert systems in medicine tended to ignore the issue of time entirely. Consultations for diagnosis were viewed as one-time events. Some provision for correcting false information was done. The MYCIN system [8], for example, used a complete recalculation system, whereas the Digitalis Advisor [10] used dependency directed backtracking to undo the error. The former approach can be very costly when applied to a large amount of data. Since the purpose of the MYCIN correction was to correct mistakes in data entry, this was not a large handicap for that system. It is, however, inadequate for a large management system where such updates are important. The strategy of dependency directed updating is therefore more appropriate. The Digitalis Advisor system was also designed around the single session concept, since only the information from the current session could be changed, but the idea of using the data dependencies can be extended to apply to multiple consultations as well. The system described in this paper follows that approach in setting up the data dependencies of the reasoning modules and using this dependency structure to drive the recalculation necessary to update the data base.

The shortcomings of a standard rule based system also lead to the development of the ONCOCIN system [9] for managing patients on cancer therapy protocols. The changes in the underlying rule processor for this system were motivated in part by the need to take data which changes over time into account. Although this system does allow handling multiple measurements and trends over time, what is still missing is an ability to change data in the past and have the system update its image of the world.

Another system that modifies the standard rule-based structure is the Ventilation Manager (VM) developed by Fagan. It accepts data as a stream, but does not have any provision for updating past information. Data that arrives is either treated as if it were current, or it is discarded as no longer reliable since the situation has changed since the time at which the data was relevant. In an application such as ventilation management, when all of the data is short-lived and results from tests and monitoring equipment is quickly available, this does not pose a serious problem. In other applications, the time delay in having data available, such as the time lag on the results of certain laboratory tests, is unavoidable. In these domains, a method of incorporating data that does not arrive in chronological order is necessary.

The VM system makes an assumption of a monotonic time-mapping between the events in the world and the time of their presentation to the program as input. It does not have any provision for handling old data that is entered into the system. This capability, however, can be important. Consider the case of blood samples being drawn from a patient and sent to the laboratory for processing. When the test results are reported, they refer to the state of the patient at the time the samples were drawn, *not* at the time the report is available. In the meantime, the state of the patient may have changed due either to the disease process or to interventions that were taken without waiting for all testing to be complete.

5. ORGANIZATION OF THE SYSTEM

The control system contains of two types of user specified entities. *Variables* are used to hold the data in the system data base. *Modules* contain the code used to perform the reasoning about the information in the data base. Time is encoded using a discrete time model with an application determined unit. The basic unit could be any period of time, since the mechanics of the updating are independent of the underlying units of the time scale. However, all times must be given specifically in terms of the basic unit time of the system.

Using the variables and modules, the system maintains the state of the data base by executing the reasoning modules when appropriate. The details of this are sketched below.

5.1. Variables

The variables in this system are divided into two classes, *point variables* and *interval variables*. Point variables are used to represent data that is associated with a specific point in time. Since a discrete time model was chosen, each instant in time is indivisible. Interval variables are used to represent data which has a value over a period of time (at least from one time point to the next one).

The point variables correspond to data samples or individual events whereas the interval variables correspond to states which have some duration. A major problem facing any temporal system designed on the basis of the control structure described here is the determination of states based on input data which is essentially a stream of point values. This is the part of the system design that is relegated to the *abstraction process*. The abstraction process is that part of the system which sets up the states (interval variables). Although this is not discussed here, an example of how this is supported in the control structure can be found in [6]. In this paper, only the effects of the interval variables will be discussed at length.

Interval variables are used to describe data that has some duration. By chosing a time interval in which all relevant interval variables have only a single value, one can achieve a *steady state abstraction*, which simplifies decision-making. This simplification is possible because it frees the reasoning process from the need to consider the effects of time or change on the decision. The price of the abstraction is that it is not possible to make the decision dependent on the facts of the change itself. This restriction is not as serious as it may at first appear. The ability of the reasoning modules to remember information (discussed below) allows past data to affect current reasoning.

The advantage of using the state abstraction of interval variables is that it allows one to state general domain principles without reference to the particular time when they may apply. The application of the rule at the proper time becomes a responsibility of the control system, not the rule writer. An example would be a rule of the following form:

```
IF the patient has low serum potassium
THEN give potassium supplements.
```

Since the conclusion depends only on the presence of the the input data, it is an example of reasoning which would benefit from the presence of the steady state abstraction. It can thus be applied by the control structure automatically and produce its conclusion each time the premises are true.

5.2. Modules

The reasoning elements of the control structure are described in *modules*. Each module defines the inputs and outputs of the reasoning code (See Figure 5). This declaration allows the control structure to track the data dependency of that particular reasoning element. For each

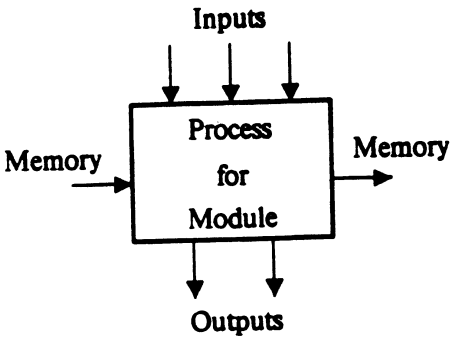

Figure 5: Process for a Reasoning Module.

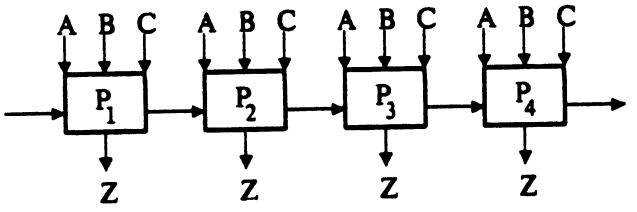

Figure 6: A Chain of Processes.

time interval in which a module is executed, the control system creates a *process*. This process will only have access to input data that concerns the time period in which it is executing. The ouput data, likewise, will only be valid over the same time period.

Since it may be desirable to have the reasoning depend either on the decisions made earlier, or on historical data derived from the earlier input data; a memory feature is also included. When a process is executed, the memory variables have the value that was stored in them by the process executing in the previous interval. These variables provide a memory capacity to the reasoning modules, which would otherwise be dependent only upon the current inputs. Since the contents of the memory variables can affect the conclusions reached, the values of the memory variables are available to the control system.

Each interval in which there is a stable set of values for the input variables has a process created and run by the control structure. The entire extent of the system data base's time line would thus be covered by a chain of processes for each module. Part of the chain that would be created for the example rule of Section 2 is shown in Figure 6. Each process (P_1–P_4) receives the appropriate interval's values of A, B and C; and when all are true asserts the conclusion, Z.

5.3. Updating Data

Modules are scheduled to run by the control system whenever one of the input or memory variables changes from its previous value. The notion of change can be controlled by the application programmer through the ability to associate

a function which tests for "sameness" with each variable. By using this data dependency scheme for running modules, it is possible to limit the updating to only those modules which are affected by the change in the data. If at some point, the new input data does not change the outputs, then the updating ends. The system propagates the effects of changed data only until a new globally stable state is reached. By limiting the recalculation to only those modules whose inputs have changed, unnecessary computation can be avoided.

6. CONCLUSION

The ability to handle data that varies over time is one that must be faced by any computer system that is used to manage patient therapy over time. The approach described here is one solution to this problem. It separates the mechanics of maintaining and updating time dependent data from the design of the rules that use this data for their decision making. This separation allows the application of general principles precisely in those time intervals in which their application is justified by the available data. The provision of a mechanism for maintaining the temporal relationships between reasoning modules fulfills a similar function in the temporal realm as the provision of a separate inference engine does for the data matching realm of traditional expert systems. This separation of function will allow more understandable systems to be produced with less programmer effort.

There are several areas for continuing research. Currently the control system requires having specific times for individual data points and for the endpoints of intervals. It is often unrealistic to expect such precise determinations of the presence or absence of states. Extending this work to allow inexact boundaries would make it possible to more accurately model non-specific knowledge. Long's work [5] gives one possible method for approaching this problem. The technical challenge is to do this in a way that also preserves the efficiency of the updating mechanism when data changes. Taking all possible combinations of overlapping endpoints could easily become computationally intractable.

More work also needs to be done on the general problem of forming state abstractions based on the stream of raw data. Another issue that is not addressed by this research is that of using the time course of the data itself to guide decision-making. This would involve the matching of temporal patterns in the data. One could use the database maintained by this control system as the basis upon which to build such a temporal pattern matching program.

An Implementation Note

The metaphor of a rule-based system was chosen for the exposition of the work in this paper because it allows a clearer presentation of the major reasoning issues underlying this work. The actual implementation of the control structure is based on a procedural rather than declarative (rule-based) style of programming. The usefulness of the state abstraction and the desirability of using this control system is not affected by this difference in the method of encoding the system knowledge.

These ideas have been tested with some simple examples in the domain of cardiology.

REFERENCES

[1] Allen, J. F., "Maintaining Knowledge About Temporal Intervals," *Communications of the ACM*, **26**:11 (November 1983) pp. 832–843.

[2] Allen, J. F., and Hayes, P. H., "A Common-Sense Theory of Time," *Proceedings, IJCAI* (1985) pp. 528–531.

[3] Blum, R. L., "Discovery, Confirmation, and Incorporation of Causal Relationships from a Large Time-Oriented Clinical Data Base: The RX Project," *Computers and Biomedical Research*, **15** (1982) pp. 164–187.

[4] Fagan, L. M., *VM: Representing Time-Dependent Relations in a Medical Setting*, Ph.D. Thesis, Department of Computer Science, Stanford University (June 1980).

[5] Long, W. J., "Reasoning About State from Causation and Time in a Medical Domain," *Proceedings, AAAI* (1983) pp. 251–254.

[6] Long, W. J., and Russ, T. A., "A Control Structure for Time Dependent Reasoning," *Proceedings, IJCAI* (1983) pp. 230–232.

[7] McDermott, D. V., "A Temporal Logic for Reasoning About Processes and Plans," Research Report RR 196, Computer Science Department, Yale University (1981).

[8] Shortliffe, E. H., *Computer Based Medial Consultations: MYCIN* (American Elsevier, New York, 1976).

[9] Shortliffe, E. H.; Scott, A. C.; Bischoff, M. B; Campbell, A. B.; Van Melle, W. and Jacobs, C. D., "ONCOCIN: An Expert System for Oncology Protocol Management," *Proceedings, IJCAI* (1981) pp. 876–881.

[10] Swartout, W. R., *A Digitalis Therapy Advisor with Explanations*, Technical Report MIT/LCS/TR-176, Massachusetts Institute of Technology Laboratory for Computer Science (1977).