Statistical Science Research Report 1

Reduced Parameter Representation of Model Components

Russell Almond

Statistical Sciences, Inc.
1700 Westlake Ave, N., Suite 500
Seattle, WA 98109
almond@statsci.com

DRAFT
4/9/92

This research is supported by the GRAPHICAL-BELIEF project, NASA SBIR contract 06.04.8802.

Reduced Parameter Representation of Model Components

Russell Almond, StatSci

ABSTRACT

Compact specification of relationships between variables in a graphical model requires reducing the number of parameters in the valuation (probability potential, belief function, or utility) expressing that relationship. Finding structure (such as logical relationships) among the possible outcomes suggests reduced parameter representations of valuations. The TS-set notation, introduced here, helps describe such logical relationships. An intelligent graphical belief model construction systems should provide resources for specifying structure at a variety of levels, specifically, the outcome set level, the parameter level, the valuation level and the graphical substructure level. This paper discusses some of the issues involved in constructing small pieces of graphical models and introduces notation to help simplify the specification.

1. Rationale

Graphical models (Belief Nets, Pearl[1988], Influence Diagrams, Oliver and Smith [1990], or Graphical Belief Models, Almond[1990]) are an increasingly popular way of representing uncertain information. Graphical models combine the mathematical rigor of probability or decision analysis models with the conceptually simple representation of the graph. Furthermore, the graph supports efficient computational strategies (Lauritzen and Speigelhalter[1988], Shafer and Shenoy[1988], Almond and Kong[1991]). Finally, by decomposing a large and complex problem into small pieces, the model graph simplifies the problem of assessing probabilistic relationships between variables. Almond[1991] suggests a number of models for those small pieces.

To illustrate the task of assessment, focus on a small part of the graphical model which represents the relationship between two variables X and Y each of which can obtain three states x_0, x_1, x_2 and y_0, y_1, y_2. The information that Y supplies about X is represented by the conditional probability of X given Y. This is actually three probability distributions over X, one for each value of Y. Each probability distribution can be represented by three numbers with one constraint (they all must sum to one). Therefore a total of 6 numbers must be specified to assess the strength of this relationship. Lauritzen and Speigelhalter suggest storing all nine numbers (including the three values implied by the constraints) in an array called a *potential*.

The nine numbers comprising the potential will rarely be known exactly; therefore in addition to assessing the best guess as to the value, some estimate as to the accuracy of the guess must be made. For example, we may want to specify the variances for the six estimates describe above. As it is not always realistic to assume that the estimates are uncorrelated (Madigan and York[1991]), we will need to specify the covariances as well. Thus the simple example described above requires a total of 42 parameters, the six expected (best) values and the 36 variances and co-variances. This number will grow rapidly as the number of variables or the size of the variables (in terms of numbers of outcomes) increases.

Introducing imprecision into the problem makes this simple example even more complex. Walley[1991] argues that precisely specified probability distributions lead to precise behaviors, even when that precision is unwarranted. This leads naturally to a theory of upper and lower probabilities, representing the set of probability measures which are candidate models for a particular phenomenon. In the simple example describe above, the set of probability measures under consideration is described with an upper and lower bound for each of the nine parameters. The situation is further complicated if we wish to imposed additional constraints, such as those implied by belief functions (Shafer[1976]); these will restrict the possible values of the 18 numbers.

There are situations in which both uncertainty and imprecision about the parameters could exist. Almond[1990,1991] describes simple models for the Bernoulli and Poisson processes. In those models, information about the parameter could come from an expert—who generally has imprecise estimates—or data—which generates uncertainty about the parameter. The mixture of an imprecise (upper and lower bound) prior with a precise (probability distribution) likelihood results in a posterior state of information for the parameter which is both uncertain and imprecise. Specifying this state requires additional parameters describing the variance and co-variance structure among the now random upper and lower bounds.

The above discussion seems to imply that the graphical model problem solving paradigm (Pearl[1988], Lauritzen and Speigelhalter[1988], Shafer and Shenoy[1988], etc.) is infeasible. Fortunately, we can exploit structure within the conditional probability distribution to simplify the model specification task. In particular, we can use this structure to reduce the number of parameters which must be specified.

Pearl[1988] introduces a model he calls the *noisy and-gate* or *noisy-and*. Consider three binary variables: X_1, X_2 and Y. The occurrence of both X_1 and X_2 usually causes Y and Y occasionally occurs even when one of X_1 or X_2 is false. This model can be specified by two parameters: the probability of Y when X_1 and X_2 are both true and the probability of Y when "X_1 and X_2" is false. Thus we have reduced a $2 \times 2 \times 2$ table to a 2×2 table, halving the number of parameters which must be specified.

Shafer[1976] introduces similar models for belief functions. A basic representation of a belief function is the mass function: an ordinary probability distribution over a set of *focal elements*—sets of possible outcomes. This representation lends itself well to a mixture of logical and uncertain reasoning. Using the example of the previous paragraph, the logical relationship "X_1 and X_2 imply Y" can be represented by its truth table; call it A. The condition that X_1, X_2 and Y are unrelated is represented by the entire set of possible outcomes—the *frame of discernment*, Θ. Thus one can produce a *discounted* relationship by assigning a probability mass function on the set of sets $\{A, \Theta\}$. This mass function has only one parameter.

It is immediately apparent that Pearl's noisy-and model and Shafer's discounted relationship models are easily extended to cover other logical relationships. In fact, the focal element language developed for belief functions provides a good method for defining new noisy relationship models for specifying probability distributions. To use the focal element language to specify probability distributions merely requires restrictions on the focal elements and their masses. Similarly, the technique of focal elements can be used to build general upper and lower probability measures (of which belief functions are a special case). As this model construction is generally done before the application of Dempster's rule of combination with its questionable independence assumption, most of the usual objections to belief functions do not apply to this method (see Walley[1991] for a discussion of these issues).

This paper takes a close look at using distributions over focal elements to specify probability and belief function models with few parameters. Such reduced parameter models are easier to assess, interpret and in some cases compute with. Some common classes of these models are defined in Section 2. In order to more simply specify probability models, Almond[1990] introduces the TS-set notation; Section 3 reviews this notation. Section 4 applies the focal element technology to some simple examples. Finally, Section 5 describes how model construction software would use these techniques.

2. Classes of Reduced Parameter Models

As we are focusing our attention on a small number of variables in a relationship, we need to identify the variables on which we are currently focused. The term *frame of variables* refers to the list $(X_{i_1}, \ldots, X_{i_n})$ of variables in the relationship. As each variable, X_i, has an associated outcome space, Θ_i, the frame of variables uniquely defines a *frame of discernment* or joint outcome space, $\Theta_{i_1} \times \cdots \times \Theta_{i_n}$. Both the frame of discernment and frame of variables can be derived from the set of indices $I = \{i_1, \ldots, i_n\}$ for the variables. Thus we will carelessly use *frame* to refer to any of the three, with the understanding that the desired *frame* can be reconstructed easily from the one specified.

Using the notation of Shafer and Shenoy[1988], let the term *valuation* refer to a general numerical relationship among several variables. Thus a valuation is a function which associates values with outcomes in a frame of discernment. There are two kinds, a *simple valuation* is one that assigns a value to each element (outcome) in the frame of discernment. A *set valuation* assigns a value to each subset of the frame of discernment. A probability potential is a good example of a *simple valuation* and a *belief function* is an example of a set valuation.

There is obviously a difference between univariate frames and multivariate frames. In particular, defining a valuation over a multivariate frame does requires the specification of the interaction between the variables in the relationship. Sometimes, to clarify our thoughts about this interaction, we separate the variables into two classes the *independent variables* whose values we consider *given* or fixed (at least from the point of view of the model elicitation; distributions for the independent variables may be separately specified) and the *dependent variables* whose values are thought to depend (possibly through a backwards constraint) on the independent variables. A model whose variables are partitioned into independent and dependent values is called a *conditional* model. We will divide the set of valuations up into four classes base on the presence or absence of independent variables and whether there is one or more dependent variables in the model.

2.1 Univariate Models with no Conditionals.

This case is very simple. One simply defines a value for each outcome or set of outcomes in the frame of discernment. Let $N = |\Theta|$ be the size of the frame of discernment. Then specifying a simple valuation requires N values (possibly fewer with constraints) and a set valuation requires 2^N values. Applying tricks to reduce the parameter space here shouldn't be required unless the number of possible outcomes, N, is large.

If N is large, it may be possible to group the outcomes into several interesting sets. In a set valuation, it may be necessary to specify values only for certain *focal elements*—sets whose value is different from the default (*e.g.*, 0). Thus, for example, a belief function may be specified with just one or two values for the focal elements, the rest being implicitly zero.

Example 2.1. Red sensor. *Imagine a sensor which detects red light input. Let* $\Theta = \{\texttt{red}, \texttt{orange}, \texttt{yellow}, \texttt{green}, \texttt{blue}, \texttt{purple}, \texttt{black}, \texttt{grey}, \texttt{white}\}$. *The red sensor may report a set valuation with the following structure:*

$$\begin{aligned} \alpha_1 &= m\left(\{\texttt{red}, \texttt{orange}, \texttt{purple}, \texttt{grey}, \texttt{white}\}\right) \\ \alpha_2 &= m\left(\{\texttt{yellow}, \texttt{green}, \texttt{blue}, \texttt{black}, \texttt{grey}\}\right) \\ 1 - \alpha_1 - \alpha_2 &= m\left(\Theta\right) \end{aligned}$$

Note that we have specified the distribution over a space containing 9 elements with 3 parameters.

As it is only realistic to talk about *focal elements* with single values for simple valuations, but another trick may be of use in this situation. For a simple valuation, it is possible to think of sets of values which are constrained to have the same value. In such cases, it might be possible to assign a single to parameter to that *equivalence set* of values.

Example 2.1. Continued. *One could formulate a simple valuations from the set of focal elements above in many ways. For example, one could assign the value* α_1 *to each of the elements of the corresponding set and* α_2 *to each of the elements of that set;* $1 - \alpha_1 - \alpha_2$ *could be assigned to every element of* Θ*. This method does present the problem with the overlapped values. For example,* `grey` *appears in all three focal elements. Its value could be produced either by summing all assignments or by taking the maximum or*

minimum over all assignments. It is unclear which convention is most useful. Note that a simple valuation created in this manner might need to be renormalized to be interpretable.

Another useful category of models occurs when the outcome space Θ is ordered. Here one can talk about increasing and decreasing sets. For example, if $\Theta = \{0,1,2,3,4,5,6\}$, one could talk about the increasing focal elements: $\{0\}$, $\{0,1\}$, $\{0,1,2\}$, etc. or the decreasing focal elements: $\{6\}$, $\{5,6\}$, $\{4,5,6\}$, etc. or all intervals. Unwin[1984] suggests using nested sets of such intervals to specify belief functions. Again for simple valuations, this system is easiest to use if the set of focal elements form a partition of the outcome space.

2.2 Multivariate Models with no Conditionals.

The transition from univariate to multivariate space yields some new structure which can be exploited when defining focal elements. All of the tricks used in the univariate space, plus some new ones are now available. In particular, we can name logical relationships among the variables, extend lower dimensional models to larger frames and talk about distinguished outcomes.

Consider the frame of variables $\langle X, Y\rangle$ with the corresponding frame of discernment $\Theta = \Theta_X \times \Theta_Y$. For simplicity assume that $\Theta_X = \Theta_Y = \{:\mathtt{T}, :\mathtt{F}\}$. There are several logical restrictions which correspond to sets of outcomes:

Name	Symbolic Name	Set
and	$X \wedge Y$	$\{(:\mathtt{T}, :\mathtt{T})\}$
not-and (nand)	$\neg(X \wedge Y)$	$\{(:\mathtt{F}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{F}), (:\mathtt{T}, :\mathtt{F})\}$
or	$X \vee Y$	$\{(:\mathtt{T}, :\mathtt{T}), (:\mathtt{T}, :\mathtt{F}), (:\mathtt{F}, :\mathtt{T})\}$
not-or (nor)	$\neg(X \vee Y)$	$\{(:\mathtt{F}, :\mathtt{F})\}$
equivalence	$X = Y$	$\{(:\mathtt{T}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{F})\}$
exclusive-or (xor)	$X \neq Y$	$\{(:\mathtt{T}, :\mathtt{F}), (:\mathtt{F}, :\mathtt{T})\}$
First true	X	$\{(:\mathtt{T}, :\mathtt{T}), (:\mathtt{T}, :\mathtt{F})\}$
First false	$\neg X$	$\{(:\mathtt{F}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{F})\}$
Second true	Y	$\{(:\mathtt{T}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{T})\}$
Second false	$\neg Y$	$\{(:\mathtt{T}, :\mathtt{F}), (:\mathtt{F}, :\mathtt{F})\}$
Just first true	$X \wedge \neg Y$	$\{(:\mathtt{T}, :\mathtt{F})\}$
If X then Y	$X \Rightarrow Y$	$\{(:\mathtt{T}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{F})\}$
Just second true	$Y \wedge \neg X$	$\{(:\mathtt{F}, :\mathtt{T})\}$
If Y then X	$X \Leftarrow Y$	$\{(:\mathtt{T}, :\mathtt{T}), (:\mathtt{T}, :\mathtt{F}), (:\mathtt{F}, :\mathtt{F})\}$
Unknown	Θ	$\{(:\mathtt{T}, :\mathtt{T}), (:\mathtt{T}, :\mathtt{F}), (:\mathtt{F}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{F})\}$

Table 2.1 All models for two binary variables

This list is exhaustive. Note that first 14 entries are grouped with their logical complements. Unknown's complement is the empty set, which is not allowed. This notation can be easily extended to non-binary variables if the set of outcomes for a variable is partitioned into a set of true outcomes and false outcomes.

For some of these relationships, extending this notation to higher dimensions is straightforward. For others, such as "If X then Y", the extension is more naturally described using the conditional notation.

The extension of the exclusive-or relationship to higher dimensions deserves special note. For n variables, it becomes the "1-out-of-n" relationship. This class of logical relationship obviously generalizes to the "k-out-of-n" relationship. This latter has been successfully used to model possible failure states of a complex system with redundancies.

Extending frames of discernment produces another common class of multivariate models. For example, consider a valuation, V_1 over the frame X_1. In order to combine V_1 with another valuation V_2 over the frame of discernment X_1, X_2, it may be necessary to expand the frame of V_1 to X_1, X_2. There are two methods for doing this expansion, for set valuations this is done by *embedded projection* (Thoma[1989]). For simple valuations, this is done by replication. Both of these can be defined in terms of cylinder sets.

Cylinder Sets. *Let A be a set of tuples defined over the frame of variables $X_1, \ldots, X_n$ and let $Y_1, \ldots, Y_m$ be a collection of additional variables. We define the* cylinder set *of tuples over the frame $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ to be the set, $A^{\uparrow}$:*

$$A^{\uparrow} = \{(x_1, \ldots, x_n, y_1, \ldots, y_m) : (x_1, \ldots, x_n) \in A, y_1 \in \Theta_{Y_1}, \ldots y_m \in \Theta_{Y_m}\}$$

The models labeled "First true", "First false", "Second true" and "Second false" in Table 2.1 are all cylinder sets.

Extending set valuations is now a matter of replacing the focal elements (sets of interest) with their cylinder sets. More formally, if V is a valuation over the frame of variables, $\mathcal{A}$, its embedded projection over the frame $\mathcal{B} \supset \mathcal{A}$ is:

$$V'(B) = \begin{cases} V(A) & B = A^{\uparrow} \text{ for } A \subset \Theta_{\mathcal{A}} \\ 0 & \text{otherwise} \end{cases}$$

In the case of belief functions, this is easily recognized as Shafer's minimal extension (Shafer[1976]).

For a simple valuation, the set of cylinder sets corresponding to the simple events in the original frame forms a partition of the new outcome space. Therefore, replicating the valuation over the new variables creates a new valuation.

$$V'(b) = V(a) \qquad \text{for } b \in a^{\uparrow} \text{ for } a \in \Theta_{\mathcal{A}}$$

This is the procedure used to extend probability potentials in Lauritzen and Speigelhalter[1988].

There is an inverse operation, marginalization which goes from a larger frame to a smaller frame. As information in generally lost in this direction, there is a need for a summary operator to summarize the lost information. This is usually addition, although is certain cases maximization has proved more useful. In particular, Pearl[1988] uses it to find the most likely configuration of the leaves, and Shenoy[1991a,b] uses it maximize expected utilities.

The concept of ordering does not work well with multivariate outcome spaces. It is possible that there cases in which orderings among the variables are appropriate.

Discrete multivariate outcome spaces are essentially contingency tables. The margins of the table describe the distribution of each of the variables in the set under consideration. The margins of the table place some constraints on the values the cells of the table may take, but the rest are free to move. Madigan[1991?] points out that when eliciting joint distributions for a graphical model, in many cases, some of the margins will be specified by other parts of the model.

Consider for a moment a 2 by 2 table corresponding to two outcome variables X and Y. We can characterize the margins of the table by two values π_X which is the probability of $X = 1$ and π_Y which is the probability of $Y = 1$. All the tables that satisfy those marginal constraints are of this form:

	$X = 0$	$X = 1$	Y margin
$Y = 0$	$(1 - \pi_X)(1 - \pi_Y) + \lambda$	$\pi_X(1 - \pi_Y) - \lambda$	$(1 - \pi_Y)$
$Y = 1$	$(1 - \pi_X)\pi_Y - \lambda$	$\pi_X \pi_Y + \lambda$	π_Y
X margin	$(1 - \pi_X)$	π_X	1

Here λ, the only parameter free to vary, is the *interaction*. If $\lambda = 0$ then X and Y are *independent*. Independence reduces the dimensionality of the problem under consideration and so is of prime importance. Chapter 3 of Pearl[1988] describes the central role of independence in specifying graphical models. The interaction is a measure of lack of independence.

Changing parameters to margins and interactions does not reduce the number of parameters, but making independence assumptions (essentially setting interactions to zero) does. Thus examining the results for many different interaction variables provides a method of testing independence assumptions. As these are often a crucial part of the graphical model construction process, this interaction parameter will prove to be very useful.

In multi-way tables, we have two-way and higher order interactions. There are ways of simplifying interaction models among parameters by using graphs to express the dependencies (Darroch, Lauritzen and Speed[1980]). This is one of the origins of the current work on graphical models.

2.3 Conditional Models with one dependent variable.

Conditional models play a central role in the elicitation procedure for graphical models suggested by both Lauritzen and Speigelhalter[1988] and Pearl[1988]. In fact they both use as their primary representations directed graphs with nodes representing variables. Associated with each node is a valuation representing the conditional distribution of that variable given its parents (nodes with no parents are assigned unconditional valuations). This method of specification always results in a consistent joint distribution if the components are all probability distributions (and all independence assumptions implicit in the graph are in fact true).

In the crudest form, such conditional representations are still simple or set valuations, with potentially large numbers of parameters requiring specification. In fact, there are fewer constraints on a conditional valuation, so there are more free parameters in the system. The key to reduced parameterization is finding structure among possible configurations of the independent variables.

Pearl[1988] describes some noisy-logic models which are useful when all variables are logical. Consider again the three binary variables: X_1, X_2 and Y. The occurrence of both X_1 and X_2 usually causes Y and Y usually doesn't happen in the absence of such variables. This knowledge can be stored as a probability distribution using only two numbers: the probability of Y true given X_1 and X_2 true and the probability of Y true otherwise. A belief function version would increase the number of parameters to four (an upper and lower bound on each of those two probabilities).

These *noisy-logic* models are easily extended to more variables and other logical combinations. Pearl[1988] actually extends the *noisy-or* model in a quite different direction. He introduces a parameter associated with each input providing the probability that the input is "inhibited." But he then fills out the table using assuming that these inhibition probabilities are independent. As is true for most independence assumptions, this one must be verified on a case by case basis and should not be taken blindly. The *noisy-or* and *noisy-and* models described here are Pearl's models simplified by allowing only global inhibiters.

One extension of the *noisy-or* model useful in reliability problems is the k-out-of-n model. Here there are n input component or subsystem state variables $(X_1, \ldots, X_n)$. The probability that Y is in a failure state depends on how many of the inputs are in a failure state. Thus there would be a failure probability for Y associated with $0, 1, \ldots, n$ failures among the components.

Note that a simple logical rules can be implemented by setting the conditional probabilities to 0 and 1. Thus the rule that a system with 5 components fails if and only if a majority of the components fail associates 0's with 0, 1 or 2-out-of-5 and 1 with 3, 4 or 5-out-of-5. However, for simple valuations, there is an important restriction here: namely the relation must be bijective. That is we must both assert that the system will fail if 3 or more components fail and will not fail if fewer than 3 components fail. Without moving to set valuations we cannot express the system will fail if 3 or more components fail and its state will be unknown otherwise; this requires set valuations. (Note: we can say that the system will fail if 3 or more components fail and its state will be *uncertain* otherwise; this, however, requires specifying the probability that the system will fail even if less than three components have failed.)

Moving from simple valuations to set valuations removes the problem of being forced to accept bijections between the independent and dependent variables. In particular, the proposition "If X then Y" is equivalent to the set $A = \{(:\mathtt{T}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{F})\}$ while the proposition "X is true if and only if Y is true" is equivalent to the set $B = \{(:\mathtt{T}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{F})\}$. A logical valuation corresponding to either statement has the single focal element A or B. A simple discounted model is produced assigning a parameter λ to the set A or B and letting the value associated with the whole frame Θ be a simple function of that parameter (for example, 0, or $1 - \lambda$).

Shafer[1976] developed these discounted set models for belief functions, and they are easily generalized for other set valuations. Note that only a subset of these discounted logical models can be described with simple valuations. In particular, for a set valuation over a set C and the frame to be expressed as a simple valuation, C must restrict the dependent variable to exactly one value for each configuration of the independent variables. Set B of the preceding paragraph obeys this restriction but Set A does not.

It is possible to perform a large number of tasks using this notation. For example, the relation $Z = X + Y$ could be represented using this system, although the result is likely to be impractical unless X and Y have very small outcome spaces.

Smetts (see Shafer[1982]) describes a method for constructing conditional belief function called *Conditional Embedding*. Consider let X be the dependent variable and Y be the independent variable. Let θ_y be

an element of Θ_y and $V(X|Y=\theta_y)$ be a set valuation describing information about X given $Y=\theta_y$. Thus V is defined over subsets of Θ_X. Define a new set valuation, $V_{\Uparrow X,Y}(A)$ as follows:

$$V_{\Uparrow X,Y}(A) = \begin{cases} V(B) & \text{if } A = (B \times \{\theta_y\}) \cup (\Theta_x \times (\Theta_y - \{\theta_y\})) \text{ for some } B \subseteq \Theta_x; \\ 0 & \text{otherwise.} \end{cases}$$

This new valuation $V_{\Uparrow X,Y}$ agrees with V when $Y=\theta_y$ and is unknown otherwise.

Smetts goes on to combine several belief functions constructed in this way using Dempster's Rule for combining independent belief functions. The independence assumptions certainly must be questioned in this situation, and several surprising results have been constructed with this method. One of the problems with describing conditional belief function is that the joint belief function is not uniquely specified by the conditional belief functions and the marginal belief function over the independent variables (Shafer[1982], Almond[1990]). Hopefully some of the methods described here will help produce alternate possibilities for describing conditional belief functions.

One way of working around such independence assumptions would be to consider Analysis of Variance like models for the probability of the dependent variable given the independents (possible after some appropriate transformation such as the log odds, $\frac{\log p}{\log(1-p)}$). Here the independent variables are thought of as factors, and the probability of of the dependent variable is the response. Probabilities can then be built up from *main effects* of the independent variables and *interactions* them. Higher order interactions could be aliased to smaller interactions. Sensitivity to interaction effects could be studied here as well.

One particular advantage of this approach, is that such models can be fit to data using conventional statistical techniques, namely logistic regression or generalized linear models (McCullagh and Nelder[1983]). These models would allow continuous variables among the independent variables as well.

2.4 Conditional Models with more than one dependent variable.

These models should obviously apply the tricks of both Sections 2.2 and 2.3. This is difficult. Fortunately such models should be relatively rare; most people prefer to use structures of the type provided the previous sections.

3. The TS-set Notation

The two types of valuations, set valuations and simple valuations, are both simplified by identifying focal elements which are associated with the parameters. Although simple valuations are later expanded into arrays, the use of focal elements reduces the number of parameter which must be specified. Because focal elements play such a critical role, especially for set valuations, it is worthwhile to spend some time developing special notation for focal elements.

In the *TS-set* notation, set valued components replace single element components in ordered pairs or tuples, allowing the descriptions of multivariate focal elements to suggest relationships among the variables represented by the focal element. This notation not only makes expressing focal elements of valuations (such as belief functions) simpler, but it also provides a convenient notation for expressing the basic set operations underlying belief functions manipulations.

Associated with each multivariate belief valuation is a *frame of variables*, $\mathbf{X} = (X_1, \ldots, X_m)$, which corresponds to the *frame of discernment* $\mathbf{\Theta}(\mathbf{X}) = \prod_{X \in \mathbf{X}} \Theta_X$. Consider a tuple of non-empty sets $(A_1, \ldots, A_m)$ where $A_i \subseteq \Theta_{X_i}$, thus it is a *pair* (or tuple) of *subsets*; we will use it as a short-hand notation for $A_1 \times \cdots \times A_m$. This notation can be used to suggest logical relationships among the variables, for example the set $(\{0\}, \Theta_{X_2})$ suggests that $X_1 = 0$ while X_2 is unknown.

The set $\bigcup_{j=1}^{n} (A_{1\,j}, \ldots, A_{m\,j})$ is a subset of $\mathbf{\Theta}(\mathbf{X})$ and hence is also a possible focal element. The *TS-set* notation $\{(A_{1\,1}, \ldots, A_{m\,1}), \ldots, (A_{1\,n}, \ldots, A_{m\,n})\}$ is used to represent this set. Although this notation can be confused with a set of tuples of sets, a set of tuples of sets is seldom of interest in specifying valuations. When it is clear from the context that the set in question is a focal element—a set of tuples—the TS-set notation can highlight logical structure in the focal element. A focal element in a TS-set representation is called a *TS-set*.

There is a special notation used for each of the component sets $A_{i\,j}$ of a TS-set. Instead of braces $\{\,\}$, brackets $[\,]$ are used to express these sets. This schema also means that a TS-set is constructed out of

three nesting and contrasting sets of brace-like markers, so that the template for a typical TS-set would be: $\{([\ldots], \ldots, [\ldots]), \ldots, ([\ldots], \ldots, [\ldots])\}$. If the component set consists of a single element, then the brackets are suppressed and just the single element is written. As a consequence, ordinary sets of ordered tuples are also TS-sets. If the component set is the entire frame of discernment, Θ_{X_i}, associated with a variable, then Θ_{X_i} or simply Θ is written instead of the set in brackets.

Example 3.1. Examples of TS-sets. *Consider TS-sets over the frame of variables X_1, X_2, X_3 where $\Theta_1 = \{0,1\}$, $\Theta_2 = \{0,1,2\}$ and $\Theta_3 = \{0,1,2,3\}$. The following equations show certain sets in both TS-set and ordinary sets of triples (tuples) notation:*

$$\begin{aligned}
\{(0,\ [1,2]\ ,3)\} &= \{(0,1,3),(0,2,3)\} \\
\{(\Theta,0,\ [1,3])\} &= \{(0,0,1),(0,0,3),(1,0,1),(1,0,3)\} \\
\{(0,\ [0,1]\ ,\ [0,1])\ ,([0,1]\ ,\ [1,2]\ ,\Theta)\} &= \left\{ \begin{array}{l} (0,0,0),(0,0,1), \\ (0,1,0),(0,1,1),(0,1,2),(0,1,3), \\ (0,2,0),(0,2,1),(0,2,2),(0,2,3), \\ (1,1,0),(1,1,1),(1,1,2),(1,1,3), \\ (1,2,0),(1,2,1),(1,2,2),(1,2,3) \end{array} \right\}
\end{aligned} \tag{3.1}$$

A given focal element has more than one TS-set representation. As single elements are used to represent sets containing themselves, both the right and left sides of Equations 3.1 are in TS-set notation. This diversity allows one to chose a TS-set representation according to taste, and in particular to choose a representation which may suggest important structure in the focal element. However, often (such as when the BELIEF package needs to test two TS-sets for equality) a canonical representation is necessary. In order to obtain such a representation from an arbitrary TS-set, the set is first *exploded*—each tuple which contains a component set is replaced by a collection of tuples all of whose components are all singletons—and then the collection of tuples is sorted by some arbitrary ordering (such as alphabetically on the names of the components in order). A TS-set in such a state is called a *normalized TS-set*; note that normalized TS-sets are also in the standard notation for a set of tuples (a subset of the multivariate outcome space). The right hand side TS-sets in Example 3.1 are all normalized.

This notation suggests a very simple storage schema for reduced parameter valuations (especially set valuations like belief functions); an extension of this schema is used in the BELIEF package. Set valuations are stored as a list of value and focal element pairs, one value for each focal element. Focal elements are stored in the normalized (TS-set) form (for easy comparison). For a set valuation with only a few focal elements but defined over a large multivariate outcome space, this storage schema is very compact. The cost is that additional set manipulations must be performed along with the multiplications and additions in order to apply the usual operators to valuations stored in this way. Section 3.1 describes methods for manipulating belief functions and other valuations stored in this form. The TS-set notation also allows BELIEF package users to compactly specify complex logical restrictions, especially when providing input data. Section 3.2 describes procedures for producing TS-sets based on logical relationships.

3.1 Basic Set operations

Shenoy and Shafer[1988] define two basic operations on valuations (and belief functions in particular): projection and direct combination. This section explores how the the BELIEF package implements these operations. Although the formulas given in Shafer[1976] are quite explicit as to how to multiply and sum m-values to perform these operations, they require manipulation of sets as well as numbers. These set operations are quite simple in the TS-set notation, as described here.

The projection operation consists of two parts: minimal extension and marginalization. Minimal extension simply transforms each focal element of a belief function into a new focal element in the extended space. To minimally extend a belief function stored as a list of m-value, focal elements pairs simply requires minimally extending each focal element—expressing the focal elements in the new frame. Suppose a belief function over the frame $\mathbf{X} = (X_1, \ldots, X_m)$ is (minimally) extended to the frame $\mathbf{X}' = (X_1, \ldots, X_m, X_{m+1}, \ldots, X_{m'})$. Let $\{(A_{1\,1}, \ldots, A_{m\,1}), \ldots, (A_{1\,n}, \ldots, A_{m\,n})\}$ be a TS-set representation of a focal element to be extented. The extended focal element is then: $\{(A_{1\,1}, \ldots, A_{m\,1}, \Theta_{X_{m+1}}, \ldots, \Theta_{X_{m'}}), \ldots, (A_{1\,n}, \ldots, A_{m\,n}, \Theta_{X_{m+}}$ All that the BELIEF package does to extend each focal element is tack the frames corresponding to the new variables onto the end of each ordered tuple. A simple valuation extended in this way will produce a valuation which is replicated over the new variables, just as is desired.

Marginalization is equally simple. Each focal element in turn is marginalized to the new frame and then the mass values for focal elements which have become the same through the marginalization are summarized (usually summed). To marginalize a focal element from a frame $\mathbf{X}' = (X_1, \ldots, X_m, X_{m+1}, \ldots, X_{m'})$ to a smaller frame $\mathbf{X} = (X_1, \ldots, X_m)$, the BELIEF package simply drops components from the tuples. That is, the focal element $\{(A_{1\,1}, \ldots, A_{m'\,1}), \ldots, (A_{1\,n}, \ldots, A_{m'\,n})\}$ becomes $\{(A_{1\,1}, \ldots, A_{m\,1}), \ldots, \ (A_{1\,n}, \ldots, A_{m\,n})\}$. Focal elements are then normalized, and the mass values for those focal elements which are now the same are combined.

Finally, projection is done by combining the marginalization and minimal extension at the focal element level and then summing the masses of identical focal elements. The BELIEF package additionally allows for the order of the variables (and thus the order of the components in a tuple) to be permuted at the time of a projection (or marginalization or minimal extension).

The most popular method for combining two belief functions is Dempster's product-intersection rule (Dempster[1968]). Following Thoma[1989] (sell also Almond[1990]) we define combination in two parts, *convolution* and *normalization*. Let m_1 and m_2 be two set valuation defined over the same frame, then their convolution is defined as follows:

$$m_1 * m_2(C) = \sum_{\substack{A,B \subseteq \Theta \\ A \cap B = C}} m_1(A) \cdot m_2(B) \qquad \forall C \subseteq \Theta \tag{3.2}$$

The summation may be replaced with another summary operator (such as maximization), and the product may be replaced with another combination operator (such as addition). Because it is illegal to place mass on the empty set for a proper belief function, Dempster *normalizes* the belief function by dividing by $1 - m_1 * m_2(\emptyset)$. This normalization step can be saved until the computations need to be interpreted (see Almond[1990]).

Equation 3.2 shows that the convolution of two set valuations requires two steps. First, each pair of focal elements—one from each belief function—is intersected and the product of the corresponding masses is computed. Second, mass values for identical focal elements are eliminated. These require two set operations: intersection and equality testing.

To explore set intersection in the TS-set notation, let $\{(A_{1\,1}, \ldots, A_{m\,1}), \ \ldots, , (A_{1\,n}, \ldots, A_{m\,n})\}$ and $\{(B_{1\,1}, \ldots, B_{m\,1}), \ \ldots, (B_{1\,n'}, \ldots, B_{m\,n'})\}$ be two TS-sets over a common frame $\mathbf{X} = (X_1, \ldots, X_m)$. The intersection is then $\bigcup_{i=1}^{n} \bigcup_{i'=1}^{n'} (A_{1\,i} \cap B_{1\,i'}, \ldots, A_{m\,i} \cap B_{m\,i'})$ where $(A_{1\,i} \cap B_{1\,i'}, \ldots, A_{m\,i} \cap B_{m\,i'})$ is considered to be the empty set if any of the components $A_{k\,i} \cap B_{k\,i'}$ is empty. If both sets are normalized TS-sets—reduced to ordinary tuples and sorted—intersection can be done with one loop through all the elements in both sets rather than using computationally expensive nested loops.

The second step of both the convolution and the projection operations is to combine all redundant focal elements. This requires a rapid test for equality of focal elements. Normalizing the TS-sets before testing for equality eliminates makes this operation more efficient, as does the trick of *naming* them (See Section 3.3).

The normalization operation only involves dividing the mass values of the various focal elements by the conflict (mass assinged to the empty set); the focal elements themselves are left unchanged. The Renormalization Theorem (Almond[1990]) states that the normalization step can be saved until last. In particular, that means that it is often desirable for the BELIEF package to store or pass unnormalized belief functions. This can be done very simply by allowing the empty set to be a possible focal element.

3.2 Using TS-sets to represent logical restrictions

More important than the computational convenience of the TS-set notation for performing the basic belief function manipulations, is the way that they can be used to specify belief functions. For the user of the BELIEF package, they provide means of compactly specifying focal elements without laboriously typing similar patterns of outcomes. Such patterns frequently arise when specifying logical relationships among the variables, and here TS-sets are of particular help.

For the moment, consider variables with only two possible values, `:T` (true) and `:F` (false). Any belief function over a single such variable would have three possible focal elements, $\{(\texttt{:T})\}$, $\{(\texttt{:F})\}$ and $\{(\Theta)\}$ (or `:T`, `:F` and Θ) which correspond to logical true, logical false and logical unknown respectively. Truth tables over multiple logical variables can be constructed by using combinations of those three values. TS-set representations can mirror such logical combinations.

Examine a two-variable valuation corresponding to the *if-then* rule, "if X_2 then X_1". If X_2 is true, then the value of X_1 is true, thus the pair $(\texttt{:T}, \texttt{:T})$ is included in the focal element describing this rule. If X_2 is false, then the value of X_1 is unknown, thus the TS-set $(\Theta, \texttt{:F})$ is included. The focal element corresponding to the statement "if X_2 then X_1" is the union of these two TS-sets: $\{(\texttt{:T}, \texttt{:T}), (\Theta, \texttt{:F})\} = \{(\texttt{:F}, \texttt{:F}), (\texttt{:T}, \texttt{:F}), (\texttt{:F}, \texttt{:F})\}$. Similarly, the *if-and-only-if* rule "X_2 if and only if X_1" is derived by noting that if X_2 is true so is X_1 and if X_2 is false then so if X_1. Its focal element is $\{(\texttt{:T}, \texttt{:T}), (\texttt{:F}, \texttt{:F})\}$.

More complex rules are created by substituting logical combinations of conditions for the single condition X_2. The simplest are the `ifall` and the `ifany` (or `ifand` and `ifor`) rules. Over the frame $\mathbf{X} = (X_1, \ldots, X_m)$, the `ifall` rule states "if all of X_2 through X_m are true then so is X_1" and the `ifany` rule states "if any of X_2 through X_m are true then so is X_1." These (along with the corresponding if and only if rules) are constructed from smaller TS-sets. For example, the `ifall` TS-set is composed of two pieces: one which expresses "if `logical-and` $(X_2, \ldots, X_m)$ then X_1 is true" and one which expresses "if `logical-nand` $(X_2, \ldots, X_m)$ then X_1 is unknown." Similarly, the `ifany` rule is composed of from the `logical-or` and `logical-nor` or its inputs, where the rules `logical-and`, `logical-nand`, `logical-or` and `logical-nor` are defined below.

Either for building complex rules, or simply for asserting logical relationships, the TS-set notation also provides a simple method for constructing truth tables corresponding to logical conjunctions and disjunctions. Two cases, `logical-and` and `logical-nor` (not or) are simple; they each correspond to focal elements with a single outcome: $(\texttt{:T}, \ldots, \texttt{:T})$ and $(\texttt{:F}, \ldots, \texttt{:F})$ respectively. The `logical-or` and `logical nand` (not and) rules are built recursively from smaller TS-sets. To construct the TS-set corresponding to the `logical-or` rule over $(X_1, \ldots, X_m)$ note that if X_1 is true, then the rule is true no matter which of the values in $(\Theta_{X_2}, \ldots, \Theta_{X_m})$ the other variables $(X_2, \ldots, X_m)$ assume. This corresponds to the TS-set $(\texttt{:T}, \Theta_{X_2}, \ldots, \Theta_{X_m})$. If X_2 is false, then one of the other variables must be true; that is, `logical-or` of $(X_2, \ldots, X_m)$ is true. This process continues recursively. The union of all the TS-sets constructed at each stage is a representation of the `logical-or` rule applied to $(X_1, \ldots, X_m)$. The `logical-nand` and `logical-xor` (exclusive or) TS-sets can be built with similar recursive procedures.

Example 3.2 Logical Relationships. *Imagine a series of focal elements describing logical relationships among three binary variables X_1, X_2 and X_3. For each relationship discussed above, Table 3.1 shows the name of the relationship, the corresponding predicate calculus notation and its representation as a TS-set.*

Another method of inputting belief functions is to construct them by conditionally embedding a collection of conditional belief functions, as defined in Section 2.3. To conditionally embed a single conditional belief function $\mathrm{BEL}(X|Y = \theta_Y)$ in the frame (X, Y), all of its focal elements must be conditionally extended into the new frame. Let B be any focal element of $\mathrm{BEL}(X|Y = \theta_Y)$. The corresponding focal element of $\mathrm{BEL}(X|Y = \theta_Y)_{\Uparrow(X,Y)}$, using the TS-set notation, is $\{(B, \theta_Y), (\Theta_X, \Theta_Y - \{\theta_Y\})\}$. Note the similarity to the if-then rule with θ_Y playing the role of true for the condition, and B playing the role of true for

Name	Relationship	TS-set
ifall	$X_1 \Longleftarrow (X_2 \wedge X_3)$	$\{(:\mathtt{T}, :\mathtt{T}, :\mathtt{T}), (\Theta, :\mathtt{F}, \Theta), (\Theta, :\mathtt{T}, :\mathtt{F})\}$
iffall	$X_1 \Longleftrightarrow (X_2 \wedge X_3)$	$\{(:\mathtt{T}, :\mathtt{T}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{F}, \Theta), (:\mathtt{F}, :\mathtt{T}, :\mathtt{F})\}$
ifany	$X_1 \Longleftarrow (X_2 \vee X_3)$	$\{(:\mathtt{T}, :\mathtt{T}, \Theta), (:\mathtt{T}, :\mathtt{F}, :\mathtt{T}), (\Theta, :\mathtt{F}, :\mathtt{F})\}$
iffany	$X_1 \Longleftrightarrow (X_2 \vee X_3)$	$\{(:\mathtt{T}, :\mathtt{T}, \Theta), (:\mathtt{T}, :\mathtt{F}, :\mathtt{T}), (:\mathtt{F}, :\mathtt{F}, :\mathtt{F})\}$
logical and	$X_1 \wedge X_2 \wedge X_3$	$\{(:\mathtt{T}, :\mathtt{T}, :\mathtt{T})\}$
logical or	$X_1 \vee X_2 \vee X_3$	$\{(:\mathtt{T}, \Theta, \Theta), (:\mathtt{F}, :\mathtt{T}, \Theta), (:\mathtt{F}, :\mathtt{F}, :\mathtt{T})\}$
logical nand	$\neg(X_1 \wedge X_2 \wedge X_3)$	$\{(:\mathtt{F}, \Theta, \Theta), (:\mathtt{T}, :\mathtt{F}, \Theta), (:\mathtt{T}, :\mathtt{T}, :\mathtt{F})\}$
logical nor	$\neg(X_1 \vee X_2 \vee X_3)$	$\{(:\mathtt{F}, :\mathtt{F}, :\mathtt{F})\}$
logical xor	$(X_1 \wedge \neg X_2 \wedge \neg X_3) \vee$ $(\neg X_1 \wedge X_2 \wedge \neg X_3) \vee$ $(\neg X_1 \wedge \neg X_2 \wedge X_3)$	$\{(:\mathtt{T}, :\mathtt{F}, :\mathtt{F}), (:\mathtt{F}, :\mathtt{T}, :\mathtt{F}), (:\mathtt{F}, :\mathtt{F}, :\mathtt{T})\}$

Table 3.1. Logical relationships expressed as TS-sets.

the consequence. Conditional embedding a collection of conditional belief functions is a straightforward extension.

3.3 Named TS-sets

The BELIEF package primarily stores belief functions (set valuations) using the *sparse m-value* representation. Each belief function is a list of pairs, where each pair consists of a mass value and a set. As discussed above, testing these sets for equality is an important step in both the convolution and the projection operations. To facilitate rapid equality tests, the BELIEF package stores focal elements using *named TS-sets*.

Each TS-set is stored as a list of tuples (as it is normalized). Checking for equality requires examining each component of every tuple, a laborious process. Furthermore, if each time a given TS-set (or a TS-set over a different frame whose element where identical) occurred in any belief function the entire TS-set was stored, then the BELIEF package could quickly run out of memory in complex problems. A more efficient scheme is needed.

Therefore, the BELIEF package *names* each new TS-set it encounters. The name is actually a pointer to the TS-set, so the TS-set is quickly available from its name. In representations of belief functions only the name is stored. The actual TS-sets are stored in a hash table, and a given TS-set is only stored once no matter how many times its name is used as a focal element in a belief function. Furthermore, in naming focal element, the BELIEF package pays no attention to the frame over which it is defined. Thus, two focal elements with identical exploded forms but over different frames are stored in the same location, even though they may have different meanings in the two context.

When the BELIEF package creates a TS-set (by intersection, projection, conditional embedding, or from user input), it normalizes the TS-set and then looks up its name. The names are stored in a hash table indexed on the normalized TS-sets for rapid retrieval. If no name is found for the normalized TS-set, it is given a new name and it is added to the hash table. To test two named TS-sets for equality, the BELIEF package need only check their names (pointers) for equality. To intersect two named TS-sets, the BELIEF package can fetch the original TS-sets and perform the intersection.

Two special TS-sets have special names. One is the *empty TS-set*, $\{\}$ and the other is the *frame* (Denoted `**frame**` in the `BELIEF` package). These respectively form the zero and the unit element of the set intersection operation. As they occur frequently, algorithms trained to recognize them can give them special treatment. This improves the speed of many operations.

The sparse m-value method for storage has several obvious advantages for reduced parameter set valuations. Generally speaking, the number of focal elements in a graphical belief function is $\prod k_i$ where k_i is the number of focal elements in the ith component. As typical values for k_i are 1, 2 and 3, the sparse m-value method of storage often performs well.

3.4 Generalized Logical Variables

Many of the multivariate descriptions found above rely on the concept of "true" and "false." Although the definition of "true" and "false" is usually easy to recognize in binary frames, it can be extended to larger frames as well. The key is that the set of outcomes must be partitioned into two sets corresponding to true and false (Θ itself always corresponds to unknown). The identification of *logical-true* and *logical-false* greatly simplifies the specification of complex logical constructions and can be easily expanded in TS-set notation.

For example, if the variable is "Hair Color" which ranges over the values $\{$:Blond, :Brunette, :RedHead$\}$ and the proposition of interest revolves around whether or not a person's hair is red, then :RedHead could be used for logical true and $[$:Blond, :Brunette$]$ would be logical false. Θ would remain logical unknown. Similarly, if the variable is "Language" and the proposition involves whether or not the language is Semitic, then the values $[$:Hebrew, :Arabic$]$ might be logical true and the others would comprise logical false. Here sets of values corresponding to logical true and logical false are substituted for :T and :F respectively in the TS-sets corresponding to the logical rules.

To facilitate specifying logical relationships, the BELIEF package supports the notation of *Generalized Logical Variables.* A generalized logical variable is one of three things:

1. The name of a binary variable. In this case the first outcome is considered to be "true" and the second the be "false."
2. A list of the form (*variable* θ_1 θ_2 ...), Where $\theta_1, \ldots$ is a list of outcomes which should be consider in the true set.
3. A list of the form (not *generalized-variable*), which specifies the logical complement of the generalized-variable.

4. Application to Model Construction

The specification a graphical model requires the specification of both the graphical structure and the valuations describing the relationships. Even if all valuations are eventually stored as simple valuations, specifying all valuations as set valuations (and possibly later converting to simple valuations) helps support reduced parameter representations. For set valuations, the valuations specification task has two subtasks: (1) specifying the focal elements and (2) specifying the corresponding parameters. As in theory many parts of the model specification will be reused, there should be resources to support each of these tasks.

To support model construction, GRAPHICAL-BELIEF needs to support four classes of resources:

1. *Graph Fragments.* Partial pieces of a graphical model (with or without valuations, or possibly with default valuations) which model a specific type of interaction.
2. *Valuations.* Preset valuations corresponding to certain situations.
3. *Focal Elements.* Commonly used focal elements.
4. *Parameter Values.* This resource would also specify distributions for parameter values in a two stage model.

These resources can be organized into resource libraries. They can also be used in conjunction with expert systems as part of Knowledge Based model construction.

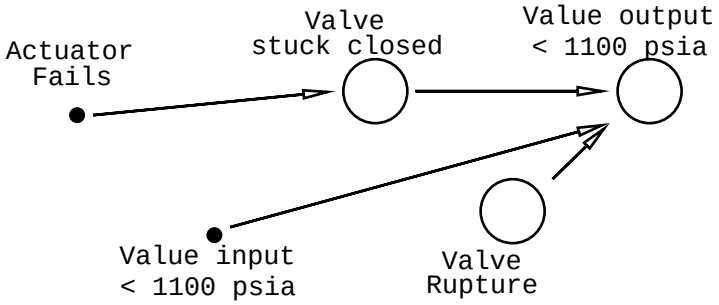

Figure 4.1. Graph Fragment resource for Isolation valve

4.1 An Example of Resources

In reliability problems, these resources will often be tied to specific component types. Thus the inclusion of a particular component into the system might indicate that an associated graph fragment belongs in the system. For example, an isolation valve might have the associated graph fragment shown in Figure 4.1. The small closed circles represent points at which other graph fragments might be joined onto this one.

Associated with the node “Value output <1100 psia” is a valuation describing the relationship between the corresponding variable and its three parents: “Valve stuck closed,” “Valve input <1100 psia” and “Valve Rupture.” This would be a logical relationship, a logical-or or a noisy-or. There could be a valuation resource for or-gates which could be used to specify this valuation.

Associated with the node “Stuck Closed” is a valuation describing the relationship of the valve in relation to the actuator position. In particular, there will be two parameters, one associated with the condition “Actuator open” and one with the condition “Actuator Closed.” The focal elements associated with these conditions can be described with a focal element resource. The parameters describing the belief in failure (which may be associated with the manufacturer of the valve) would be described with a parameter resource.

4.2 Libraries, Bookcases and Cut and Paste.

One of the most obvious ways to organizes graphical model construction resources is to use a *library* of model fragments. In GRAPHICAL-BELIEF there will be one library for each of the four resource types described above. As many parts of the model can simply be copied from the library, libraries will help system designers rapidly build models from small pieces.

Libraries allow the model specification project to be divided into logical task groups; for example, an engineer in charge of purchasing and testing might be responsible for developing the parameter library while a safety engineer might be responsible for developing the graph fragment library. Finally the system designer might be responsible for putting the graph fragments together to form the final model for her system or subsystem.

The name "library" implies not only a collection of resources but also a good "card catalog", that is index system. A *bookcase* is a simplified library with no index. Bookcases are useful for rapidly prototyping parts of the model. Later, systematic cataloging and indexing for the resources in the bookcase can be built, turning it into a library. The existing mechanism in BELIEF for naming TS-sets and associating names with values is essentially a bookcase.

Finally, the cut and paste mechanism should be tightly tied to the resource libraries. When a user performs a cut (or copy) and paste operation, it is often because the cut segment of the graph is a logical unit which is replicated. This means it is a candidate for addition to the resource library. GNU Emacs supports a "kill ring" which records cuts and copies in reverse chronological order. A similar "kill ring" bookcase could capture copy and cut commands. The user could browse this kill ring and identify re-usable model fragments with names, placing them in a personal bookcase.

One additional difficulty with the cut and copy operation is determining at what level of detail to cut and copy. In particular, if a fragment of graph is copied, then should the underlying valuations, focal elements and parameters be copied as well? There is probably no good answer to this question, and it may need to be answered with a dialog at either cut or paste time (possibly with different cut and paste commands). Yet another way to resolve the problem is to let the old valuation become the new default valuation after it is pasted. This might be a noisy default, one that issues a warning when it is used, until is explicitly accepted as appropriate for the new situation.

4.3 Knowledge Based Model Construction.

It is easy to envision a graphical model which includes far to much detail. For example, a complete medical expert system would contain a lot of information irrelevant to a particular case, such as information about pregnancy complications which would not be needed for male patients. Knowledge Based Model Construction (Breese *et al.*.[1991]) offers a method for building small graphical models germane to a particular situation.

It works by using two layers. The upper layer is a conventional rule based system which identifies model components which may be relevant to a particular situation. In particular, it would identify graph fragments and associated valuation which would be appropriate in a particular context. The lower layer is the graphical model assembled by the rule based system; the graphical model which can provide probabilistic information about various decision alternatives and hypothesis.

The rule based system would also know how to turn many common queries into manipulations of the graphical model. Thus it would act as an interpretation system, hiding the details of the graphical model from users who aren't interested in the technical details. It would also be able to turn numerical explanations from the graphical model into natural language explanations for delivery to the end user.

The combination of graphical models and traditional AI techniques represented in knowledge based model construction is one of the most promising new techniques in the field. In order for it to work properly, graphical model systems must be able to support large libraries of model component resources. It is here that the reduced parameter models, and distinguished set representation schemes described here will have their biggest impact.

Bibliography

Almond, Russell G.[1989a]. "Operating Instructions for the BELIEF package." Technical Report S-128, Harvard University, Department of Statistics (the BELIEF package reference manual).

Almond, Russell G.[1989b]. "The BELIEF Package: A System for Exploring Graphical Belief Models." Technical Report S-129, Harvard University, Department of Statistics (the BELIEF package code).

Almond, Russell G. [1990]. *Fusion and Propagation in Graphical Belief Models: An Implementation and an Example.* Ph.D. dissertation and Harvard University, Department of Statistics Technical Report S-130. To be published as a monograph from Van Nostrand Reinhold.

Almond, Russell G. [1991]. "Building Blocks for Graphical Belief Models." *Journal of Applied Statistics,* **18**, 63–76.

Almond, Russell G. and Kong, Augustine [1991]. "Some Heuristics for Building an Optimal Tree of Cliques from a Graph or Hypergraph." University of Chicago, Department of Statistics, Research Report 329. (Submitted for Publication).

Breese, John S., Goldman, Robert and Wellman, Michael P. [1991]. "Knowledge-Based Construction of Probabilistic and Decision Models: An Overview." Workshop presented at AAAI-91.

Darroch, J. N., Lauritzen, S. L., and Speed, T. P.[1980]. "Markov Fields and Log-Linear Interaction Models for Contingency Tables," *The Annals of Statistics,* **8**, 522-539.

Dempster, Arthur P. [1968]. "A Generalization of Bayesian Inference (with discussion)." *Journal of the Royal Statistical Society, Series B,* **30**, 205-247.

Madigan, David [1991?]. Private Communication.

Madigan, David and York, Jeremy [1991]. "Strategies of Bayesian Updating of Graphical Models" Presented at *23rd Interface Conference*, Seattle, WA.

McCullagh, Peter and Nelder, John A.[1983]. *Generalized Linear Models.* Chapman and Hall.

Oliver, Robert M., and Smith, James Q.[1990]. (eds.) *Influence Diagrams, Belief Nets and Decision Analysis.* John Wiley and Sons.

Pearl, Judea [1988]. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, California.

Shafer, Glenn [1976]. *A Mathematical Theory of Evidence.* Princeton University Press.

Shafer, Glenn [1982]. "Belief Functions and Parametric Models." *Journal of the Royal Statistical Society, Series B,* **44**, 322-352.

Shafer, Glenn and Shenoy, Prakash P.[1988]. "Bayesian and Belief-Function Propagation." School of Business Working Paper No. 192. University of Kansas.

Shenoy, Prakash P.[1991a]. "A Fusion Algorithm for Solving Bayesian Decision Problems," in *Uncertainty in Artificial Intelligence, Proceedings of the Seventh Conference,* pp 361-369.

Shenoy, Prakash P.[1991b]. "Valuation-Based Systems for Discrete Optimization," in *Uncertainy in Arficial Intelligence,* **6** (to appear).

Shenoy, Prakash P. and Shafer, Glenn [1990]. "Axioms for Probability and Belief-Function Propagation." in *Uncertainty in Artificial Intellegence,* **4**, 169-198. Reprinted in Shafer and Pearl[1990] (eds), *Readings in Uncertain Reasoning*, Morgan-Kaufmann.

Lauritzen, Steffen L. and Spiegelhalter, David J. [1988]. "Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion)." *Journal of the Royal Statistical Society, Series B,*, **50**, 205–247.

Thoma, H. Mathis [1989]. "Factorization of Belief Functions." Ph.D. Thesis. Harvard University, Department of Statistics.

Unwin[1984]. "A Non-Bayesian Approach to Uncertainty Analysis in PRA: Shafer's Theory." Sandia National Laboratories Technical Report 6450-84-43.

Walley, Peter[1991]. *Statistical Reasoning with Imprecise Probabilities.* Chapman and Hall.