Integrated Network Management, II
I. Krishnan & W. Zimmer (Editors)
Elsevier Science Publishers B.V. (North-Holland)

Automated Knowledge Acquisition from Network Management Databases

R. M. Goodman[a] and H. Latin[b]

[a]Department of Electrical Eng., California Institute of Technology, Pasadena, CA 91125

[b]Systems Technology, Pacific Bell, Concord, CA 94520

Abstract

In this paper we describe a technique for automatically learning rules from network management databases. Our motivation for this is to alleviate the knowledge acquisition bottleneck inherent in developing expert systems for integrated network management. We outline our ITRULE rule induction algorithm, show how useful rules can be extracted from trouble ticket and alarms databases, and show how these rules can be automatically loaded into a standard expert system shell, thus virtually instantly producing a prototype expert system.

1. INTRODUCTION

As part of an on-going collaborative project (NETREX) between Caltech and Pacific Bell aimed at producing real-time expert system modules [1], we have been faced with the problem of developing rules via the traditional techniques of knowledge acquisition. This is a very time consuming process in terms of human resources, particularly expert availability. We have therefore investigated various automated knowledge acquisition techniques aimed at speeding up this process. In particular we have been concerned with the automated induction of rules from network management databases. These databases include trouble ticket databases, alarms databases, and topology databases. This area of learning from examples is referred to as machine learning, and a number of statistical and neural network algorithms exist that enable rules or correlations between data to be learned. In this paper we outline our approach to automated rule induction via our own algorithm ITRULE (Information Theoretic Rule Induction). The ITRULE algorithm possesses a number of significant advantages over other algorithms in that the rules that are generated are ranked in order of informational priority or utility. It is thus an easy matter to directly load the rules into a standard expert system shell (such as NEXPERT), utilize an inferencing scheme based on these rule priorities, and have a working expert system performing inference in a matter of minutes. We have implemented the ITRULE suite of programs on a number of platforms (Sun, Mac, PC), and linked these into a number of expert system shells (NEXPERT, KES). This approach means that the expert system developer can "instantly" generate and run a tentative expert system with little domain expertise. This "bootstrap" expert system can then be used to refine the rules in conjunction with the domain expert in a fraction of the time of traditional "cold" question and answer knowledge acquisition techniques.

2. RULE INDUCTION AND UTILITY MEASURES

The motivation for defining rule utilities arises from the desire to provide a more quantitative and rigorous theory for both rule induction (acquisition and learning of rules) and rule-based inference (reasoning using the learned rules).

Let us consider the problem of induction first. The desire to transform such databases into a condensed set of rules demands as a prerequisite the ability to rank rules in some manner. Hence we arrive at the need for a clearly defined rule-preference measure or a rule utility.

Previous work in this area has focussed on either qualitative rule modeling (e.g., Michalski [2]) or else has artificially restricted the nature of the solution to that of a classification or decision tree, usually via the ID3 algorithm or its variants (e.g., Quinlan [3]), or the CART algorithm [4].

It is worth emphasizing that our approach is fundamentally different from these classification and tree-oriented approaches. The problem we are trying to solve is that of *generalized rule induction*. In fact classification is a special case of our approach. In essence, generalized rule induction involves the induction of rules relating *all* attributes and is not just restricted to symptom-class rules. In this manner a model using such a rule set can in principle reason towards any prescribed goal or attribute from any given initial conditions. We feel that this level of generality is a requirement in many domains where the system must exhibit common-sense reasoning. Classification-oriented rule sets (including tree structured rule sets as a special case) do not possess this generality. In domains where the inputs (observable data) and outputs (the goal or class) do not change, a predetermined hierarchy of rules is an appropriate solution, and indeed decision trees have proved remarkably efficient for many such problems [5].

However in some domains, such as network management, all the evidence or attribute variables may not always be observable, i.e., we can only observe a subset of the attributes for any given problem instance. For example, it may not be possible to conduct a particular device test because of the network fault, and we may have to "reason around" the unavailable test. In these circumstances decision trees are severely limited. The ability to deal with such variable inputs is a reasonable requirement of an intelligent reasoning system. In summary, our approach to rule induction can be viewed as a more general and direct method than previous work in this area.

The second motivation for the idea of rule utilities originates from the desire to improve the modeling of rule-based *inference*, in particular the aspect of conflict-resolution or control. Conflict resolution techniques as previously developed in the literature have been primarily qualitative rather than quantitative in nature. While this approach is well motivated it leads to a brittleness and domain-specificity when applied to real problems. By defining a fundamental measure of rule utility, which is in some sense theoretically correct, we can implement a very general and implicit control scheme that automatically incorporates both forward and backward chaining as implemented in most expert system shells.

3. THE ITRULE INFORMATION THEORETIC APPROACH

Consider the problem of quantifying the utility or goodness of a particular rule, where a rule is considered to be of the form

$$\text{If } \mathbf{Y} = y \text{ then } \mathbf{X} = x \text{ with probability } p$$

For our purposes we may treat $\mathbf{Y}$ and $\mathbf{X}$ as discrete-valued or categorical attributes. Expressions of the form $\mathbf{Y} = y$ are attribute-value assignment statements, where y is an element of $\mathbf{Y}$'s alphabet. Note that $\mathbf{Y} = y$ can be a conjunctive Left Hand Side (LHS), i.e. A = a & B = b. The probability p is simply the conditional probability $p(x|y)$, that is, the probability of the rule right hand (RHS) side given that the rule left hand side is true. While acknowledging that other parameters such as certainty factors or likelihood ratios are often used, we consider the conditional probability as the most basic and well-established rule-belief parameter. Note that we are primarily interested in probabilistic rules, that is, there is an implicit uncertainty in the rule."Factual rules" where $p = 0 or 1$ are a special case of the theory. This belief parameter p alone is not sufficient to measure a rule's utility. Clearly the utility must be influenced by such factors as the probability $p(\mathbf{Y} = y)$ which reflects the average probability that the left-hand side of the rule will evaluate to true, i.e., that the rule will fire.

Let us adopt an information theoretic approach to this problem and define the information which the *event* y yields about the variable $\mathbf{X}$, say $f(\mathbf{X};y)$. Based on the requirements that $f(\mathbf{X};y)$ is both non-negative and that its expectation with respect to $\mathbf{Y}$ equals the Shannon average mutual information $I(\mathbf{X};\mathbf{Y})$, Blachman [6] showed that the *only* such function is the j-measure, $j(\mathbf{X};y)$. More recently we have shown that $j(\mathbf{X};y)$ possesses unique properties as a rule information measure [7]. In general the j-measure can also be interpreted as a special case of the cross-entropy or binary discrimination (Kullback [8]) between the LHS and RHS probability distributions. We further define $J(\mathbf{X};y)$ as the *average* information content where $J(\mathbf{X};y) = p(y).j(\mathbf{X};y)$. $J(\mathbf{X};y)$ simply weights the instantaneous rule information $j(\mathbf{X};y)$ by the probability that the left-hand side will occur, i.e., that the rule will be fired. A rule with high information content must be both a good predictor and have a reasonable probability of being fired, i.e., $p(y)$ can not be too small.

The J-measure is then the basis of our rule utility measure in that it defines the average number of bits of information in the Shannon sense that we obtain when the rule is "fired". The final step in developing the general utility measure is to incorporate a cost term $c(y)$. This allows us to incorporate the subjective cost of measuring the LHS variables. The final utility measure used is then $U(xy) = J(\mathbf{X};y) - c(y)$. This utility measure can then be considered a "goodness" measure relative to the "best" rule.

4. THE ITRULE ALGORITHM

We have developed a suite of algorithms collectively called the ITRULE algorithm [9] which uses the J-measure to derive the most informative set of rules from an input data set. The algorithm takes as input a sample set of features vectors where each of the features is discrete-valued.

As output the algorithm produces a set of K probabilistic rules, ranked in order of decreasing utility. The parameter K may be user-defined or determined via statistical significance tests based on the size of the sample data set available.

For each right-hand side of interest the algorithm begins by first considering first-order general rules, i.e., rules with a single variable on the left-hand side. It then proceeds to search for more informative higher order rules which are specialized versions of the original rule, by adding terms to the left-hand side in a depth-first manner. The search is made highly efficient by using information theoretic criteria to guide the search and constrain the search space. A ranked list is kept so that new rules may be compared with

the utility of the rule currently in the Kth position in the list. Rules which exceed this threshold are inserted in the list, those that are not are rejected.

ITRULE uses information theoretic search criteria and small sample statistics to very quickly come up with a candidate set of rules. Furthermore these rules are directly output into the format of the expert system shell in use (NEXPERT, KES in our case) to give an almost instant prototype expert system.

The ITRULE software also has facilities for both manual and automated rule editing, in order to optimize the set of rules to a number of user selected criteria, for example: accuracy, simplicity, lowest cost, self-consistency, etc. In addition, the ITRULE software implements a number of routines for processing continuous attribute data (such as time) into categorical data as required by the algorithm. These routines are manual, information theoretic, statistical, and neural network inspired. Thus for example given a particular time attribute, ITRULE can indicate that it makes sense to categorize the attribute's values into bins of $\leq 5mins, \leq 30mins, and \geq 30mins$.

5. RULE-BASED INFERENCE WITH UTILITY MEASURES

The utility measure associated with each rule is then used in the expert system shell to perform conflict resolution. For example in NEXPERT a priority value is associated with each rule, and after collecting candidate rules which can fire (their LHS's are true) the rule with the highest priority is actually fired. ITRULE associates two utility values with each rule: one for forward chaining based on the j-measure, and one based on the J-measure for backward chaining. In this way ITRULE provides a complete "instant" prototyping system for an expert system shell.

6. EXAMPLES USING NETWORK MANAGEMENT DATABASES

We now describe some of the applications of ITRULE to analyzing network management databases within the Caltech - Pacific Bell collaboration. In Figure 1. we show an ITRULE analysis of a Pacific Bell Network Management trouble ticket database. The database logs trouble reports on a large (25,000 terminals) distributed data network. The number of trouble reports on such a large network run into the hundreds per day. In this case the ITRULE analysis is being used as a data summarizing tool and the rules are being very effectively used for the benefit of humans trying to understand the way in which troubles are closed out, and how long it takes to do so. The top half of figure 1. shows a small portion of the trouble ticket database (the actual historical database is of course huge). The trouble database (and the input format that ITRULE needs) is in the form of a spreadsheet where each line is a trouble record. The attributes (or fields) of the database refer to the equipment ID, the tests that were made in isolating the trouble, who the trouble was referred to, the final close out category, and who fixed the fault. The middle portion of Figure 1. shows the attributes which correspond to the data fields above them. Note that most attributes are categorical, apart from "time - down" which in the raw data is a continuous variable. This attribute was manually categorized by Trouble Center experts who were interested in resolution times of greater or less than one hour. The ITRULE process was directed to produce rules about how long it took to close out troubles, and who actually fixed the problem. The lower part of Figure 1. shows a portion of the ITRULE analysis, giving the 13 most informative rules extracted from the data. The first numeric entry in each rule line is the probability p, the

"correctness" of the rule. The last entry is the utility measure (equivalent to the relative J-measure in this case). The rules reveal the fundamental statistics of trouble closeouts in a very human readable way. For example, rule 2 shows that if faults are software based they almost always fixed by the highly skilled testers, alternatively, rule 8 shows that the simpler one terminal call-fail problems are often fixed by the less technically specialized trouble call screeners (usually by resetting the control unit - which is revealed in a rule further down the list). Rule 11 reveals that if there are hardware problems with 4540A terminals then these take over an hour to fix (they involve a referral and a site callout). Note that within the context of this trouble ticket domain rule 1 is an "obvious" rule. When troubles have been referred to the OCS group, then if that group solves the problem they also always fix the problem. The ability of ITRULE to extract these "obvious" rules is of great benefit when bootstrapping a prototype expert system. One of the biggest problems with the conventional technique of interviewing experts is that they have difficulty adjusting themselves to describe the elementary rules of their domain, that is the $2 + 2 = 4$ type rules. These rules are therefore often laborious to extract, however, the ITRULE approach easily finds these rules.

We now show a more complex example in which ITRULE is used to analyze a time varying network alarms database. The objective is to automatically develop rules for a higher level expert system whose output is a (real time varying) prioritized list of the most important network alarms. This alarms list is then presented to the network administrators and gives a real time picture of the most important problems in the network. In order to achieve this we need an expert module that understands the relation between line alarms, so that when presented with the latest alarms evidence it can predict the likelihood of other related alarms occurring, and thus compute a postulated severity measure for each alarm that has occurred. For example, if we can learn that the occurrence of a particular sequence of alarms usually indicates that a serious outage may occur in the near future, thus affecting a large number of users, we can assign a high priority to fixing the detected alarms. The top part of Figure 2. shows a small portion of the raw alarms database. Note the volume of alarms, approximately one every 30 seconds - it is impossible for any human to get an overall picture of the network from such volumes of data, or to understand the relation between alarms. This raw data is then preprocessed into a the spreadsheet form (as in Figure 1.) suitable for ITRULE input. To do this we consider the behavior of each network line over a period of several hours. The attributes of the problem are than the number of alarms, the number of stations with alarms, and the actual alarm types, etc. Each example (row) in the input spreadsheet (not shown) thus refers to the history of a particular line in terms of the type of alarms experienced over the observation period. The lower half of Figure 2. lists the attributes used, and gives an ITRULE analysis of the alarms data, showing a portion of the most important rules. Note that we are not interested in human readability here - just in automatic rule production. The binary value 1 indicates the occurrence of a particular alarm. Note again the "obvious rule", rule 1, which indicates that if a CU fail occurs then a CU restoral will occur soon after.

Figure 3. shows a NEXPERT display of the alarm rules. These rules were generated by ITRULE as in Figure 2. and saved in a NEXPERT loadable file. The alarm expert system shown was generated in a few minutes, and is ready to perform inference. The top screen in Figure 3. shows a portion of the rule network, showing the complex alarm inter-relations. The lower left portion of Figure 3. shows an overview of the several hundred rules generated by ITRULE. The lower right shows a rule being displayed in the NEXPERT rule editor.

7. CONCLUSIONS

The ITRULE process we have outlined provides a valuable tool for automated knowledge and rule induction in the prototyping of expert systems. In particular ITRULE is able to produce meaningful alarm correlations, and could be used to identify higher-level "events" consisting of sequences of alarms. Using ITRULE the knowledge engineer can quickly generate a fully functioning alarms expert system, to significantly improve interaction and knowledge acquisition with the expert. We continue to develop ITRULE applications for real time network management.

8. ACKNOWLEDGMENTS

This work is supported in part by Pacific Bell, and in part by the Army Research Office under Contract No. DAAL03-89-K-0126.

9. REFERENCES

1. R. M. Goodman, J.W. Miller, P. Smyth and H. Latin, 'Real Time Autonomous Expert Systems in Network Management,' *Proceedings of the first IFIP International Symposium on Integrated Network Management*, Boston, May14-17, 1989.

2. R. S. Michalski and R. L. Chilausky, 'Learning by being told and learning from examples', *International Journal of Policy Analysis and Information Systems* **4**, pp.125-161, 1980.

3. J. R. Quinlan, 'Induction of decision trees,' *Machine Learning*, vol. 1, 81–106, 1986.

4. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and regression trees*, Belmont, CA: Wadsworth, 1984.

5. R. M. Goodman and P. Smyth, 'Decision tree design from a communication theory standpoint,' *IEEE Trans. Information Theory*, vol 34, no. 5, 979–9 94.

6. N. M. Blachman, 'The amount of information that y gives about X,' *IEEE Transactions on Information Theory*, **IT–14** (1), 27–31, 1968.

7. P. Smyth and R. M. Goodman, "The information content of a probabilistic rule," submitted to *IEEE Trans. on Information Theory*, June 1990.

8. S. Kullback,*Information Theory and Statistics*, New York: Wiley, 1959.

9. P. Smyth and R. M. Goodman,'Deriving rules from databases — the ITRULE algorithm,' to be published *IEEE Trans. on Knowledge and Data Engineering*, 1990.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	TROUBLE TICKET DATABASE																			
2	B0	428	40	4540A	?	CUF	ALLDEV	?	OK	?	?	?	?	?	TERM	CU	RC	CS	CLIENT	underlhr
3	B0	444	C1	TC174	?	CFL	ONE_DEV	?	OK	?	?	?	?	?	TERM	CU	RC	CS	CS	underlhr
4	B0	11	C4	4540A	NO_CO	?	?	OK	NOK	?	?	?	PCO	?	NO_TRBL	?	?	?	?	overlhr
5	B0	02D	C1	TC174	NO_CO	CUF	?	?	?	?	?	?	PCO	?	NO_TRBL	?	?	?	?	overlhr
6	B0	43E	40	4540B	NO_CO	?	?	?	?	?	?	?	PCO	?	NO_TRBL	?	?	?	?	overlhr
7	B0	04D	40	TC174	NDE_DWN	CUF	ALLDEV	?	?	?	?	?	TESTER	?	TELCO	LINE	?	?	?	underlhr
8	B0	37	40	4540A	?	CUF	ALLDEV	?	NOK	NOK	?	?	TESTER	OCS	TELCO	LINE	?	TESTER	PCO	overlhr
9	B0	20C	40	$CC	?	CLF	ALLDEV	?	?	?	?	?	TESTER	?	NO_TRBL	?	?	?	?	overlhr
10	B0	46E	C1	4540A	?	CUF	ALLDEV	?	?	?	?	?	OCS	?	TERM	CU	HRDWR	OCS	OCS	overlhr
11	B0	61E	C3	4540A	S/R	?	?	?	NOK	?	?	NOK	TESTER	?	TCU_SWR	CU	SFTWR	TESTER	TESTER	underlhr
12	B0	65F	C1	TC174	INTRMNCU	?	?	?	?	?	?	?	OCS	?	TERM	D/S	LOOPBK	OCS	OCS	overlhr
13	B0	65F	40	TC174	?	CUF	ONE_DEV	?	?	?	?	?	OCS	?	DATA_SET	D/S	HRDWR	OCS	OCS	overlhr
14	B0	65F	40	TC174	INTRMNCU	?	?	?	?	?	?	?	OCS	?	TERM	CU	HRDWR	OCS	OCS	overlhr
15	B0	666	C1	4540A	?	CUF	ALLDEV	NOK	?	NOK	?	?	TESTER	?	DATA_SET	D/S	CONFIG	TESTER	CLIENT	underlhr
16	F0	18	C1	4540A	?	CUF	ALLDEV	?	NOK	NOK	?	?	PCO	?	TELCO	TCKR	HRDWR	PCO	PCO	overlhr
17	D0	58	40	4540A	ALL_DWN	CFL	SOME_DEV	?	NOK	?	OK	?	TESTER	?	TCU_SWR	LINE	SFTWR	TESTER	TESTER	underlhr
18	D0	439	40	3274	TRM_DWN	?	?	?	?	?	?	?	OCS	?	TERM	CU	HRDWR	TESTER	OCS	underlhr
19	F0	04C	40	5546I2	?	LMF	?	?	?	?	?	?	TESTER	?	TCU_SWR	LINE_MOD	SFTWR	TESTER	TESTER	underlhr
20	F0	51	40	4540A	?	CUF	ALLDEV	?	?	?	?	?	TESTER	?	TERM	CU	HRDWR	OCS	OCS	underlhr
21	F0	06A	40	4540A	ALL_DWN	?	?	?	NOK	OK	OK	?	TESTER	?	TCU_SWR	LINE	SFTWR	TESTER	TESTER	underlhr
22	F0	06C	40	4540A	ALL_DWN	CUF	ALLDEV	?	NOK	NOK	?	?	TESTER	?	NO_TRBL	LINE	?	?	?	underlhr
23	F0	06C	40	4540A	S/R	CUF	ONE_DEV	?	OK	?	?	NOK	TESTER	?	MUX	MUX	RESET	TESTER	NCG	underlhr
24	F0	264	40	4540A	ALL_DWN	CUF	ALLDEV	?	NOK	?	?	?	TESTER	PCO	TELCO	DSX_JMPR	HRDWR	TESTER	PCO	overlhr
25	F0	268	40	4540A	?	CUF	ALLDEV	?	NOK	NOK	?	?	TESTER	OCS	TERM	CU	HRDWR	OCS	OCS	overlhr
26	F0	41E	40	TC174	?	CFL	ALLDEV	?	NOK	?	?	?	TESTER	OCS	TERM	CU/STN	HRDWR	OCS	OCS	overlhr
27	F0	60D	C1	TC174	?	CUF	?	?	NOK	NOK	?	?	TESTER	OCS	TERM	CU/STN	HRDWR	OCS	OCS	overlhr
28	D0	428	40	4540A	?	CUF	ALLDEV	?	OK	?	?	?	?	?	TERM	CU	RC	CS	CLIENT	underlhr
29	D0	444	C1	TC174	?	CFL	ONE_DEV	?	OK	?	?	?	?	?	TERM	CU	RC	CS	CS	underlhr
30	B0	11	C4	4540A	NO_CO	?	?	OK	NOK	?	?	?	PCO	?	NO_TRBL	?	?	?	?	overlhr
31	B0	02D	C1	TC174	NO_CO	CUF	?	?	?	?	?	?	PCO	?	NO_TRBL	?	?	?	?	overlhr
32	B0	43E	40	4540B	NO_CO	?	?	?	?	?	?	?	PCO	?	NO_TRBL	?	?	?	?	overlhr
33	B0	04D	40	TC174	NDE_DWN	CUF	ALLDEV	?	?	?	?	?	TESTER	?	TELCO	LINE	?	?	?	underlhr
34	B0	37	40	4540A	?	CUF	ALLDEV	?	NOK	NOK	?	?	TESTER	OCS	TELCO	LINE	?	TESTER	PCO	overlhr
35	B0	20C	40	$CC	?	CLF	ALLDEV	?	?	?	?	?	TESTER	?	NO_TRBL	?	?	?	?	overlhr
36																				
37																				
38																				
39																				
40																				

```
ATTRIBUTES:

A  NODE       B  LINE       C  STATION    D  Dev_type    E  SYMPTOM
F  DEV_STAT   G  DEV_no     H  AL_ST      I  RC_CU       J  IL_XIL
K  DL_XDL     L  BCT        M  Refer1     N  Refer2      O  Close_Cat
P  Soln_Dev   Q  Soln_Actn  R  Solved_By  S  Fixed_By    T  time_down

RULES:

Rule Format:  IF lhs1 (AND) lhs2....  THEN rhs

Stats:  prob(rhs/lhs), j-measure, J-measure, Utility

   IF                    AND                   THEN

 1 Solved_By OCS                               Fixed_By OCS       0.967 0.955 0.379 1000
 2 Soln_Actn SFTWR                             Fixed_By TESTER    0.912 1.658 0.347 916
 3 Close_Cat TELCO                             Fixed_By PCO       0.879 1.897 0.285 753
 4    DEV_no ALLDEV   time_down underlhr       Fixed_By CLIENT    0.589 0.863 0.161 425
 5   SYMPTOM NO_CO                             time_down overlhr  0.923 0.704 0.161 424
 6    Refer1 TESTER   Fixed_By TESTER          time_down underlhr 0.937 0.836 0.143 378
 7    DEV_no ONE_DEV    Refer1 TESTER          Fixed_By NCG       0.690 1.788 0.122 321
 8  DEV_STAT CFL        DEV_no ONE_DEV         Fixed_By CS        0.690 1.788 0.120 317
 9 Solved_By TESTER                            time_down underlhr 0.763 0.210 0.094 247
10   SYMPTOM ALL_DWN  Soln_Dev LINE             time_down underlhr 0.929 0.629 0.077 204
11  Dev_type 4540A   Close_Cat TELCO           time_down overlhr  0.934 0.558 0.064 167
12 Solved_By CS                               time_down underlhr 0.900 0.531 0.055 145
13   SYMPTOM NDE_DWN                          time_down underlhr 0.846 0.309 0.025 65
```

Figure 1. Trouble ticket analysis using Itrule

RAW ALARMS STREAM:

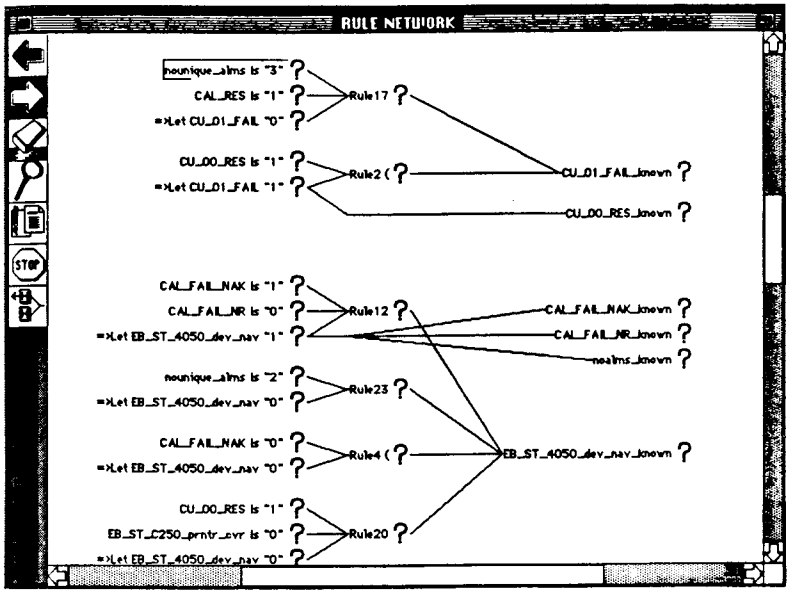
DATE	TIME	NODE	ID	CONDITION
10/28	10:19:49	D2	T061	POL RES 41507
10/28	10:19:40	D0	T049	LINE FAIL 040
10/28	10:19:40	D0	T000	LINE CONDITION ERROR 040 02
10/28	10:19:40	D1	T049	LINE FAIL 270
10/28	10:19:40	D1	T000	LINE CONDITION ERROR 270 02
10/28	10:19:40	D2	T049	LINE FAIL 457
10/28	10:19:40	D2	T000	LINE CONDITION ERROR 457 02
10/28	10:19:34	D0	T013	EBCDIC STATUS *4050* 01914
10/28	10:19:34	D0	T008	CAL FAIL NAK 01914
10/28	10:19:00	D2	T000	BINS 0549,EMBN 1518,MG/M 1554,MRAT 0021
10/28	10:18:00	D3	T000	BINS 0602,EMBN 1548,MG/M 1371,MRAT 0022
10/28	10:17:45	D3	T010	CAL RES 64A00
10/28	10:17:00	D1	T000	BINS 0575,EMBN 1531,MG/M 2088,MRAT 0034
10/28	10:16:45	D0	T049	LINE FAIL 02A
10/28	10:16:44	D0	T052	LINE OUT 02A
10/28	10:16:31	D2	T010	CAL RES 45710
10/28	10:16:19	D2	T010	CAL RES 4570A
10/28	10:15:54	D2	T010	CAL RES 45702
10/28	10:14:18	D3	T011	CU 00 FAILED 64A
10/28	10:14:00	D0	T000	BINS 0522,EMBN 1550,MG/M 2024,MRAT 0030

ATTRIBUTES:

#stats.w/alms	#alms	#unique.alms
CAL.RES	CU.01.FAIL	CU.00.RES
EB.ST:C250-prntr.cvr	EB.ST:4050-dev.nav	LINE.MOD.INOP
AHP.E(FACS).LINK.OK	T060	T015
POL.RES	LINE.OUT	EB.ST:40C2-tx.err
CAL.FAIL.DC	CAL.FAIL.NAK	INTR.NODE.LNK.FAIL
T018	CAL.FAIL.NR	INTR.NODE.LNK.OK
T048	LINE.FAIL	POL.FAIL.DC

RULES:	IF		AND		THEN		UTILITY
1	CU.01.FAIL	1			CU.00.RES	1	1000
2	CU.00.RES	1			CU.01.FAIL	1	857
3	CAL.FAIL.NAK	0	CAL.FAIL.NR	0	CAL.RES	0	710
4	CAL.FAIL.NAK	0			EB.ST:4050-dev.nav	0	645
5	EB.ST:4050-dev.nav	1			CAL.FAIL.NAK	1	597
6	CAL.RES	0	CU.00.RES	1	CAL.FAIL.NAK	0	570
7	#alms	under10			#stats.w/alms	1	556
8	CAL.RES	0	EB.ST:4050-dev.nav	0	CAL.FAIL.NAK	0	549
9	CAL.RES	0	INTR.NODE.LNK.OK	0	#unique.alms	2	491
10	CAL.RES	0	LINE.MOD.INOP	0	#unique.alms	2	489
11	#stats.w/alms	1	#unique.alms	2	#alms	under10	476
12	CAL.FAIL.NAK	1	CAL.FAIL.NR	0	EB.ST:4050-dev.nav	1	457
13	#alms	under10			#unique.alms	2	455
14	#stats.w/alms	>1	CAL.FAIL.DC	1	#unique.alms	4+	441
15	#stats.w/alms	1	CAL.FAIL.NAK	0	#alms	under10	416
16	#stats.w/alms	1	EB.ST:4050-dev.nav	0	#alms	under10	413
17	#unique.alms	3	CAL.RES	1	CU.01.FAIL	0	413
18	CAL.RES	1	EB.ST:4050-dev.nav	0	CAL.FAIL.NR	1	410
19	#unique.alms	2			#stats.w/alms	1	385
20	CU.00.RES	1	EB.ST:C250-prntr.cvr	0	EB.ST:4050-dev.nav	0	382

Figure 2. Alarms analysis using Itrule



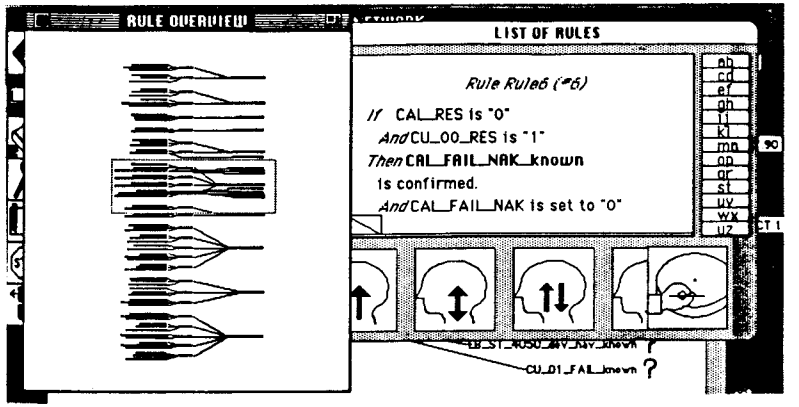


Figure 3. Rules automatically loaded into expert system shell