Hypergraph Grammars for Knowledge Based Model Construction

Russell G. Almond, StatSci
1700 Westlake Ave, N., Suite 500, Seattle, WA 98109
almond@statsci.com

ABSTRACT

Graphical belief networks, including Bayes nets and influence diagrams, can be represented with directed hypergraphs. Each directed hyperedge corresponds to a factor of the joint distribution of all variables in the model. A hyperedge replacement grammar is a collection of rules for replacing hyperedges with hypergraphs. A hyperedge replacement grammar for graphical belief networks defines a collection of graphical belief models.

Hyperedge replacement grammars have several interesting implications in the construction of graphical models. (1) They provide a way to represent the process of constructing a graphical model. (2) Coupled with an object-oriented variable type system, provide a convenient method for searching through candidate factors to fill a particular slot in the model graph. (3) They provide a method for integrating high-level and detailed views of a graphical model. (4) They provide a mechanism for representing uncertainty about the model structure.

Keywords: *Graphical Belief Networks, Knowledge Based Model Construction, Valuation Based Systems, Object Oriented Model Construction.*

1.0 Introduction

Many researchers have suggested using graphs to represent the structure of complex multivariate models. These models have gone by many names: *Bayes nets, influence diagrams,* and *graphical belief functions* are fairly common. Each of these models differs slightly in terms of what kind of distributions they represent: Bayes nets only allow probabilities, influence diagrams allow probabilities and utilities, graphical belief functions allow belief functions (and as a special case, probabilities). However, all of these types of models can be placed under a common framework, *graphical belief networks.* (Shenoy and Shafer[1990] describe the common framework in more detail.)

Despite the well known advantages of graphical belief networks, there are two difficulties: (1) very large models may still be computationally intractable, and (2) the model must be constructed before any question can be answered. Ideally, a graphical belief model could be assembled from existing model fragments like a jigsaw puzzle. This paper explores what shape the pieces must be in order for the computer to provide support for the assembly operation. The construction method described here may offer a solution to the first problem as well. This paper describes the results of attempting to implement these ideas in the GRAPHICAL-BELIEF computer environment for graphical modelling.

There are two roughly similar approaches to the problem of model construction: *Knowledge based model construction* and *Model fragment libraries. Knowledge based model construction* (Breese, Goldman and Wellman[1991], Goldman and Breese[1992], Charniak and Goldman[1993]) uses a conventional rule-based expert system to build the graphical model. Note that by building a model which is just large enough to solve a particular query, this approach also offers a way to address the very large model problem. *Model fragment libraries* (Almond, Bradshaw and Madigan [1993]) store fragments of models in an library or database. The modeller can then search through this library for factors or other model fragments which might be appropriate to the current model and re-use them in the new context.

Egar, Puerta and Musen[1992] propose using graph grammars for manipulating influence diagrams. Using the directed hypergraph representation of a graphical belief network model, the manipulations of the graph are represented by *Hyperedge replacement grammars* (Habel[1992]). This paper explores how the directed hypergraph representation can be used to support both knowledge based model construction and model fragment libraries.

Section 2 describes the directed hypergraph representation and its correspondence to more familiar representations. Section 3 describes hyperedge replacement models and some simple applications to graphical belief network construction. Section 4 shows how an object-oriented type system for the variables can produce intelligent search strategies for model fragments. And finally Section 5 discusses the implication of these ideas.

2.0 The directed hypergraph representation

In this paper, we use a *directed hypergraph* to represent the factorization of a large probability (or belief function) model. For example if $A1$ is independent of $A2$, we can factor the probability distribution over the variables, $A1, A2$ and E, into three pieces as follows:

$$p(A1, A2, E) = p(E|A1, A2)p(A1)p(A2)\ . \tag{1}$$

Figure 1a shows the directed hypergraph representation of this factorization. The round *nodes* represent the variables in the model. The square *hyperedges* represent the factors of the joint probability distribution. Each hyperedge has a number of *condition variables* and a *consequence variable*. *Tentacles* (arrows) go *from* the condition variables to the hyperedge and go *to* the consequence variables.

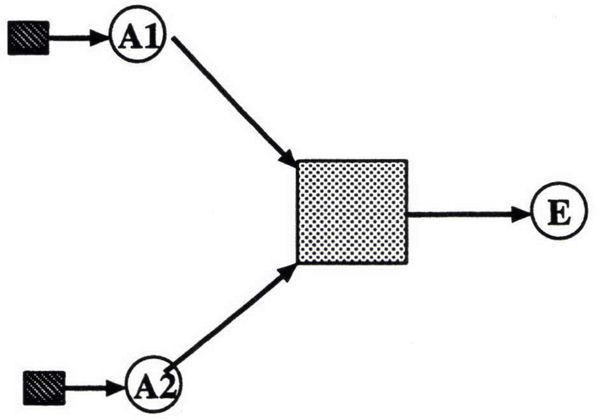

Figure 1a. Hyperedge (Replacement LHS)

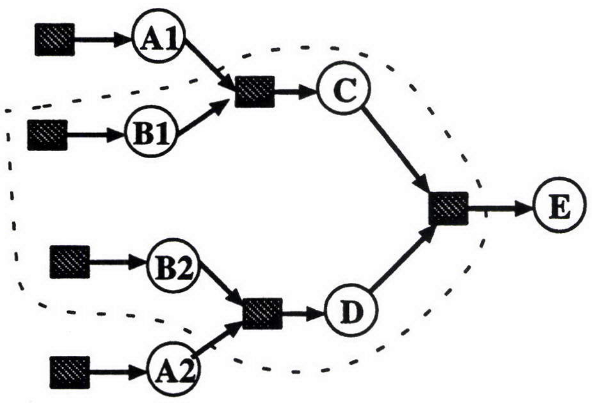

Figure 1b. Model Fragment (replacement RHS)

The real strength of the directed hypergraph representation comes from the fact that graph fragments (corresponding to model fragments) can be represented by directed hyperedges! Figure 1 illustrates this idea. Examine the collection of factors found within the dotted line in Figure 1b. They represent the distribution:

$$p(E|A1, A2) = \sum_{B1,B2,C,D} \cdots \sum p(B1)p(B2)p(C|B1, A1)p(D|B2, A2)p(E|D, C) \tag{2}$$

Thus, the graph fragment within the dotted line is a factor and can be represented by a hyperedge; this is the dotted hyperedge in Figure 1a.

In the model fragment shown in Figure 1, $A1$ and $A2$ are input (or condition) variables and E is an output (or consequence) variable. $C, D, B1$ and $B2$ are interior variables. If there was more model extending beyond $A1$, $A2$ or E, then $A1$ and $A2$ and E would form a Markov boundary for the interior variables: any other variables are independent to C, D, $B1$ and $B2$ given $A1$, $A2$ and E.

2.1 Implicit Independence Assumptions

Each model fragment (or hyperedge replacement rule) contains an implicit independence assumption: **any variables interior to the rule must be conditionally independent of all other variables in the model given the condition and consequence variables.** Placing a model fragment into our library implicitly carries this assumption. Note that this means that all parents of the consequence variables must be in the fragment (at least as conditional variables).

For example, consider two identical power supplies placed in parallel (to improve the reliability of a system). One failure mode for the power supply is to be crushed by a heavy object falling on it (say during an earthquake). If the two power supplies are close together, the same heavy object could crush them both. If the *crushed* variable was on the interior of the model fragment describing the failure of a single power supply, this would result in a unwarranted independence assumption among the *crushed* variables corresponding to the two power supplies.

There is no way to get around the independence assumptions. Instead, we need to carefully check any model to make sure the implicit independence assumptions are reasonable. Such searching for common causes is an important part of the model construction process.

3.0 Hyperedge Replacement Grammars

Habel[1992] defines the concept of a *hyperedge replacement grammar*. The grammar is nothing more than a collection of rules describing when you can replace a hyperedge with a more elaborate graph fragment. The replacement process continues until all non-terminal hyperedges are replaced. In our case, the end result of the replacement process is a belief network. Thus a hyperedge replacement grammar defines a collection of models (a *language* of belief networks).

Figure 1 shows a typical rule in a hyperedge replacement grammar. The rule replaces a non-terminal hyperedge (shaded with dots in the Figure 1a) with a hypergraph fragment with the specified inputs and output (the contents of the dashed line in Figure 1b). The hyperedge being replaced (dotted edge in Figure 1a, left hand side of equation 2) is the *left hand side (LHS)* of the replacement rule and the graph fragment which replaces it (dotted line in Figure 1b, right hand side of equation 2) is the *right hand side (RHS)* of the replacement.

Note that we may want to qualify the applicability of a rule according to the labels of the nodes. For example, if we have a fragment of a reliability model which describes the interaction between a valve and an actuator, we may only want to allow a hyperedge to be replaced if its condition and consequence nodes are an actuator and a valve. We can build this predicate for the rules into the hyperedge label, with a *type signature*. Section 4 develops this idea further.

3.1 Model Construction

We can describe the process of model construction with just two node labels. The directed hypergraph model is fully specified only when there is a valuation (probability distribution, belief function or utility) associated with each hyperedge. We will label hyperedges which have a defined valuation with a "V". Hyperedges which do not yet have a defined valuation, we label with a "?".

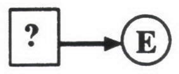

Figure 2a. Initial Graph

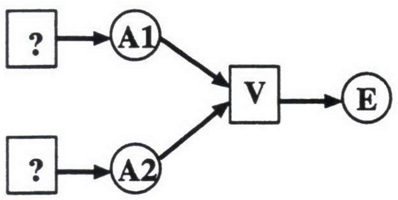

Figure 2b. New graph fragment

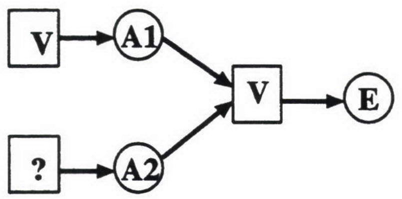

Figure 2c. Set Valuation

Figure 2 shows a typical model construction by successive refinement. Figure 2a is the initial model with the target node E and one undefined hyperedge. In the first step (Figure 2b), we refine the hyperedge by replacing it with a model fragment which contains one defined hyperedge and two undefined hyperedges. In the second step (Figure 2c), we refine the hyperedge for $A1$ by defining its valuation. This process continues until all hyperedges are assigned valuations and we can stop.

Note that sometimes the appropriate input variables will already be in place in the graph, as in Figure 3a. To handle these cases, we need an another graph manipulation which merges repeated variables, as in Figure 3b. For example, in the case of the redundant power supplies discussed in Section 2.1 the *crushed* variable was already present. Forcing the merging operations into the hyperedge replacement framework is rather awkward; it is better to think of it as a separate operation.

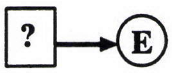

Figure 2a. Graph with Repeated Variable

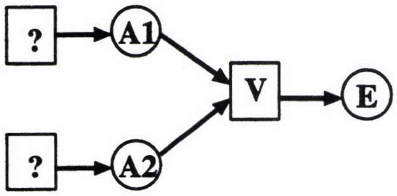

Figure 2b. Variable C Merged

3.2 Collapsed views

Very large graphical belief networks will be difficult to display on a single page or screen of information. In that case the graph must be partitioned into several smaller pieces which represent subsystems of the system being modelled. If the interactions between the subsystems are limited to a few variables, this representation should be simple to work with. The modeller would look at a system level network model where much of the detail about subsystems would be *collapsed* into hyperedges. Expanding the subsystem hyperedge would reveal its graphical structure.

Figure 1 provides an illustration of this idea. Figure 1b is an expanded view of the subsystem which is collapsed into the dotted hyperedge in Figure 1a. Presumably, Figure 1a would be embedded in a much larger graph. Subsystem model graphs could themselves have collapsed hyperedges; this can be carried as far as necessary.

The collapsed hyperedges have an important implication for fusion and propagation as well as for model construction and display. The collapsed hyperedge represents the conditional distribution of the outputs given the inputs (*e.g.*, $p(E|A1, A2)$). This conditional distribution could be precalculated for the collapsed hyperedge. (This is especially easy in the case of probabilistic model, where it is sufficient to condition on each configuration of the inputs and calculate the distribution of the outputs.) If the modeller does not fix the values of any of the interior variables in the subsystem, this cached distribution will not require recalculation.

3.3 Model Uncertainty

Often we will be uncertain about the structure of the model, even if we have built that structure from data. Madigan and Raftery[1994] point out that ignoring that model uncertainty can lead to poorly calibrated predictions; averaging over the models leads to better predictions. Properly accounting for model uncertainty in the general case requires a general representation for the space of possible models.

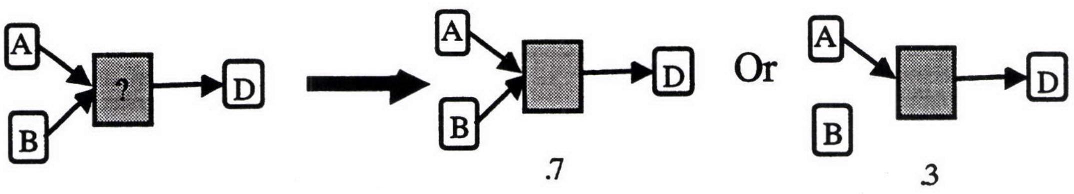

Figure 4. Probabilistic replacement rule for model uncertainty

Hyperedge replacement grammars can be pressed into service once again to represent model uncertainty. Figure 4 shows a probabilistic rule which tries to assess whether or not B should be a parent of D. The hyperedge $A, B \rightarrow D$ on the left hand side has two possible replacements, one in which B is connected to D and one in which it is not. The two possible replacements are assigned probabilities .7 and .3. These probabilities could be updated as more data about B and D make it possible to better assess the relationship.

4.0 Type signatures and candidate replacements

While hyperedge replacement grammars provide a mechanism for constructing models like a jigsaw puzzle, automated and semi-automated model construction procedures require a method for deciding if a particular piece is appropriate for a particular slot. A simple way to do this is to restrict our search based on the types of variables involved in the hyperedge. This defines a type signature for each hyperedge: an edge is appropriate if it matches the type signature. In GRAPHICAL-BELIEF the variable types are based on a variable object system.

4.1 Variable object system

Some model fragments may be more generally applicable than others. For example, a model fragment describing a valve–actuator system might have two input variables, one of type *motor operated valve* and one of type *actuator,* and one output variable of type *valve–actuator system.* A model fragment describing the reliability of the valve brand H24C, might be specific to just that valve brand. A series system model might take two *fail-nofail* variables as inputs and have a system variable as an output.

Valuation	Signature		
Name	Inputs		Outputs
CV-H24C Failure Rate	()	⇒	(Valve CV-H24C)
Valve-Actuator System	(Valve, Actuator)	⇒	(Valve–Actuator-System)
Series System	(Fail-Nofail, Fail-Nofail)	⇒	(Subsystem-Variable)
Valve-Operation	(Valve-fault, Control-Signal)	⇒	(Valve-Status)

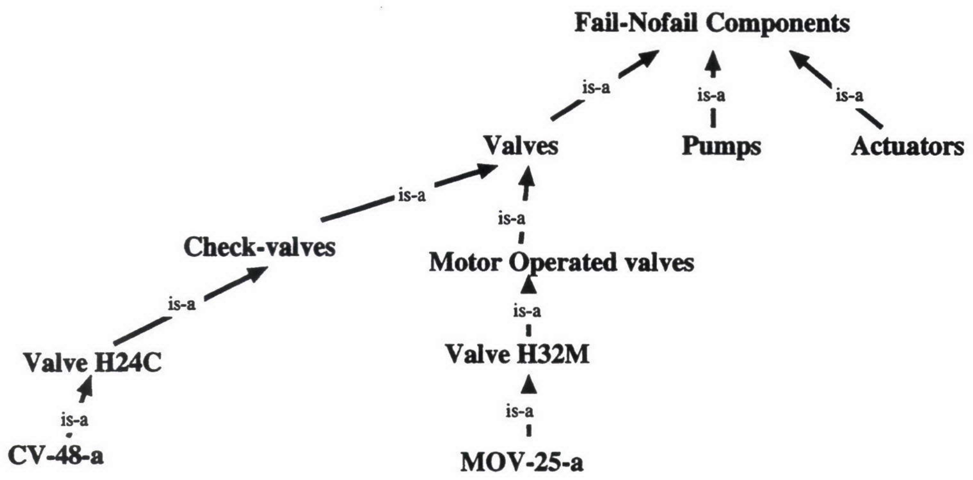

Figure 5. Inheritance graph for parts

To capture the generality, we introduce a class structure among variables. Figure 5 provides an example for variables representing the fault states of valves. At the top we have the most general classes, like *fail-nofail.* Lower down we have more specific classes, like *valve* and *motor operated valve.* The lowest level variables correspond to specific brands of valves and other components.

Going from {`true`, `false`} to a more general outcome space is an example of the *refinement* operation. Shafer[1976] describes the refinement operation in connection with the frame of discernment (outcome space) of a belief function. One variable can inherit from another when the frame of discernment of the one is a refinement of the frame of the other. Note that using the concept of refinement on output variables will lead naturally to belief function models, but applying it to input variables will preserve the probabilistic nature of a Bayesian network or influence diagram model.

One way GRAPHICAL-BELIEF uses this object system is for type checking. Thus if we place a valve–actuator system valuation into the model, GRAPHICAL-BELIEF will only allow an input attachement if it is of type "valve" or "actuator". The computer can also use the type information to figure out the correspondence between its internal table (or graphical model) describing the valuation and the variables.

In order to obtain maximum re-usability we would like the types of the variables in the rules to be as general as possible. Consider the LHS of the replacement in Figure 1 (dotted hyperedge in Figure 1a). It has two inputs: $A1$ and $A2$ and one output E. A rule is a *candidate* replacement for that hyperedge if (1) its input variables are subclasses (refinements) of $A1$ and $A2$ and (2) its output variable is a subclass of E.

As an example, consider a valve modelled by two variables: *valve-fault* and *valve-status*. A valuation (a probability or belief function distribution) which linked the two variables would have one output, of type *valve-status*, and two inputs, one of type *valve-fault* and one of type *control-signal*. Another valuation describing the reliability of the component might have no inputs and one output of type *valve-fault for Valve H32C*. Presumably, the *control-signal* input value would be linked to some other part of the model (*i.e., the actuator*).

Generalizing variables through an object system is one way to extend the scope of a replacement rule beyond its original use. Goldman and Charniak[1993] present an alternative approach to generalizing variables through unification. In their model construction rules, model variables are lists containing semantic variables, for example, `(grass-wet ?date)`. The semantic variables are set when the fragment is placed in the model.

4.2 Component libraries

Almond, Bradshaw and Madigan [1994] propose putting the model fragments into libraries. There are two general types of replacement rules, one in which the RHS is simply a valuation with the given type signature (changing a non-terminal hyperedge to a terminal one) and ones in which the RHS is a graph fragment (possibly containing more non-terminal hyperedges). However, as all replacement rules would be stored sorted by type signature, the modeller can select freely among the various kinds of replacements.

The model component library enables a kind of distributed engineering of graphical belief networks (Figure 5). For example, in building a reliability model, one engineer (a systems level reliability expert) would build small model fragments describing the interaction among a few components. Another engineer (a testing and purchasing expert) would develop model for the various commonly used components from the usual vendors. A third engineer (a system designer) would then assemble the model from the fragments created by the other two engineers, drawing on their expertise in the form of stored model fragments. The type signature system ensures that all model components are used for their intended purpose. As Field test data become available, testing and evaluation engineers update the models and model fragments in the library.

4.3 Knowledge Based Model Construction

Many authors (*e.g.*, Breese, Goldman and Wellman[1991], Goldman and Breese[1992], Goldman and Charniak[1993]) have suggested using a rule base expert system to drive the model construction process. The rules in a hyperedge replacement grammar could form the core of such a system.

In knowledge based model construction, the library of model components becomes a rulebase. In other words, attached to each replacement rule, is a predicate telling when it is appropriate to the problem. The rules would be predicated both on the current structure of the hypergraph (*i.e.*, the type signatures) and on other aspects of the problem to be solved not represented in the graphical model.

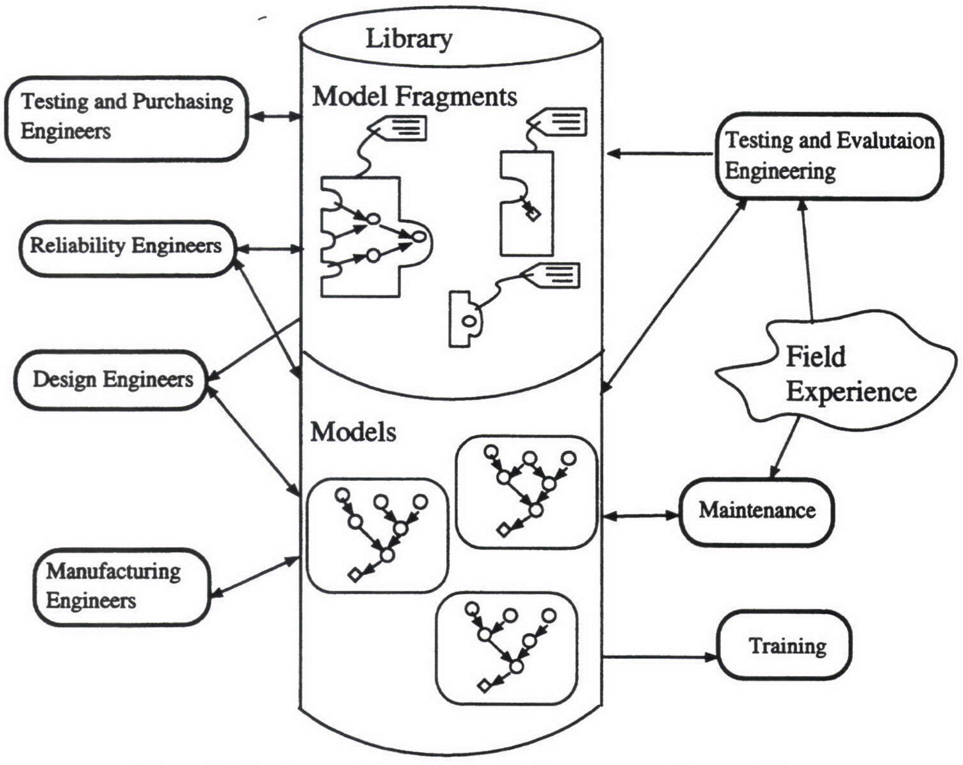

Figure 5. Engineers interacting with GRAPHICAL-BELIEF *library*

Obviously, a replacement must be applicable in the sense of the previous section before it can be used for model construction. The system would select from among applicable replacements on the basis of production rules which would take the context of the particular query into account. For example, it a medical expert system would select between a detailed model of the cardiovascular system and a sketchy model giving a general indication of cardiovascular fitness on the basis of production rules describing whether or not the detailed cardiovascular model would be necessary for answering a particular query.

5.0 Conclusions

Hypergraph replacement rules provide a straightforward way of manipulating both the model graph and the underlying probability distribution. Hyperedge replacement grammars offer a number of exciting possibilities: linking overviews and detailed views of a graphical model (including ways of doing the computations separately), representing model uncertainty—one of the emerging frontiers of belief network research—and the ability to share knowledge, in the form of model fragments and valuations, both in libraries of rules and as more formal expert systems. This knowledge sharing will be essential for large projects within an organization as well as enabling researchers to better share their results.

Goldman and Charniak[1993] describe the FRAIL3 system for knowledge based model construction. They note that each rule in their rule base is equivalent to adding a hyperedge to the graphical model; thus their system is based on hyperedge addition rather than replacement. When two rules both have the same consequence variable, FRAIL3 adds both and combines them with a causal combination function (*e.g.*, noisy-or). The hyperedge replacement approach would first select the meta-rule for combining the information sources (*i.e.*, causal combination function), and then the applicable information rules.

In order to make a replacement rule generally applicable, we must be able to generalize it. The variable object system described in Section 4 is one way of representing generalizations. The marriage of the variable object system and knowledge based model construction forms *object oriented model construction* with many of the advantages of object-oriented programming and design techniques. In particular, we can recognized fragments of models (hyperedges) which can be used again and store these in libraries or rule bases for later re-use when appropriate. The type signatures of the model fragments can be used as a search criteria to find appropriate fragments.

Tossing rules into a rule base expert system with little thought for their interaction often produces unexpected behavior. The same is true for model fragment libraries or rule bases. Each model fragment carries an implicit independence assumption: the variables on the interior of the fragment are independent from the variables not in the fragment given the variables on the boundary (conditions and consequences). Unfortunately these assumptions are difficult to verify within the context of building the library. Unless these assumptions are approximately correct, the resulting model will be unrealistic.

Acknowledgements

I would like to thank Jeff Bradshaw for getting me excited about knowledge based model construction, and David Madigan for listening patiently while I explained how the world could be saved with hyperedge replacement grammars. John Boose and Debra Zarley had helpful comments on an earlier draft of this paper. I would also like to thank Steve Hanks, Chuck Schafer and Thien Nguyen for helping me work out the details of how to apply these ideas to GRAPHICAL-BELIEF.

This research is supported by the GRAPHICAL-BELIEF project, NASA Phase II SBIR contract NAS 9-18908, "Graphical belief model for system reliability."

References

Almond, Russell G., Jeffery Bradshaw, David Madigan [1994]. "Reuse and Sharing of Graphical Belief Network Components." in P. Cheeseman and W. Oldford (eds.) *Selecting Models from Data: Artificial Intelligence and Statistics IV*, Springer-Verlag, 113–122.

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

Egar, J.W., A.R. Puerta, M.A. Musen[1992]. "Graph-grammar assistance for modelling of decision." *Proceedings of the Seventh Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, 7:1–19. Banff, Alberta, Canada.

Habel, A. [1992]. *Hyperedge Replacement: Grammars and Languages.* Springer-Verlag.

Goldman, Robert and John S. Breese[1992]. "Integrating Model Construction and Evaluation" in Dubois *et al.*(eds). [1992] *Uncertainty in Artificial Intelligence, Proceedings of the Eighth Conference,* Morgan Kaufman, 104–111.

Goldman, Robert and Eugene Charniak[1993]. "A Language for Construction of Belief Networks." *IEEE Pattern Analysis and Machine Intelligence* (15) 196–208.

Madigan, David and Adrian Raftery[1994]. "Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam's Window." To appear in *Journal of the American Statistical Association.*

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

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