N90-27290

ATS Displays - A Reasoning Visualization Tool for Expert Systems

William John Selig

Magnetospheric Physics Branch
NASA/Marshall Space Flight Center
Huntsville, Al 35812

James D. Johannes

Computer Science Department
University of Alabama in Huntsville
Huntsville, Al 35899

ABSTRACT

Reasoning visualization is a useful tool that can help users better understand the inherently non-sequential logic of an expert system. While this is desirable in most all expert system applications, it is especially so for such critical systems as those destined for space-based operations. A hierarchical view of the expert system reasoning process and some characteristics of these various levels is presented. Also presented are Abstract Time Slice displays, a tool to visualize the plethora of interrelated information available at the host inferencing language level of reasoning. The usefulness of this tool is illustrated with some examples from a prototype potable water expert system for possible use aboard Space Station Freedom.

Introduction

We interact with expert systems for a variety of reasons: to develop/debug them; to analyze them; to use them to obtain answers in their programmed area of expertise; to learn a tutored subject from them; or just to understand the underlying inferencing process. During all these uses, one can benefit from an understanding of the reasoning process of the expert system. While it can be argued that this understanding is useful for all expert system applications, it becomes increasingly important for systems in critical application environments, such as space-based systems.

There are two major impediments to obtaining this understanding. First, humans have a basic limitation on the number of concepts that can be maintained in immediate attention, the combination of short term memory and the processing done therein. This limit is the oft cited seven ± two "chunks", where a chunk is some unit concept. Second, the information germane to acquiring this understanding is usually presented at too low a conceptual level. There is so much detail presented that one expends significant mental effort trying to combine the detailed information into a coherent "picture", a higher level conceptualization (7). The first limitation is innate. The second limitation arises because current information presentation methods present too much information at too detailed a level. If the information were to be presented at an appropriately higher conceptual level, the basic human cognitive limitation could be at least partially circumvented.

As an example, consider a set of real numbers related by the equation, y=cos(10*PI*x/100)exp(-x/20), x=0,99, an exponentially decaying sinusoid. If this set of numbers is conveyed as the list in Figure 1,

00	1.00000	0.904673	0.732029	0.505911	0.253002
05	-3.40425e-08	-0.228926	-0.414205	-0.542300	-0.606420
10	-0.606531	-0.548712	-0.443998	-0.306851	-0.153453
15	5.63291e-09	0.138850	0.251228	0.328922	0.367813
20	0.367879	0.332811	0.269298	0.186114	0.0930739
25	-9.67717e-08	-0.0842171	-0.152378	-0.199501	-0.223090
30	-0.223130	-0.201860	-0.163338	-0.112884	-0.0564522
35	1.15318e-07	0.0510803	0.0924217	0.121004	0.135311
40	0.135335	0.122434	0.0990693	0.0684676	0.0342401
45	-1.04287e-07	-0.0309818	-0.0560566	-0.0733924	-0.0820701
50	-0.0820850	-0.0742601	-0.0600886	-0.0415277	-0.0207676
55	-3.78490e-08	0.0187914	0.0340001	0.0445147	0.0497780
60	0.0497871	0.0450410	0.0364456	0.0251878	0.0125962
65	-2.66556e-08	-0.0113976	-0.0206221	-0.0269996	-0.0301919
70	-0.0301974	-0.0273188	-0.0221053	-0.0152772	-0.00763999
75	1.40223e-09	0.00691299	0.0125079	0.0163761	0.0183123
80	0.0183156	0.0165697	0.0134075	0.00926608	0.00463387
85	8.10506e-09	-0.00419293	-0.00758645	-0.00993257	-0.0111070
90	-0.0111090	-0.0100500	-0.00813209	-0.00562015	-0.00281059
95	6.15401e-09	0.00254315	0.00460141	0.00602442	0.00673673

Figure 1

roughly equivalent to the textual traces available from most current expert system shells, one would be hard put to mentally determine the interrelationships of those numbers. If instead this data is presented in a graphical form as in Figure 2,

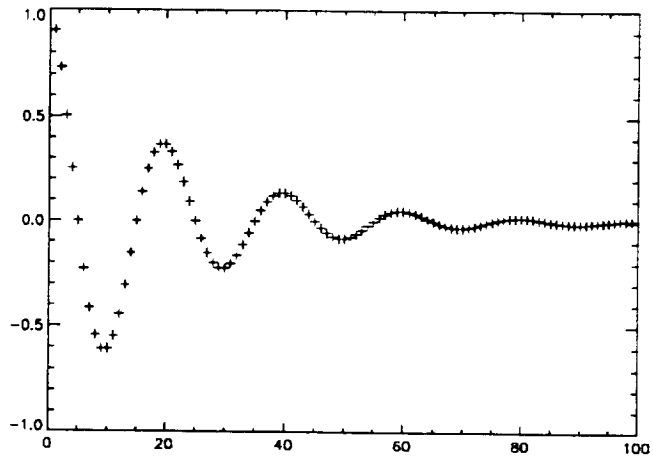

Figure 2

the interrelationships of the data values are immediately obvious. The detail of the exact individual values has been lost, but information on a higher conceptual level has been obtained, and with much less work on the part of the observer.

This is exactly the problem facing the individual trying to understand the reasoning processes of an expert system. Current environments present too much data at too low a level of detail for easy understanding of the processes involved. This problem in another form is the basis for the explanation systems/language generation systems field (4, 6). For similar reasons, in the realm of procedural programming, algorithm animation research is flourishing in order to illustrate the underlying processes of various sequential algorithms (3, 5, 11).

This paper focuses on the reasoning processes of expert systems and a tool for visualizing a subset of those processes. While there are various paradigms for expert system reasoning (1, 12), it was decided to initially focus this research effort on the forward-chaining rule-based paradigm. Per this focus, CLIPS, a readily available expert system shell, was chosen as the research vehicle.

Levels Of Reasoning

The reasoning of an expert system may be viewed on a continuum from that of the microcode of the hardware up through the programming language in which the host inferencing language is implemented on up to a "black-box" view of the application in which one sees only inputs and outputs. Pragmatically, the lowest level view which is of interest is that of the host inferencing language in which the expert system is implemented since this is the most primitive level at which the reasoning of the application may be specified.

Given an existing (or planned) expert system application, it can be viewed as having two distinct reasoning components. One is that of the application itself and is represented by the most abstract processes that define the reasoning of the expert system application. The result of the reasoning of this component consists of the inputs and outputs of the application, some subset of which are provided by and/or to the user. These inputs and outputs comprise the black-box view of the application. The other component is that of the inferencing language in which the application is (or will be) actually implemented. This is the part seen more by the expert system developer.

At the reasoning level of the host inferencing language, the reasoning is in terms of the primitives of that language. In most rule-based expert systems, these primitives are the facts and rules. The operations on these primitives are the assertion and retraction of facts, an initialization operation which asserts a set of predefined facts, the input and output of information, and the activation, deactivation and firing of rules to/from the agenda. While some rule-based expert system development languages (e.g. ART) also include the operation of existing rules defining new rules, for this research effort that operation is not considered.

The specifics of the application do not affect the actual reasoning primitives or the operations upon them, just the sequence order of their occurrence. Thus, at this reasoning level one can provide an ad hoc reasoning visualization. Users of this level would be system implementors desiring a gestalt of the low level reasoning in order to detect anomalous behavior and indications of application (in)efficiency.

At the highest level, that of the abstract reasoning process, the application reasoning is viewed in its most abstract form. This typically corresponds to metarules in more complex applications. At this level one is considering the application as conceived by the designer, not as it will be implemented.

Between these two endpoint levels are various mixtures of the two which are conceptually grouped into a realized application level. At this level the reasoning primitives are application oriented, even if they are associated with host inferencing primitive operations. One is interested in the concepts which rule firings represent, not merely that rules have fired. However, the visualization of the reasoning processes involved is in a form that is related to the inferencing language implementation of the application as opposed to the abstract reasoning processes, devoid of implementation considerations. This distinction between the abstract and realized application levels is similar to the conceptual versus implementation distinction made by Buchanan and Smith(2). For visualization of these

application based reasoning levels an ad hoc method is not possible. Instead, one must provide flexible facilities to allow users to construct their own custom visualizations(9).

The above discussion leads to the proposal of a hierarchy of reasoning levels. Conceptually, one can identify a continuous hierarchy of the reasoning process, based on a decreasing amount of detail, grouped into three ranges as shown in Figure 3. This hierarchy is viewed as a continuum since the abstract application level covers a range of detail as does the realized application level. One may also consider a range of detail at the host inferencing language level (e.g. all rules, some rules only, rule groupings). Note that this is but a more abstract view of our previous work on this subject(10).

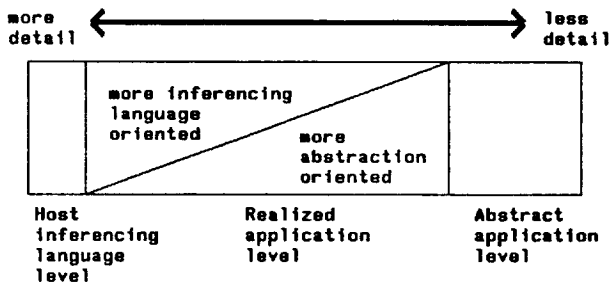

Figure 3

In focusing on the lower level primitive reasoning of an expert system, such as would be of interest during development/debugging and efficiency analysis, one is interested in the reasoning at the host inferencing language level. A model of a rule-based host inferencing language may be represented as the 4-tuple

$$M_{hil} = [F, R, A, O]$$

where

F = the set of facts, initially empty

R = the set of rules with antecedent patterns to be matched by facts $\in$ F

A = the agenda; an ordered set of instantiations of rules $\in$ R matched by facts $\in$ F; initially empty

O = the set of operations that modify F or A, or perform I/O

The operation set, O, consists of fact assertion, fact retraction, rule activation, rule deactivation, rule firing, input, output, and an initialization operation which causes predefined facts to be asserted.

Fact assertion and retraction are associated with the execution of the consequents of a rule instantiation that has been removed from the agenda for "firing". The initialization operation may be considered a pseudo rule firing in this respect since it also asserts facts. I/O is associated only with a rule firing.

Rule activation and deactivation are associated with changes to the set F. If a fact is added to F such that some rules $\in$ R are completely matched resulting in new instantiations of those rules, these instantiations are added to the set A. That is rule activation. If a

fact $\in$ F is retracted and the previous assertion of that fact had resulted in the addition of instantiations of some rules being added to the agenda, then any of those instantiations remaining on the agenda are removed. That is rule deactivation.

Of particular interest at this reasoning level is the traffic over time on both the fact list and the agenda, and the cause/effect interrelationships of that traffic. One would like to see a gestalt of the overall flow of the host inferencing language reasoning process and thus identify various aspects of operation, both normal and abnormal. One would like to be able to see sequences of rule group firings indicating phased rule operations along with the phasing control. Errant rule firing from inappropriate rule groups is also of interest. Also desirable would be the ability to identify excess activations and deactivations which may be caused by inefficient consequent sequencing. Fact and rule effectivity and agenda traffic density should also be easily observable. The information of interest at this reasoning level is similar to the information of interest to a person monitoring a wide area network system. The content of the actual data moving about the network does not matter. What is of concern is being able to tell that it is moving correctly and efficiently and that current and hopefully even potential problems can be identified easily.

Abstract Time Slice Displays

Abstract time slice (ATS) displays provide solution to the problem of the visualization of the host inferencing language reasoning level of a forward-chaining rule-based expert system. In providing a visualization of the above described reasoning there are two key issues, presentation at the appropriate level of detail and prevention of information overload even at that appropriate level. The appropriate level of detail is defined as that level at which the user immediately grasps the concept being presented. There should be little or no mental processing involved in assimilating the symbols. For presentation at the appropriate level of detail, ATS displays use unique symbols to represent the individual primitives and operations of the host inferencing language reasoning model. This obviates the need for the user to perform text to concept transformation. Instead the information is presented graphically. To prevent overload, ATS displays are static, thus providing support to the user's limited short term memory. They depict rules being activated and deactivated as the result of facts being either retracted or asserted (not respectively) as a result of other rule firings. They also show I/O as a result of rule firings. All of this information is displayed in an interrelated manner over time.

The program that generates ATS displays presently runs as a separate program taking as input two files, a segmented list of the rule names in the application and a file containing the full trace output from a run of the application. ATS outputs five files, the main display file and four adjunct files. The main display file is an ASCII file of PostScript code that creates the ATS displays on a laser printer. The adjunct files contain the details of the facts and I/O on both a time-slice and a sequential time-compressed basis. Since this capability is ad hoc and at the host inferencing language level, it could be integrated into CLIPS itself providing file output and/or direct graphical display. To understand the ATS displays, a few symbols must be defined. These are displayed and notated in Figure 4 while Figure 5 depicts various other information in the displays.

The causative symbol at a particular time slice indicates that some fact operation has caused either a rule activation or deactivation to happen (at some later time slice). A solid line connects the causative symbol to the effect caused.

The causing symbol indicates that some fact or I/O operation has been caused as a direct action of a fired rule's consequent. A solid line connects all of the actions caused by a particular rule firing, thus giving some indication of the scope and activity of a fired rule.

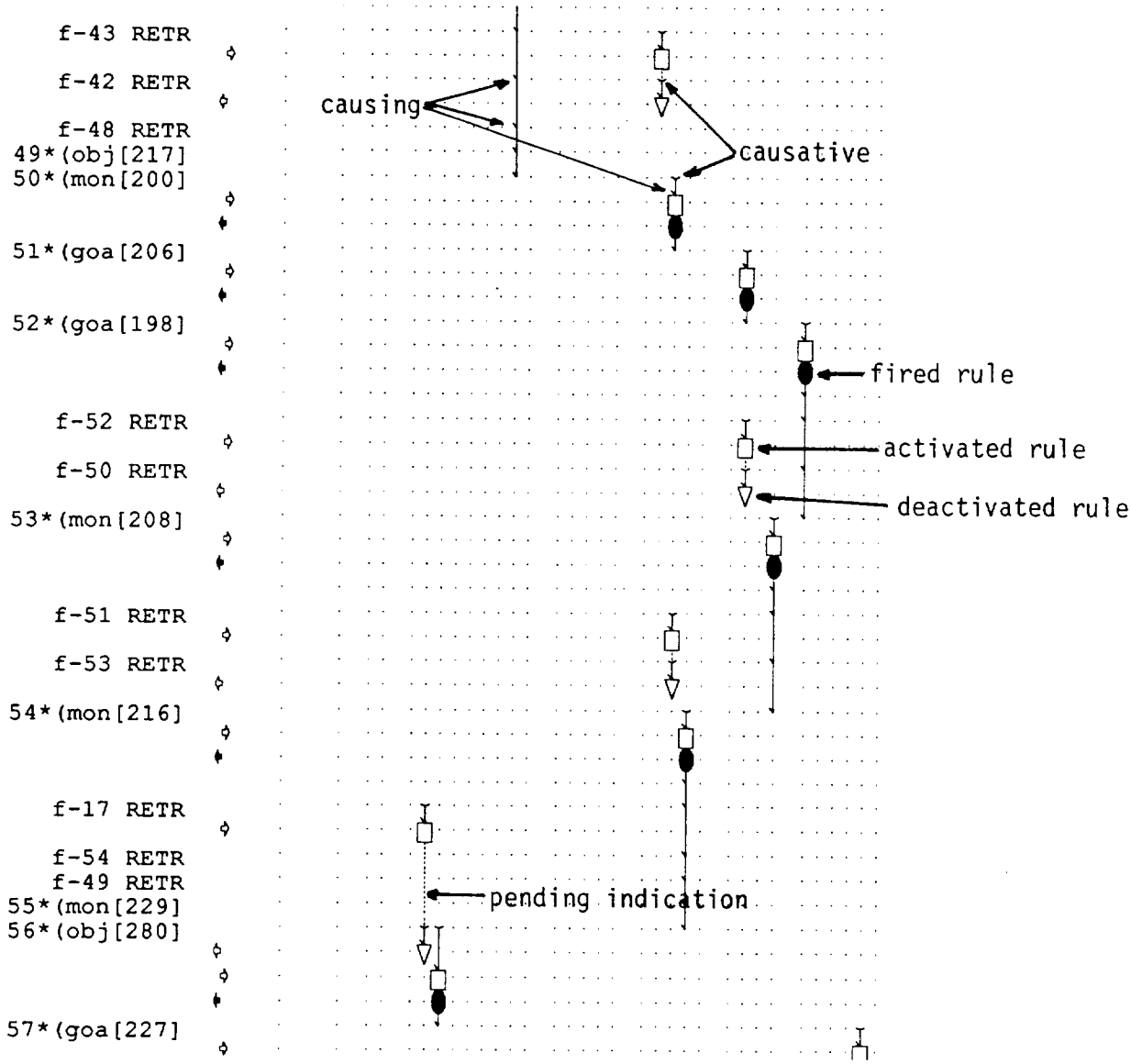

Figure 4

The causing symbol is also used to indicate a rule activation/deactivation associated with a fact operation.

The symbols for a rule activation, deactivation and firing are shown. A dotted line after a rule activation symbol is used to indicate a rule pending on the agenda. Since it is possible that more than one instance of a rule may be pending on the agenda at a given time, this is indicated by a widening of the dotted line. This indication is meant to be qualitative, not quantitative.

At the left of the rule portion of the display are symbols indicating agenda traffic. A hollow right arrow indicates a rule activation, while a hollow left arrow indicates a rule deactivation. A solid left arrow indicates a rule firing. Thus, the information inherent in the rule symbols elsewhere has been collapsed on the left side of the rule display. Note the

consistency between the hollow and solid symbols for both indications of rule activity.

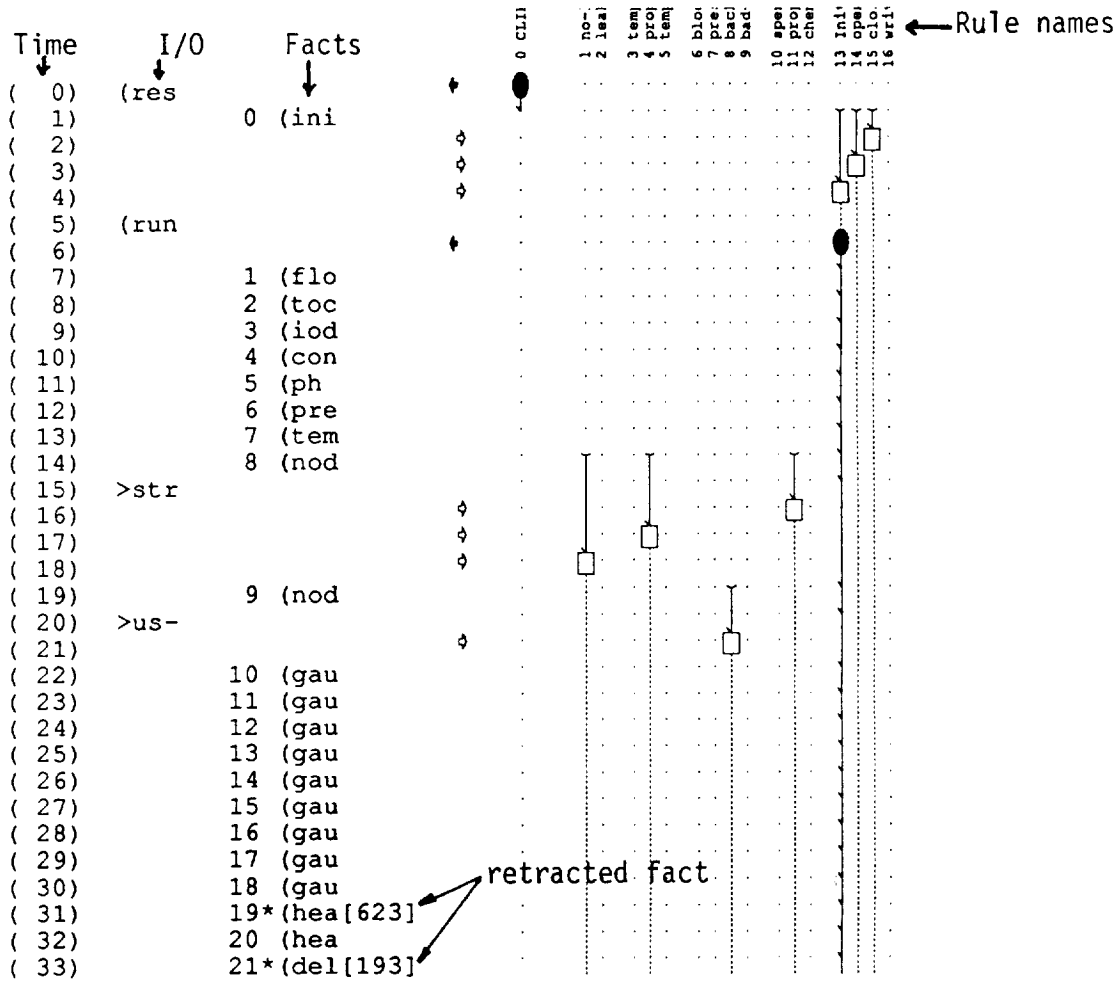

Figure 5

The rule names are listed across the top of the display while the leftmost columns indicate time, I/O, and fact information respectively. The time column is straightforward. A time slice is defined to be one trace output line. The I/O column contains both CLIPS directives, such as '(reset)' and '(run)', and actual application I/O. The CLIPS directives are indicated by a leading left parenthesis. Actual I/O is indicated by an imposed leading '>'. In the fact column is the information of the fact number and the first few characters of the fact itself. For the full details of the fact (or the I/O) one can refer to one of the adjunct files. Additionally, if the fact has been retracted, there will be an asterisk after the fact number and the retraction time in brackets after the initial fact characters. There will also be, at the retraction time, an indication of that fact number being retracted.

To aid in visual grouping of the abstract symbols, the user may specify in the segmented rule list the order of the rules defined and a spacing between their display. Thus, one could tell when rules of different types are active and, based on one's knowledge of the intended reasoning, be able to identify rules firing out of place.

Examples

In Figure 6 one can see several instances of rules activating due to some fact operation and then deactivating due to another fact operation within the consequent range of the same rule. Investigation showed that this was caused by inefficient consequent sequencing in the rules involved. Rearrangement of the consequents of the affected rules resulted in the disappearance of the excess agenda traffic.

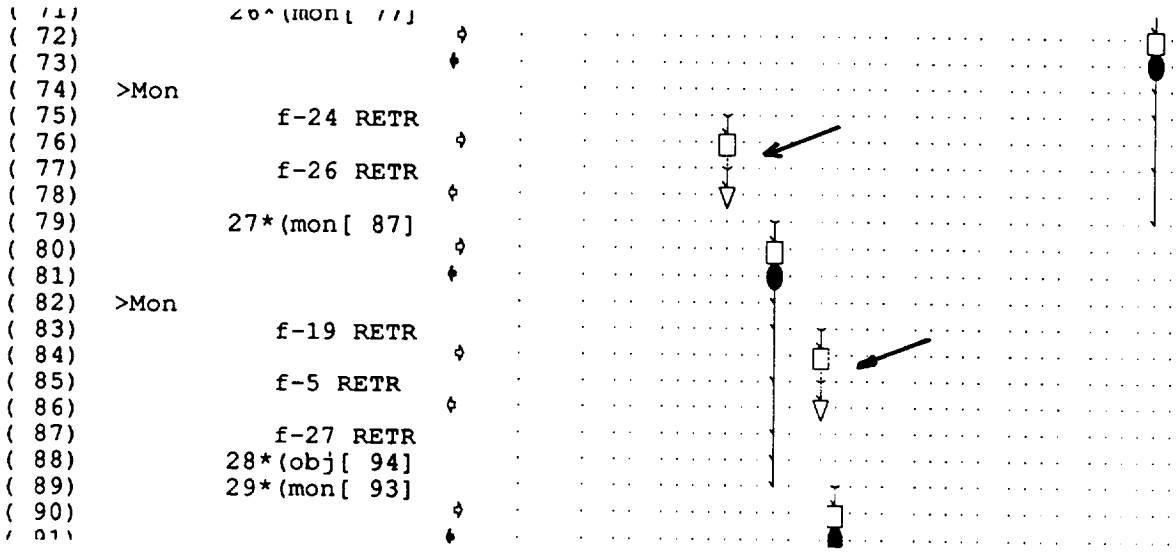

Figure 6

In Figure 7 the rules in the far right group are phase and sequencing control rules. The sequencing control rules are those that are cycling on a short time frame. The phase control rules are those that go on the agenda and then sit for a long time before firing. It can also be seen that after this transition the grouping of the remaining rule firings has changed.

In Figure 8 a rule is firing without any effects from its consequents, neither I/O or fact operations. While this appears to be an error, analysis showed that this rule was asserting a fact already in the fact list. Thus, CLIPS did not show that fact as being reasserted. In general, fact reassertion is inefficient and indicates a possible need for rule logic modification.

ATS displays were used during the development of a demonstrational prototype of an expert system application for controlling the potable water subsystem of the Environmental Control and Life Support System (ECLSS) aboard Space Station Freedom(8). There were two main contributions of ATS displays to that effort. First was the identification of inappropriate fact list changes causing rules to deactivate immediately after being activated. Second, the displays provided a picture of the overall patterns of rule activations giving a quick "state of health" view during subsequent knowledge base enhancements.

Conclusions

ATS displays have shown themselves to be useful for reasoning visualization at the host inferencing language level of a forward-chaining rule-based expert system by providing a global interrelated view of the large amount of available information. This

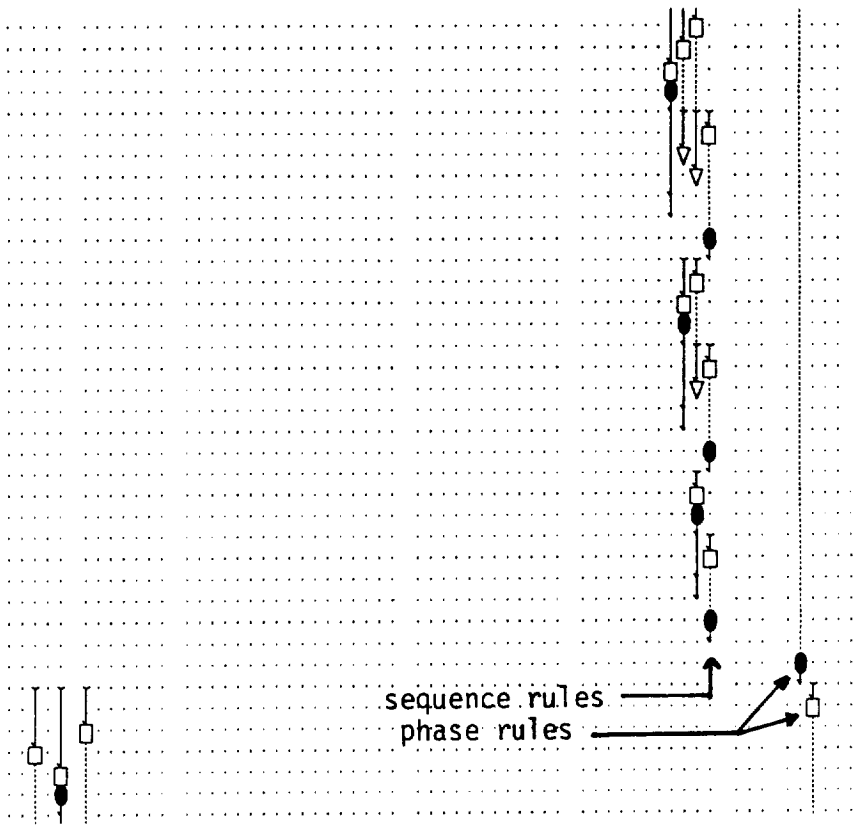

Figure 7

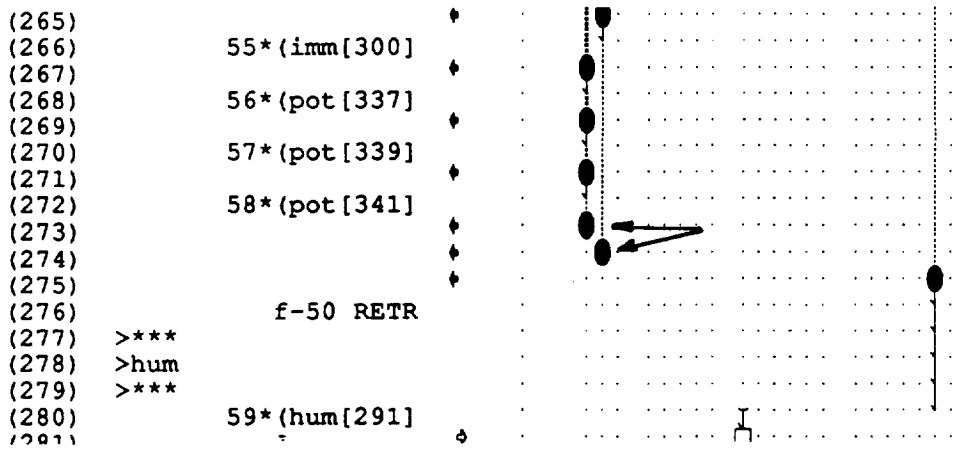

Figure 8

visualization is particularly useful during the development/debug and analysis phases of the expert system application building process. This provision of a gestalt not previously available allows the user to more easily identify and isolate problem areas then would be possible using detailed traces alone.

Acknowledgement

The authors would like to thank Dr. Dan Rochowiak of the Johnson Research Center at the University of Alabama in Huntsville for his application of ATS displays to the ECLSS potable water subsystem demonstrational prototype and for his many discussions of that effort.

References

1. Barr, A. and E. A. Feigenbaum, *The Handbook of Artificial Intelligence - Volumes I-III,* William Kaufmann, Los Altos, 1982.

2. Barr, A., P. R. Cohen, and E. A. Feigenbaum, *The Handbook of Artificial Intelligence - Volume IV,* Addison-Wesley, New York, 1989.

3. Bentley, J. L. and B. W. Kernighan, "A System for Algorithm Animation - Tutorial and User Manual," Computing Science Technical Report No. 132, AT&T Bell Laboratories, Murray Hill, NJ, 1987.

4. Bridges, S. M., "A Theory and Strategy for Justification Production by Expert Planning Systems," PhD Dissertation, University of Alabama in Huntsville, 1989.

5. Brown, M. H., *Algorithm Animation,* The MIT Press, Cambridge, Mass., 1987.

6. McKeown, K. R., *Text Generation,* Cambridge University Press, Cambridge, 1985.

7. Model, M. L., "Monitoring System Behavior In a Complex Computational Environment," Tech. Rep. CSL-79-1, XEROX PARC, 1979.

8. Schunk, R. G. and W. R. Humphries, "Environmental Control and Life Support Testing at The Marshall Space Flight Center," Proc. 17th Intersociety Conference on Environmental Systems, 1987.

9. Selig, W. J. and J. D. Johannes, "Reasoning Visualization in Expert Systems - The Applicability of Algorithm Animation Techniques," Proc. Third International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1990. In Preparation

10. Selig, W. J. and J. D. Johannes, "Towards Reasoning Visualization in Expert Systems," Proc. Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, pp. 1001-7, 1989.

11. Stasko, J. T., "TANGO: A Framework and System for Algorithm Animation," PhD Dissertation, Brown University, 1989. Also as Tech. Rep. CS-89-30

12. Tanimoto, S. L., *The Elements of Artificial Intelligence: An Introduction Using LISP,* Computer Science Press, Maryland, 1987.