A Prototype Belief Network-based Expert System Shell

Shijie Wang
(803) 777-2400
wang@cs.scarolina.edu (...!usceast!wang)

Marco Valtorta
(803) 777-4641
mgv@cs.scarolina.edu (...!usceast!mgv)

Department of Computer Science
University of South Carolina
Columbia, South Carolina 29208, U.S.A.

Abstract

A belief network-based expert system works in a way entirely different from a rule-based expert system. In such systems, a high degree of non-determinism is present in the process of belief propagation. Existing belief network-based systems thus provide no control mechanism and let their users make decisions at every stage in the process of evidence gathering and belief propagation. However, for any real world expert system with large complex belief network, it is important to have efficient control of the inference procedure.

In this paper we describe a novel expert system shell that incorporates a general inference control mechanism for efficient belief propagation in belief networks. The control knowledge supplied by domain experts is encoded in some action rules. With these action rules, the system can effectively direct the inference procedure, help the user to gather the most relevant evidence based on the results of previous stages, while the user of the system is still allowed to take the initiative.

1. Introduction

Belief networks have been proposed as a formalism for implementing knowledge bases and also as a computational architecture for reasoning under uncertainty. With respect to the conventional MYCIN-style rule bases, belief networks overcome the problems arising from a truth-functional approach to evidence propagation, by applying a model-based (or intensional) approach, as explained, e.g., in [Pearl, 1988, chapter 1].

A major problem of belief networks is the large amount of computer time that is required to perform computation of beliefs in the worst case [Cooper, 1987]. This led to proposals for approximations, such as the one presented in [Gordon and Shortliffe, 1985]. More recently, various researchers have proposed schemes for efficient computation of beliefs in networks of special structure, such as trees. Examples of these schemes are described in [Pearl, 1986], [Kong, 1986], [Shafer and Logan, 1987], [Lauritzen and Spiegelhalter, 1988], and [Mellouli, 1988].

There are two major classes of belief networks, Dempster-Shafer networks and Bayesian networks. The body of this paper describes results pertaining to Dempster-Shafer networks, but much of the presentation applies to Bayesian networks as well.

At the present time, only few applications of belief networks and very few tools for their construction have been developed. For Dempster-Shafer belief networks, there is an experimental system called DELIEF that implemented an efficient belief propagation scheme based on qualitative Markov trees (see [Shafer et al., 1987], [Mellouli, 1988] and [Zarley, 1988]). For Bayesian belief networks, the authors know of only one commercially announced tool for the construction of belief networks, the expert system shell HUGIN [Andersen et al., 1989], and only one commercially available application, a medical assistant developed at Stanford. A large, but not commercially available, application of belief networks is the MUNIN[1] system [Andreassen et al., 1987]. MUNIN assists a physician in the diagnosis of muscle diseases using electromyography. HUGIN is approximately in the same relation to MUNIN as EMYCIN is to MYCIN. Currently, MUNIN has approximately 1000 nodes, most corresponding to findings, pathophysiological states, and diagnosis. MUNIN covers only a small portion of the domain of electromyography.

[1] The second author has maintained a loose association with the MUNIN project since the fall of 1985.

The authors have started a program of research in the fall of 1988 with the overall objective of investigating the practical applicability of Dempster-Shafer belief networks as an alternative to rule bases. Theoretical results achieved so far concern the construction and maintenance of belief networks [Valtorta and Loveland, 1989] and the computation of beliefs in graphs that are not trees [Wang, 1989, Chapter 3; Wang and Valtorta, 1990a]. Practical results include the conversion of parts of a large rule base into a belief network [Wang, 1989, Chapter 5; Wang and Valtorta, 1990b], and the construction of an expert system shell for belief network-based expert systems. This shell, called BELFUN[2], will be described in this paper.

BELFUN has been used to re-implement part of the knowledge base used in the PLAYMAKER expert system, a component of XX, an intelligent support system for the exploration geologist under developmen at the University of South Carolina and Vanderbilt University with the support and involvement of several oil companies [Biswas and Anand, 1988]. We decided to use an existing rule base to simplify the knowledge acquisition phase, but this choice forced us to confront certain specific problems of the conversion of rules into belief networks. These problems will be reported elsewhere.

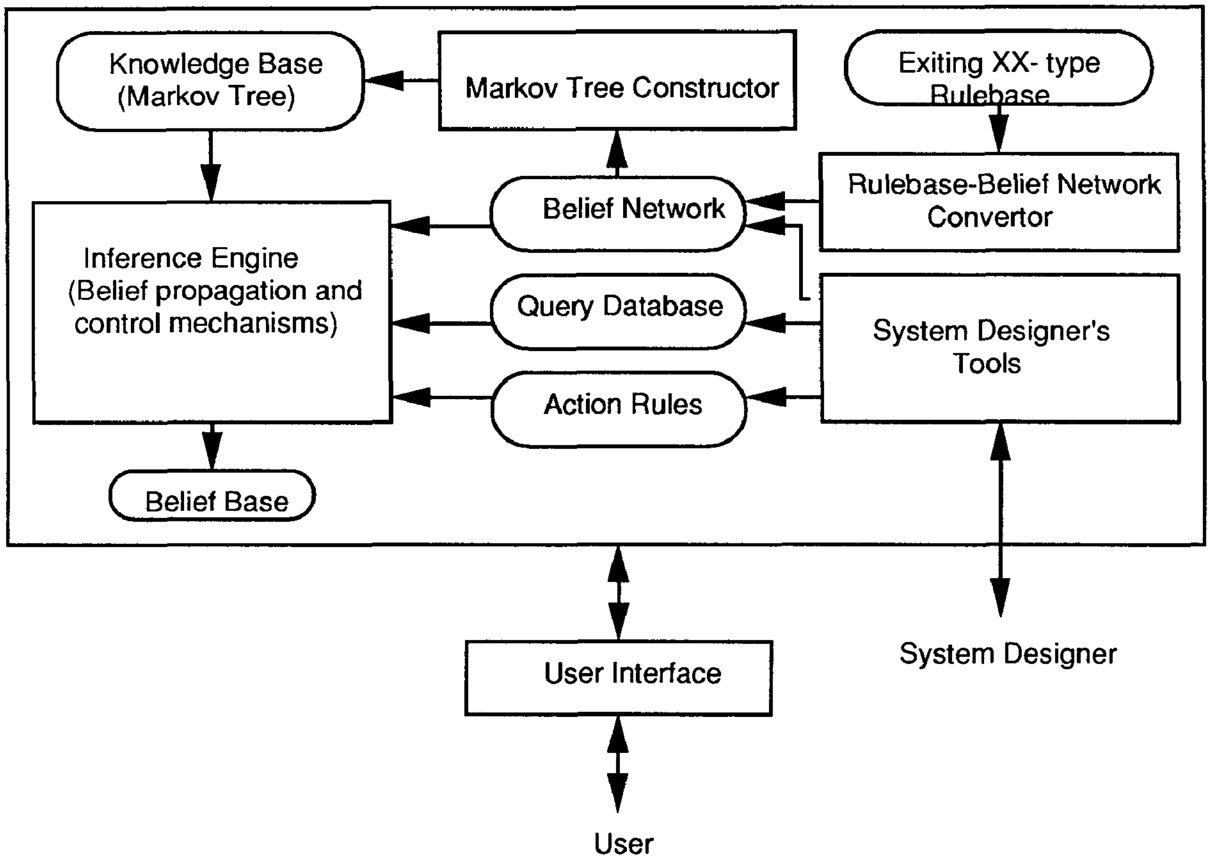
2. BELFUN System Architecture and Knowledge Base Construction

BELFUN incoperates the Dempster-Shafer theory of belief functions, belief propagation schemes, Markov tree construction procedures, and some useful techniques adapted from MIDST [Biswas and Anand, 1988], MYCIN [Shortliffe, 1976], DELIEF [Zarley, 1988], and CONVINCE [Kim and Pearl, 1987] into a complete evidential reasoning system. The system is implemented in AKCL (Austin-Kyoto Common LISP) on a Sun3/60 workstation running SUNOS UNIX[3] 4.0. Figure 1 gives an overview of the system architecture, where program modules are represented by rectangular boxes, data structures are shown in rectangles with rounded corners, while arrows indicate information flow in the system.

The system runs in system designer mode or user mode. The system designer mode is provided for knowledge engineers to construct, test, and modify the belief network, query database [Biswas and Anand, 1988], and control action rules. The user

[2] BELFUN stands for BELief FUNctions.

[3] Trade marks of Sun Microsystems and AT&T, respectively.



Knowledge Base (Markov Tree)
Markov Tree Constructor
Exiting XX- type Rulebase
Belief Network
Rulebase-Belief Network Convertor
Inference Engine (Belief propagation and control mechanisms)
Query Database
System Designer's Tools
Action Rules
Belief Base
User Interface
System Designer
User

Fig. 1 BELFUN System Architecture

mode is designed for users who need experts' consultations on some particular problems in the domain of application.

Currently, very limited facilities are available to knowledge engineers. We have implemented a simple menu driven system that allows the knowledge engineer to run some sessions for the purpose of testing or demonstration. In the user mode, we adopt a goal-directed and mixed-initiative evidence aggregation mechanism. Interaction between the system and the user is guided by a monitor working with some action rules and some queries supplied by domain experts. However, the inference procedure can be altered by the user if he wants to take the initiative. Currently, the user interface is a simple dialogue system. Later, it will be extended to provide a user-friendly multi-window graphical working environment.

A belief network constructed from expert's knowledge about a domain of application may not possess the Markov property of qualitative independence (original definitions of found in [Shafer, Shenoy and Mellouli, 1987], [Mellouli, 1988]). A major task of our system is to transform an arbitrary belief network into a qualitative Markov tree which will serve as the knowledge base of the expert system under construction. This complex task is accomplished by the module *Markov tree constructor*. A series of transformations is performed, including (1) representing the belief network as a hypergraph and reducing it to a *non-reducible block* (usually non-empty) and constructing a *Markov tree* (usually incomplete) at the same time; (2) constructing a maximal intersection network for the block (if it is not empty) and finding a Markov tree representative that is then merged with the partial Markov tree obtained in (1) to form a complete Markov tree; (3) finding a partially fused tree and constructing an *alternating tree* based on that; and (4) attaching to the alternating tree those individual nodes still absent, and finally copying belief functions from the original belief network and creating vacuous belief functions for newly-generated nodes. For details of the construction procedure, see [Wang, 1989], [Mellouli, 1988], [Zhang, 1988] and [Zarley, 1988].

3. BELFUN Inferencing and Control Structure

A fully-specified belief network contains all information necessary to answer all questions about all individual and joint variables in the belief function model, either directly by user-supplied evidence or indirectly through propagation (cf., e.g., [Pearl, 1988]). The belief network can be viewed as a network of autonomous processors that work in parallel and communicate only with immediately neighbors, thus no global control mechanism is needed in principle. In any real-world expert system, however, careful design of global control mechanism is crucial, and in fact control knowledge and the partitioning of a knowledge base are important components of human expert's knowledge (cf. [Shafer et al., 1987], [Valtorta et al., 1984]). Therefore, BELFUN includes efficient control mechanisms suitable for belief network-based systems working with any monolythic belief network or partitioned belief networks.

3.1 Design Considerations for Inference Control Structure

In rule-based systems like MYCIN and MIDST, a rule is fired when relevant evidence has been gathered so that all conditions in that rule are satisfied and the rule is selected by some conflict resolution strategy, therefore, domain knowledge as well as control knowledge can be represented using the same formalism of production rules. Firing a rule may then cause some conclusions to be made and/or some actions to be taken, say, request for evidence about some particular variable, which may in turn cause some other rule(s) to be fired, thus forming chains of firing rules (either forward chaining or backward chaining). The inference procedure in such systems is thus primarily controlled by the system.

A belief network-based system works in a way entirely different from a rule-based system. During the process of belief propagation the order in which a node in the belief network is reached could be arbitrary, i.e. independent from the particular values of input evidence, as long as the belief network is a Markov tree. The user of the system can thus have the full freedom to direct the inference procedure in his own way. DELIEF and HUGIN, the two existing belief network-based systems, take this approach.

In DELIEF, a typical session proceeds in three stages [Zarley, 1988]. The user or knowledge engineer can construct a belief function network through a graphical interface. Upon completion, the belief network is transformed into a Markov tree. The resulting Markov tree is a rooted directed tree where the directions are simply a byproduct of the construction procedure. After the user has provided some evidence pertinent to a specific situation, he can simply select the main menu item "propagate," and the inference engine is activated to propagate the impact of new evidence throughout the network by first propagating from the leaves towards the root and then from the root towards the leaves. This procedure is called *complete propagation* . The system has some facilities for the user to examine the updated belief distributions. If later some new evidence is obtained, the propagation procedure is repeated. Obviously, much repetitive computation is involved in this iterative process.Also, the order for gathering evidence is solely up to the user and the system is merely a passive tool for computing beliefs.

Similarly, HUGIN's inference procedure is also totally controlled by the user although it can provide more sophisticated facilities for the user to do that [Andersen et al., 1989]. At any stage, the user can choose a particular node and provides some evidence to that node, and then the procedure *DistributeEvidence* is invoked to propagate its impact to the whole network. Alternatively, the user can specify a particular node of interest as the destination and calls the procedure *CollectEvidence* to propagate beliefs in the whole network towards this node. When multiple pieces of evidence are input to the system, a call to *CollectEvidence* followed by a call to *DistributeEvidence* performs a complete propagation as in DELIEF. Here the problem is that the user is supposed to be a non-expert who also needs help in collecting evidence, not just in deriving conclusions, especially when working with some large belief network like MUNIN, derived from complex problem domains. An efficient automated reasoning system should be able to take appropriate actions at proper times, guide the user to gather most relevant evidence in every stage, based on the outcome of previous stages. DELIEF and HUGIN apparently lack such a desirable property.

Based on these observations and arguments, we have designed an efficient control mechanism for BELFUN that incorporates domain experts' control knowledge. In a belief network-based system, evidence and rules are all cast into the conceptual framework of belief functions. Hypotheses, conditions and conclusions are all in the form of (variable value) with associated belief value and represented by a basic probability assignment. Clearly, actions do not fit in this formalism. Therefore, it is inappropriate to encode action rules as belief functions stored in some nodes of belief network. Our solution is to adopt such a strategy that control knowledge is represented by various action rules, separated from domain knowledge proper.

3.2 Goal-Directed Evidence Aggregation in Partitioned Belief Networks

Any monolithic belief network can be viewed as a special case of partitioned belief networks. We will discuss the control structure for the general case of partitioned belief networks.

In any partition of a belief network, we can always identify a variable as a goal variable. In general, there can be more than one such variable. For simplicity, we consider the case of single goal variable as is the case of the XX belief network. A goal variable can be, for example, the one used in action rule(s) for partition transfer (see Fig. 2), that is, we have to get a satisfactory amount of belief in a

```
(setq *action_rules*
   '( (rule01  ( (*current_partition* dep_set) )
               ( (askfirst nil) )
      )
      (rule02  ( (max_hypothesis (source_seismic yes) )
                 (max_hypothesis (basin_margin crossed) )
               )
               ( (askq play_lies) )
      )
      (rule03  ( (max_hypothesis (source_seismic yes) )
                 (max_hypothesis (basin_margin not_crossed) )
                 (max_hypothesis (basin_margin_shelf penetrated) )
               )
      )
      . . .
      (rule07  ( (max_hypothesis (depo_set shelf) )
                 (*exitcheck* yes) )
               )
      )
   )
)
(setq *goal* 'depo_set) )
```

Fig. 2 The action rules and the goal variable for the partition "general depositional settings."

hypothesis about this variable before the system descends to next partition, and its different values will lead the system to different partitions accordingly. Evidence aggregation and belief propagation are directed to achieve this goal.

For any partition of a belief network, there will be a (usually small) set of action rules. In our system, we have three types of action rules that are closely related to the three types of actions in the XX system (for more details, see [Biswas and Anand, 1988]). Fig. 2 shows some of the action rules associated with the partition "general depositional settings" of the XX rulebase, where *max_hypothesis* is a predicate to test whether or not the hypothesis given as its argument is the leading hypothesis about this variable, that is, whether or not it has the largest belief value. Other predicates can be used if appropriate. For example, when constructing a belief function network from MYCIN-type rulebase, some predicates used in these rules, say *KNOWN, NOT-KNOWN, DEFINITE-NOT* , might be encoded in action rules (see [Buchanan and Shortliffe, 1984]).

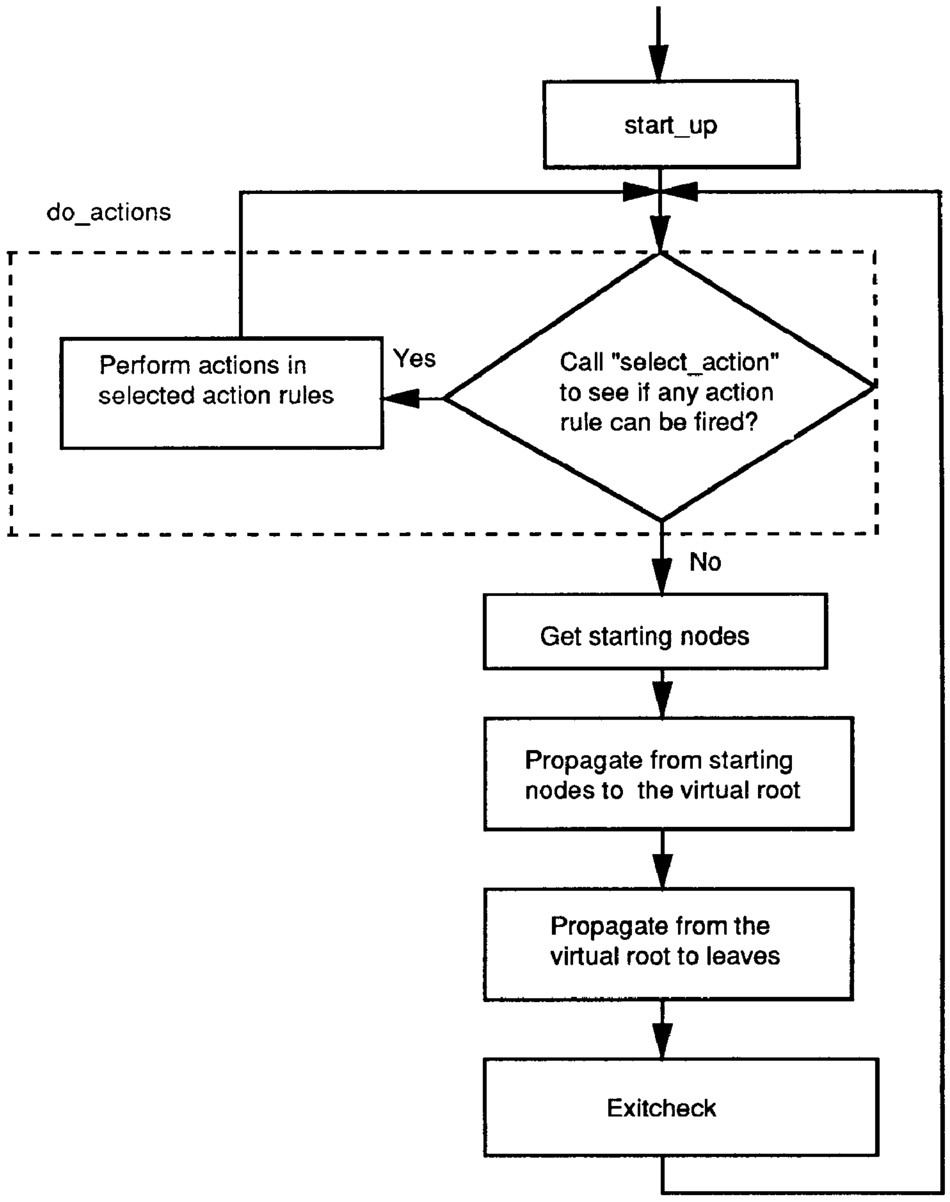
Askfirst rule There is only one such rule for a partition. It is always invoked at the very beginning upon entering a partition. The special variable **current_partition** holds the name of the currently active partition. The action routine *askfirst* takes as argument a variable name and presents to the user a specific query about this variable along with possible answers which are provided by domain experts and stored in a query database (see Fig. 1). If a null argument is given, *askfirst* will encourage the user to enter whatever evidence the user considers relevant to the current goal. The propagation process will start from those nodes for variables involved in this action and action(s) caused immediately by this action, that is, those nodes with user-supplied evidence.

Askq rule There can be several action rules of this type. Each may invoke the action routine *askq* several times. Each time, *askq* asks the user a question about the variable given as its argument. The user is allowed to input evidence about other variables that he considers more relevant to the current goal; and nodes for these variables will be put on the top of the list of nodes for propagation next step. In this way, the user takes the initiative to adjust the direction of the propagation procedure. Thus, the approach of mixed initiative reasoning in MIDST is adapted to our belief network-based system except that no backward chaining process is involved.

Next_partition rule This type of rule is designed to control transfer from the current partition to some other partition according to some particular conditions. Such a rule will always be the last one to fire. For a monolithic belief network or the last partition in a partitioned belief network, a null argument will be given to the action routine *next_partition* , thus bringing the inference procedure to an end. When such a rule is fired, the action routine *next_partition* first saves the current status of the belief network, that is, the resulting belief distribution obtained through evidence aggregation and belief propagation in all previous partitions (if any) as well as the current partition, which are the contents of a global working memory called *belief base* . It then calls a procedure to set-up for working with the next partition. This includes loading the belief network (a Markov tree) and associated action rules, and merging the contents of the belief base into the belief network. For each node in the belief network, it creates slots for accommodating inputs from user, projections from neighbors and storing updated belief functions, and finally, making the next partition the value of the special variable **current partition** so that the *askfirst* rule for the new partition will be fired immediately.

Note that **exitcheck** is another special variable in the system. When all available evidence relevant to the current partition has been collected, and its impact has been propagated throughout the network, a procedure named *exitcheck* is called. It first shows the user the current status of the goal variable, all *next_partition* rules and status of each condition in these rules along with expert suggested thresholds. If the user is satisfied with these results, **exitcheck** is set and then an action for partition transfer will take place next. However, if the user is not satisfied yet, he can supply additional information to the system and the propagation procedure will be repeated starting from those nodes with new evidence. When no more evidence is available, if the user is still not satisfied, or for any other reasons, he may suspend the current session and later come back with some newly acquired evidence to resume the session at this point. Note that the procedure *exitcheck* can provide some thresholds suggested by domain experts, which can be used as default thresholds, while the user has the final say to decide whether the exit condition is satisfied or not. We can say that the thresholds are adaptively set by the user. This strategy is certainly justifiable since in many practical situations, the user may not have a sufficient source of information or enough time to acquire adequate evidence, but still wants to obtain some approximate results based on evidence he can supply from a limited source or within limited time.

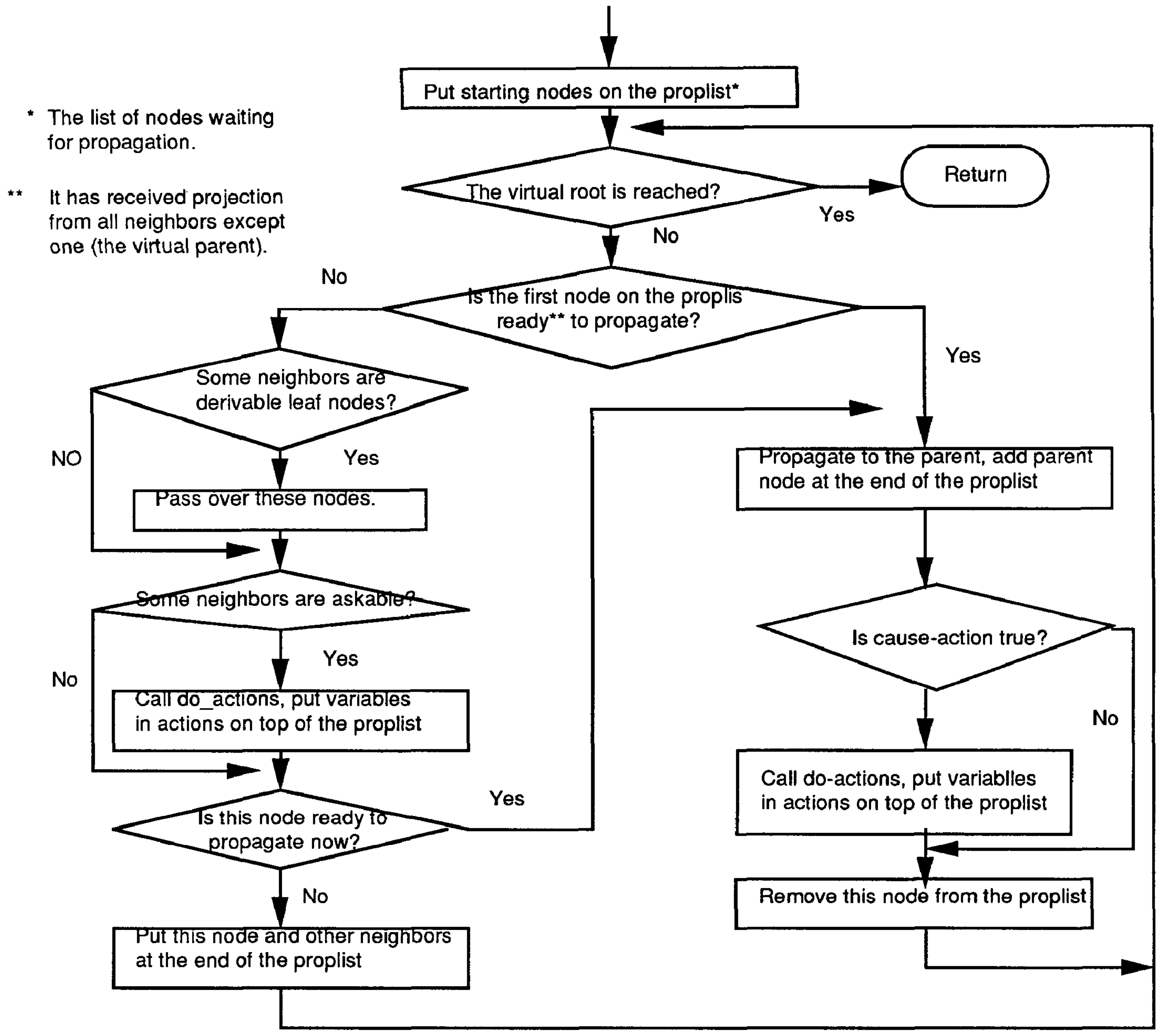
The control structure is illustrated in Fig. 3. The *start-up* procedure is executed to begin a new session or resume a suspended session. This procedure first asks the user the entry point partition and then calls


start_up
do_actions
Perform actions in selected action rules
Yes
Call "select_action" to see if any action rule can be fired?
No
Get starting nodes
Propagate from starting nodes to the virtual root
Propagate from the virtual root to leaves
Exitcheck

(a) Overall Inference procedure

Fig. 3 The inference/control structure in BELFUN

the set-up procedure to do all necessary initialization work mentioned above in the description of *next_partition* rule . The procedure *do_actions* calls the procedure *select_actions* and then performs actions in selected action rules. Selection of appropriate action rules is based on the current status of the belief base and the values of the two special variables **current_partition** , and **exitcheck** As consequences of some actions, the *belief base* and the two special variables may be changed. Therefore, some other actions may need to be taken next. In action rules, conditions on these two variables involve no uncertainty while a condition of other types is satisfied if the hypothesis in the condition is the leading hypothesis for the variable concerned. Intuitively, belief propagation should start from those nodes that have non-vacuous belief functions, either inherited from previous partition (if any) or obtained directly from user supplied evidence. We give the nodes in the latter case higher priorities for propagation. Normally, the inference procedure is completed when a partition transfer rule is fired for a monolithic belief network or for the last partition of a partitioned belief network. Note that any node can be the root of the belief network (Markov tree) depending on the order that evidence is supplied. The resulting belief distribution will be the same no matter which node is selected as the root. The parent-child relation between



Put starting nodes on the proplist*
* The list of nodes waiting for propagation.
** It has received projection from all neighbors except one (the virtual parent).
The virtual root is reached?
Yes
Return
No
Is the first node on the proplis ready** to propagate?
No
Yes
Some neighbors are derivable leaf nodes?
NO
Yes
Pass over these nodes.
Some neighbors are askable?
No
Yes
Call do_actions, put variables in actions on top of the proplist
Is this node ready to propagate now?
Yes
No
Put this node and other neighbors at the end of the proplist
Propagate to the parent, add parent node at the end of the proplist
Is cause-action true?
No
Call do-actions, put variablles in actions on top of the proplist
Remove this node from the proplist

Fig. 3 (b) Propagate to the virtual root.

adjacent nodes is dynamically established and no longer exists after propagation towards leaves passed the nodes. When propagating towards the root, if a node has received projections from all its neighbors except one, then that one is said to be its parent node while the others are its children. We borrow these terms for rooted directed trees only for convenience in our descriptions.

We say a variable is *askable* if its value can be obtained directly from user input. Such a variable appears in some *askq rule(s)* . A variable is said *derivable* if its value can only be derived through belief propagation (it is called *verifiable* in MIDST). During the main process for evidence gathering (propagation towards root), when propagation reaches such a node, it still has vacuous belief function. If a node has some neighbors that are derivable leaf nodes, propagation will pass over this node by simply putting them in its children list as if they had actually sent messages to it. No actual projection is needed in this case since projection of a vacuous belief function always results in a vacuous belief function. If some of its neighbors are askable (usually also leaf nodes), *do_actions* is called to see if conditions in action rules about the variable of this node are satisfied or not based on the current status of the belief base. If the conditions are satisfied, take

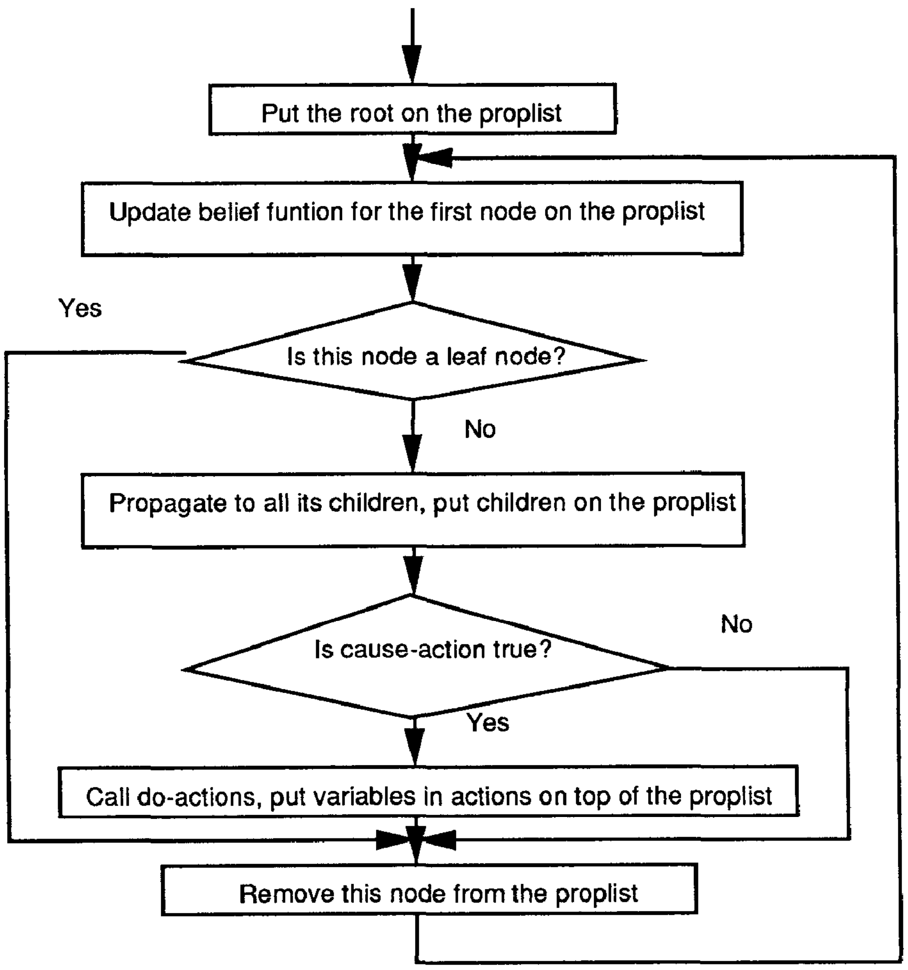

Put the root on the proplist
Update belief funtion for the first node on the proplist
Yes
Is this node a leaf node?
No
Propagate to all its children, put children on the proplist
No
Is cause-action true?
Yes
Call do-actions, put variables in actions on top of the proplist
Remove this node from the proplist

Fig. 3 (c) Propagate to leaves.

the actions specified in the action rule (normally *askq* rule in this case). The attribute *cause_action* of a variable is set to true if some actions are associated with it. Such a variable appears in the condition part of some action rule(s). After the action in the action part have been taken, it is reset so that the same action rule will not fire twice. Therefore, if the attribute *cause_action* of a variable is true, then this variable can trigger some action but the action has not yet been taken. If an action depends upon some derivable variable, the action might not be triggered until propagation towards leaves reaches the node of this variable, since only at that time can this node receive projection from its parent node so that it can update its belief function. The projection received from the parent node reflects the accumulated impact of all evidence input to the other part of the network, that is, the subnetwork containing its parent node obtained by breaking the link between this node and its parent node.

In the current implementation of BELFUN, each condition in an action rule fails immediately if the variable in the condition does not have a computed belief value. Conditions in action rules cannot initiate a belief computation. They can only test belief values. For example, with reference to rule03 in Figure 2, if no belief value has been computed for the variable *basin_margin*, the condition *(max_hypothesis (basin_margin not_crossed))* fails, and the rule will not fire. (Of course, the condition will also fail if *not_crossed* is not the value of *basin_margin* with the highest belief.) In some applications, we may need a more sophisticated procedure that tries to satisfy all conditions in some action rule about a derivable variable. The procedure will probably be engaged in recursive calls in a way similar to backward chaining in rule-based systems since whether or not a condition is satisfied may in turn depends upon conditions in some other action rules.

It should not be hard to see that most evidence relevant to the current partition is collected during the process of propagation towards the virtual root. In the stage of propagation from the root towards leaves, updated beliefs are distributed throughout the belief network. In most cases, we can reasonably assume that at this point, the exit condition could have been satisfied, although the same procedure can be repeated if it is not so. The time needed for completion of one cycle in terms of projection operations is clearly proportional to the length of the longest path in the network. This control structure is efficient in view of the fact that propagation at every node in the network is done always at the right time and repetitive propagation is minimized only subject to the availability of evidence.

4. Conclusions

We conclude the paper by outlining a direction for further work that would enhance the effectiveness of BELFUN as a tool for the construction of expert systems.

There is an essential difference between inference procedures in a rule-based system and in a belief network-based system. In a rule-based system, for any specific situation, there is only a particular subset of rules in the knowledge base that successively fire in some order to obtain the desired results. In a belief network-based system, no matter what kind of propagation procedure is employed, we always have to propagate all evidence throughout the belief network (knowledge base). The problem domain under consideration may be too large and too complex to propagate all beliefs. Also, it may not even be necessary for practical reasons to propagate beliefs throughout an entire belief network because of the knowledge structure of the problem domain. Therefore, partitioning a large belief network is desirable when dealing with a complex problem domain, so that each time we only need to propagate beliefs within a bounded area of a belief network. Even with a partitioned belief network, the user may be interested only in propagating beliefs from some particular source nodes to some specified target nodes. In the stage of knowledge acquisition and knowledge base construction, the knowledge engineer or domain expert often likes to see the varying aspects of some evidence upon some particular variable(s) closely related to the observation. He should be allowed to switch from editing mode to a mode where he can run a test efficiently on a small subnetwork only and then switch back to make necessary modifications based on the results of the test (cf. [Biswas and Anand, 1988]). Also, in this way, the system can help the system designer to partition a large belief network in a more appropriate way. More generally, in any problem domain involving simulation, prediction, and planning, the capability of a knowledge-based system to perform "what-if analysis" is a most desirable property (cf. [Kim and Pearl, 1987]). We are therefore investigating a mechanism for bounded belief propagation. We are also looking for other suitable domains of application for BELFUN.

References

Andersen, S.K., K.G. Olesen, F.V. Jensen, and F. Jensen. "HUGIN--A Shell for Building Bayesian Belief Universes for Expert Systems." *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1989, 1080-1085.

Andreassen, S., M. Woldbye, B. Falck, S.K. Andersen. "MUNIN--A Causal Probabilistic Network for Interpretation of Electromyographic Findings." *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987, 366-372.

Biswas, G. and T.S. Anand. "An Expert System Shell for Mixed Initiative Reasoning." Technical Report CS-88-02, Department of Computer Science, Vanderbilt University, 1988.

Buchanan, B.G. and E.H. Shortliffe (eds.) *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Reading, MA: Addison-Wesley, 1984.

Cooper, G.F."Probabilistic Inference Using Belief Networks is NP-Hard."Stanford University Knowledge Systems Laboratory Memo KSL-82-27, May 1987 (revised July 1988).

Gordon, J. and E.H. Shortliffe. "A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space." *Artificial Intelligence*, 26 (1985), 323-357.

Kim, J.H. and J. Pearl. "CONVINCE: A Conversational Inference Consolidation Engine."*IEEE Transactions on Systems, Man, and Cybernetics*, 17, 2, 120-132, 1987.

Kong, C.T.A. "Multivariate Belief Functions and Graphical Models." Ph.D. Dissertation, Department of Statistics, Harvard University, 1986. (Available as Research Report S-107, Department of Statistics, HarvardUniversity.)

Lauritzen, S.L. and D.J. Spiegelhalter. "Local Computations with Probabilities on Graphical Structures and their Applications to Expert Systems." *Journal of the Royal Statistical Society, Series B (Methodological)*, 50 (1988), 157-224.

Mellouli, K. "On the Propagation of Beliefs in Networks Using the Dempster-Shafer Theory of Evidence." Ph.D. Dissertation and Working Paper No. 196, School of Business, University of Kansas, April 1988.

Pearl, J. "On evidential reasoning in a hierarchy of hypothesis." *Artificial Intelligence*, 28, 9-15, 1986.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan-Kaufmann, 1988.

Shafer, G. and R. Logan. "Implementing Dempster's Rule for Hierarchical Evidence." *Artificial Intelligence*, 33, 271-298, 1987.

Shafer, G., P.P. Shenoy, and K. Mellouli. "Propagating Belief Functions in Qualitative Markov Trees." *International Journal of Approximate Reasoning*, 1, 349-400, 1987.

Shortliffe, E.H. *Computer-Based Medical Consultations: MYCIN*. New York: American Elsevier, 1976.

Valtorta, M. and D.W. Loveland. "On the Complexity of Belief Network Synthesis and Refinement." Technical Report TR89011, Department of Computer Science, University of South Carolina, Columbia, November 1989.

Valtorta, M., B.T. Smith, and D.W. Loveland. "The Graduate Course Advisor: A Multi-Phase Rule-Based Expert Systems." *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, Denver, CO, 1984, 53-57.

Wang, S. "BELFUN -- A Belief Function Based Expert System Shell." M.S. Thesis, University of South Carolina, Columbia, November 1989.

Wang, S. and M. Valtorta. "An Extended Algorithm for Markov Tree Construction and Nonserial Dynamic Programming." In preparation. To appear in 1990.

Wang, S. and M. Valtorta. "On the Conversion of Rule Bases into Belief Networks." In preparation. To appear in 1990.

Zhang, Lianwen. "Studies on finding hypertree covers for hypergraphs." Working Paper No. 198, School of Business, University of Kansas, Lawrence, KS, 1988.

Zarley, D. "An Evidential Reasoning System." Thesis, School of Business, University of Kansas, Lawrence, 1988.