1.0 Introduction

Artificial intelligence (AI) is a broad subject area that spans the range of topics from natural language query languages for end users to state of the art research projects currently being conducted that mimic human reasoning. In this paper we will discuss the use of rule based expert systems that apply the concepts of approximate reasoning and a proposed implementation for extending the SAS language to incorporate these features. In particular, we will discuss:

- the nature of export knowledge and its representation,
- fuzzy logic,
- rule evaluation, and
- the subroutines and functions required to implement these facilities for SAS users.

While not extensive, the proposed implementation can provide the fundamentals of rule based expert reasoning to SAS users.

2.0 Knowledge Representation

Rather than pursue an abstract tutorial about the nature of knowledge and its representation, it is much easier to consider a simple data selection question typical of those presented to users every day. Consider the problem presented to a user who wishes to select records from a SAS client data base that represent his companies big clients. To identify these clients, the user must develop boolean selection criteria to select the desired records. In this example, the user might decide to identify big clients based on annual sales volume, SAS variable YR_SALES. Assuming that one-half million dollars is selected as the threshold value, the SAS subsetting IF statement would be:

\[
\text{IF YR_SALES} > 500000;\]

500000; to select the companies big clients.

Unfortunately, this IF statement might exclude a large number of clients that were almost big, i.e., just a little less than one-half million in annual sales, that senior management might still consider to be both big and important. Using boolean selection criteria, such over-sights can plague the casual user who does not have a detailed understanding of the underlying data that would assist them in selecting the threshold value. Moreover, the whole concept of boolean selection criteria may appear foreign to casual users who normally discuss their logical criteria in terms or words like almost, close, and maybe.

To represent our knowledge about any process, we must analyze and diagram the series of questions that we ask ourselves when we evaluate the process. More simply stated, if someone asked your advice in how to identify a big client, what steps would you suggest? These steps could be used to generate a set of one or more rules that could be used to identify big clients. Expert system developers use the name policies to denote these sets of rules.

Although the rule proposed to identify big clients is quite simple, it depends on the analyst's sense of what the adjective big means in terms of annual sales volume. In reality, such values are more adaptive rather than a constant in nature. If the analyst fails to identify a reasonable set of clients using one value a filter, then the analyst may reduce the threshold values and repeat the data selection. By reducing the threshold values, the analyst leaves the realm of boolean logic (i.e., clearly true or false) and enters the realm of hunch playing that distinguishes the expert. Simply defined, the expert uses experience and intuition to make high probability assertions based on incomplete or vague data. Rule based expert systems use the notion of fuzzy logic to represent the gray area of rule evaluation where the rule is not clearly true or false.

3.0 Fuzzy Logic

After the expert has developed policies (i.e., sets of rules) that describe their approach to analyzing the information, they must then develop a vocabulary that defines adjectives like good and bad for metrics evaluated by the rules. Although these definitions initially appear perplexing to the analyst, they can be defined in a straightforward manner as we will see in the following paragraphs.

The first step in defining an adjective like big for a client is to determine the minimum value for which the assertion is always true. For this example, let's use a value of one-half million dollars to identify large clients. If another user would have chosen a different value, then the prior sentence serves as a good example of the disparity of expert knowledge and opinion. Conversely, we must also choose a minimum value below which the assertion that a client is big is always false. For this minimum value, the author selected one hundred thousand dollars. Thus we have established the following traditional boolean tests for client size:

\[
\begin{align*}
\text{IF YR_SALES} & > 500000 \text{ THEN BIG = TRUE} \\
\text{IF YR_SALES} & < 100000 \text{ THEN BIG = FALSE}
\end{align*}
\]

The two IF statements shown above clearly define the gray area in our logic. That is, neither IF statement is true between the sales off one hundred thousand and one-half million dollars. Fortunately, fuzzy logic allows us to represent this gray area between values that are completely true or false.

It is useful to redefine the two IF statements shown above as a graph. The Y-axis is the truth of the assertion (i.e., 0 or 1) and the X-axis is the range of values over which the adjective is defined. If we choose to define big for sales over the range of 0 to one million dollars, then we would obtain the graph shown in Figure 1.
Clearly, between the values of one hundred and one-half million dollars, there is some probability that the assertion that the client is big is true. Using the fuzzy logic, we can define this probabilistic truth function over the range of values for which the assertion is neither absolutely true or false. Although a wide variety of shapes (normal distributions, exponentials, linear functions, spline curve fits... etc.) could be used to define this truth function, consider the simple linear function shown in Figure 2.

Using the truth function shown in figure, we can evaluate the probabilistic truth in the policies (i.e., sets of rules) we have defined for evaluating a process or selecting data. Another interesting feature of these truth functions is the ability to modify them to represent the sense of words that qualify the vocabulary adjectives like quite, very, or almost. Numbers in the range of a to 1 have interesting properties when they are exponentiated. For exponents greater than 1, the result of the exponentiation is a smaller value and for exponents less than 1, the result of the exponentiation is larger than the number raised to the power. However, the key is that the result of the exponentiation is still defined over the range of a to 1. Thus we may define words (called hedges) that raise the specified truth functions to powers. For instance if we define very as an exponent of 2 and almost as an exponent of 1/2 (i.e., the square and square root respectively) we can qualify the previously defined truth function as is now shown in Figure 3.

Although we did not use hedges in this simple example, they are language features which allow us to evaluate finer shades of reasoning in the policies that we develop.

4.0 Rule Evaluation

Once the content of the rules and the vocabulary for evaluating metrics have been defined, formal policies (i.e., sets of rules) can be specified for the analysis. Consider rules of the form:

IF variable IS truth-function THEN variable IS truth-function

As is the case with any IF statement, a rule is comprised of two clauses. They are the assertion and the dependent clauses. The assertion in the prototype rule shown above involves only a single variable and truth-function (i.e., vocabulary word) pair. However, any number of these pairs may appear in the assertion clauses related by OR or AND operators. As is the case with a boolean IF statement, there may only be one variable specified in the dependent clause. The principal difference between the prototype rule shown above and the boolean IF statement with which we are most familiar is that the variable in the dependent clause of the rule will be assigned the measure of truth determined in the assertion clause rather than a simple true or false outcome.

Although we have defined a vocabulary word for evaluating device activity (i.e., big for clients) we have not yet defined a vocabulary word for evaluating the results of the rule. For that purpose a definition of the adjective positive is shown in Figure 4. Positive is defined as a simple linear function over the range of 0 to 100, that is the probability of truth expressed in the assertion expressed as a percent.
Clearly, sequential evaluation does not mimic the apparently complex. Often, the expert neglects to add special case rules when they initially define their expert system. When they are presented with an outcome from their expert system that does not agree with their own opinion, they often find that the discrepancy is a result of their failure to include the wealth of rule exceptions that comprise a key part of their expert knowledge. When such rules must be added to an existing policy, they may simply be inserted at any convenient place without concern of confusing order dependent outcomes.

5.0 Proposed SAS Implementation

The rule based expert system implementation for SAS discussed in this paper requires a structure parallel to SAS's program data vector to be developed and the definition of five subroutines and functions. The subroutines and functions are:

- vocabulary word definition subroutine,
- vocabulary word graphic display,
- forward truth function,
- inverse truth function, and
- policy evaluation subroutine.

In addition, there are a number of other non-essential routines that would make the use of the implementation somewhat more convenient. All of these data structures and routines are discussed in the following sections.

5.1 Vocabulary Data Vector (VDV)

The vocabulary data vector is a parallel structure to the existing SAS program data vector. The VDV contains:

- the name of the SAS data element for which the adjective is defined. If no SAS data element name is defined, then the vocabulary word is assumed to be a global adjective that could, for example, be used to test any SAS variable against the assertion big. If the adjective big is defined as both a global and a specific vocabulary word, then the specific adjective takes precedence for the SAS data element name for which is was specified,
- the name of the adjective, i.e., vocabulary word,
- the range over which the adjective is defined, i.e., the minimum and maximum values for which the truth function is valid.
- the shape of the truth function,
- a variable length list of X-Y pairs used to define the truth function of the specified shape.

5.2 Vocabulary Word Definition Subroutine (VOCAB WD)

The vocabulary word definition subroutine, VOCAB WD, passes a series of parameters (i.e., those defined in the prior section) to the management routine for the VDV. The subroutine returns a completion code if 1) the SAS variable name specified does not exist in the PDV, or 2) if the vocabulary word and SAS variable name specified replaced an existing entry in the VDV.

5.3 Truth Function Graphic Display (PLOTWD)

The truth function graphic display subroutine produces a box plot of the truth function specified in the subroutine call.

5.4 Forward Truth Function (TRUTHFUN)

The forward truth function evaluates the truth in an assertion for a specified SAS data element and returns a truth value between zero and one. For example, using the YR.SALES example discussed above a call to the TRUTHFUN with a value of $300,000 would return a truth value of 0.5.

5.5 Inverse Truth Function (VALUEFUN)

The inverse truth function evaluates the returns the content of a SAS data element corresponding to the truth value passed to function. For example, the POSITIVE adjective discussed above would return a value of 50 when called with a truth value of 0.5.
5.6 Policy Evaluation Subroutine (EVALPOLC)

The policy evaluation subroutine, EVALPROC, is the key routine in the implementation. To achieve the order independence discussed in section 4, the subroutine must recursively (or through some other stack management algorithm) evaluate multiple rules that either 1) have assertion clauses that are order dependent or 2) share the control of variables in dependent clauses.

5.7 Other Non-Essential Routines

While not absolute requirements, a number of other subroutines and functions would make the implementation easier to use. Principal among them are routines to store and retrieve a VDV from disk. Using such functions, two users could share sets of rules but have unique vocabularies that reflected their own opinions about the data being managed.

6.0 Remarks

Rule based expert system concepts provide the user a convenient means of managing the uncertainty associated with any decision or data selection process. In this paper, we have examined how the SAS language can be extended using expert system primitives to provide such capabilities to its users.