Constructing Flexible Dynamic Belief Networks from First-Order Probabilistic Knowledge Bases

Sabine Glesner and Daphne Koller

Computer Science Division, University of California, Berkeley, CA 94720,
{glesner,daphne}@cs.Berkeley.EDU

Abstract. This paper investigates the power of first-order probabilistic logic (FOPL) as a representation language for complex dynamic situations. We introduce a sublanguage of FOPL and use it to provide a first-order version of *dynamic belief networks*. We show that this language is expressive enough to enable reasoning over time and to allow procedural representations of conditional probability tables. In particular, we define decision tree representations of conditional probability tables that can be used to decrease the size of the created belief networks. We provide an inference algorithm for our sublanguage using the paradigm of *knowledge-based model construction*. Given a FOPL knowledge base and a particular situation, our algorithm constructs a propositional dynamic belief network, which can be solved using standard belief network inference algorithms. In contrast to common dynamic belief networks, the structure of our networks is more flexible and better adapted to the given situation. We demonstrate the expressive power of our language and the flexibility of the resulting belief networks using a simple knowledge base modeling the propagation of infectious diseases.

1 Introduction

In recent years, *belief networks* (or *Bayesian networks*) [10] have emerged as the method of choice for reasoning under uncertainty. Belief networks utilize the underlying causal structure of the domain to make probabilistic reasoning practical, both in terms of knowledge representation and in terms of inference. The causal structure is represented graphically, as a directed acyclic graph. Each random variable of the modeled domain is represented by a node while edges represent direct causal influences between nodes. The topology of the network encodes a set of independence assumptions: that each node is independent of all its nondescendants given its parents. These independence assumptions allow us to concisely specify a joint probability distribution over the random variables in the network. It suffices to specify, for each node in the network, a *conditional probability table (CPT)*, which lists the probability that the node takes on each value given each possible combination of values for its parents. The structure of the network can also be used to make the probabilistic inference process more efficient. For example, the *join tree* algorithm, which is also used in our implementation, is a clustering method that has been implemented in the HUGIN system [1].

In spite of their success, belief networks have a significant limitation as a general framework for knowledge representation, since they lack the ability to generalize over (a potentially infinite set of) individuals. In fact, belief networks are essentially a probabilistic extension of propositional logic, and therefore do not even have the fundamental

concepts of individuals, their properties, and the connections between them. In this paper, we use the theory of first-order probabilistic logic (FOPL) to overcome these difficulties. We show how FOPL can be used as a representation language for concisely describing *first-order belief networks*. We describe a sublanguage of FOPL, built over Prolog, and demonstrate its expressive power. In particular, we present a knowledge base modelling the propagation of infectious diseases. An example of another knowledge base, modeling the behavior of traffic, can be found in [4].

We present an inference algorithm for this language using the paradigm of *knowledge-based model construction*. This algorithm takes a knowledge base in our language and a particular situation and generates a standard belief network appropriate for this situation. More specifically, the algorithm takes a particular finite domain as input, and propositionalizes the knowledge base for that domain. It also takes a query Q and evidence E, and constructs a belief network with an appropriate structure. Standard belief network inference algorithms can then be used to derive probabilities for Q given E in the resulting network.

The paradigm of knowledge-based model construction from first-order schemata has been used before [2, 3, 5, 8, 6] (see also Sect. 2). There are a number of ways in which our approach extends some or all of these previous works.

First, our work focuses on dynamic domains, where we wish to represent a situation that changes over time. The standard approach for modeling such domains is using *dynamic belief networks*. The fundamental idea is to divide time into *time slices*, each representing the state of the world at a specific point in time. This state is described using some set of random variables; in general, each time slice in the network contains a copy of all these variables. Such a network is typically represented by describing the probabilistic relations between random variables in one time slice and the probabilistic relations between the random variables of two consecutive time slices. These are taken to be the same for all time slices. We use our first-order representation language to model these dynamic situations. Therefore, the networks resulting from our knowledge-based model construction process are *flexible dynamic belief networks* whose structure differs from one time slice to the other according to the given situation.

Our algorithm is based on the one given in [6]. There, the representation language does not allow a modular specification of the different influences on an event. For example, when modeling the propagation of infectious diseases, we want to specify the probability of infection from different sources separately, and then combine them as appropriate. As in [8, 5], our representation language allows a modular specification of the influences on each event in our system. These are then automatically combined by our system, using a combination function specified by the user. Typical combination functions are noisy-or, noisy-and, min, or max.

We also use the power of a first-order language to allow a compact functional representation of probabilities. Among other things, this allows us to represent continuous-valued variables in our framework. For example, we could have the location of a car be a Gaussian random variable whose mean depends on the location and velocity of the car at the previous time slice. Our first-order language is also rich enough to allow an easy specification of parameterized probabilities. For example, we could have the variance of this Gaussian depend on the width of the time slice, thereby allowing us to model the

fact that the accuracy of our predictions decreases over time. The first-order character of the representation language can also be used to specify that certain variables should only appear in certain time slices. For example, we might have a variable denoting the weather appear less often (say once an hour). This can easily be specified in our language, and our algorithm will automatically generate networks with the appropriate structure.

The first-order character of our language can be used to formulate conditional probability tables very concisely. In particular, we can describe the CPT as a decision tree, thereby modeling situations where one variable affects another only in certain circumstances. For example, in certain circumstances the event "Person x has AIDS" might directly affect the event "Person y has AIDS", whereas in others it does not. Such a representation can be used by a knowledge-based model construction algorithm to produce much smaller networks tailored to specific circumstances.

2 FOPL and KBMC

The approach of first-order probabilistic logic puts a probability distribution on the set of possible worlds where a possible world is a model in the sense of classical first-order logic. In this framework it is possible to assign *probabilities* or *degrees of belief* to formulas which are true in some but not necessarily in all of the possible worlds. The degree assigned to a formula φ is the sum of the probabilities of those worlds in which φ is true. For the formal definitions we refer to [7]. FOPL is highly undecidable [7] but, as in first-order logic, it is possible to isolate interesting tractable cases. In the paradigm of *knowledge-based model construction*, systems rely on a representation language based on FOPL; they take a knowledge base written in this language and reduce it, given a particular example (or instance), to a propositional model, mostly specified in form of a belief or decision network. This approach has been successfully used in various systems [2, 3, 5, 8, 6, 11]. The essential idea behind most of these approaches is the same, although the details differ. We describe the work of Haddawy and Krieger [6] both as a prototypical knowledge-based model construction approach and as the starting point for our own work.

Haddawy and Krieger present an algorithm for dynamically constructing belief networks from first-order probabilistic knowledge bases. The overall aim is to represent a class of propositional belief networks using a first-order knowledge base. They choose a subset of the FOPL language defined by Halpern in [7] thereby giving their approach a well-defined semantics. Their knowledge base represents the topology and conditional probability tables of a first-order belief network in the FOPL language. For example, such a network for part of the burglar alarm domain might have three nodes, *Alarm(x)*, *Neighbor(n,x)*, and *Call(n,x)*, and two edges, from *Alarm(x)* and from *Neighbor(n,x)* to *Call(n,x)*. This models that for each x and any neighbor n of x, n is likely to call x whenever x's alarm goes off. The CPT for such a network can easily be represented in FOPL. The CPT for the node *Call(n,x)* would be represented as a collection of formulas such as $\forall x, n \Pr(\mathit{Call}(n, x) \mid \mathit{Alarm}(x) \wedge \mathit{Neighbor}(n, x)) = p$, one for each possible combinations of parents' and child's values. We call $\mathit{Call}(n, x)$ the head and $\mathit{Alarm}(x), \mathit{Neighbor}(n, x)$ the body of this rule.

Haddawy and Krieger make the following four assumptions in order to maintain the correspondence between the knowledge base and the belief networks it encodes: (1) Each rule body must also appear as the head of some rule; (2) all variables in the body must appear in the head; (3) the knowledge base does not contain two distinct rules that have ground instances (ground instances contain no variables) with identical heads; (4) the knowledge base is acyclic, i.e., there does not exist a set of ground instances $R_1, \ldots, R_n$ of the rules in the knowledge base such that *head*$(R_1) \in$ *body*(R_2), *head*$(R_2) \in$ *body*(R_3), ..., *head*$(R_n) \in$ *body*(R_1). While the first and last restrictions seem to be inherently necessary for the specification of first-order belief networks, the second and third jeopardize the applicability of the representation language in many domains. We will show in the following sections how to weaken them.

Haddawy and Krieger present an algorithm that takes a probabilistic first-order knowledge base together with a query and a set of evidence as input and outputs the minimal belief network that contains all information relevant to the query. The algorithm performs three major steps: First, backward chaining from the query is done to find all predecessors of the query. In the second step, backward chaining from all the evidence nodes is performed. In the third step, by forward and backward chaining from the query, all nodes that do not influence the query node (there is no active path between them) are pruned away. The result of the third step is the minimal network containing all information relevant to the query. The constraints on the language guarantee that the knowledge base defines a unique distribution over any finite domain. The above described procedure is sound and complete with respect to this distribution.

3 Representation Language

As in the previous works on knowledge-based model construction, our fundamental structure is essentially a first-order belief network. We have a network whose nodes are the atomic formulas $R(x_1, \ldots, x_n)$ in our language. Each node is described using a rule, which defines the influences on the event $R(x_1, \ldots, x_n)$, and the associated probabilities. For example, the topology of the above mentioned network would be represented in our language, which is a subset of the Prolog programming language, as

```
rule(calling,call(N,X),[neighbor(N,X),alarm(X)]).
```

We want to weaken the restrictions that Haddawy and Krieger posed on their language. Although we still require that every formula in the body of some rule matches with the head of some other rule, we provide the user a more expressive language to describe the CPTs (see below). The acyclicity requirement seems to be necessary to keep a well-defined joint probability function in the belief networks. We enforce acyclicity using time parameters. Every predicate has a time argument such that the time argument in the rule head is at least one time step later than the time arguments of the predicates in the rule body. This allows us to model situations where there are cyclic dependencies that manifest themselves over time. For example, in one of our knowledge bases, where we model the behavior of traffic, we formulate a rule

```
rule(hlocation,hloc(X,T),[hloc(X,Q),hvelo(X,Q)]) :- Q is T-1.
```

stating that the horizontal location of car X at time step T depends on the horizontal velocity and location of X one time step earlier.

The assumption that only one rule can match with an atomic formula seems too restrictive in many domains. In the AIDS knowledge base, the random variable aids(X,T) has the intended meaning that person X was infected with the HIV virus at time point T. There are two influences which we model in our knowledge base: whether X was already infected one time step earlier or whether X had contact with another person Y who was infected.

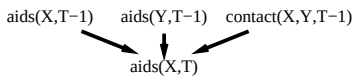

Fig. 1. Schema of the knowledge base for the AIDS domain

We represent these two influences using the construct of a *many-rule* which summarizes different influences on a random variable.

```
many_rule(aids,aids(X,T),[already_infected,sexual_contact]).
```

The third argument in this predicate is a statement containing the names of all rules that influence the head aids(X,T).

In this example, we also want contact with different individuals Y to influence aids(X,T). We therefore allow the head of a rule to contain variables that are not instantiated given a complete instantiation of the body. For example, in the rule above, it is not specified who the partner Y of X was, given only the instantiation of aids(X,T). In principle, this free variable Y can be instantiated with every domain element. (This is the approach taken in our implementation.)

In both of these cases, we allow several rules or rule instantiations to influence a single random variable. In such cases, it is necessary to specify how the influences coming from different rules or rule instantiations should be combined. The combination functions are specified in combination statements. We allow noisy-or, noisy-and, min, and max. Noise-or is typically used to combine independent causes for an event. This is a good model for the AIDS domain, so that our knowledge base for that domain contains the assertion:

```
comb(aids,noisy_or).
```

Of course, we also need to specify CPTs for each individual rule. Haddawy and Krieger describe CPTs by stating a probability for each combination of parents' and child's values. We can do the same by using probability statements of the form

```
prob(calling,call,[neighbor,alarm],0.98).
```

Here, the name calling has to match with the name in a rule statement. The second argument is an instantiation of the head of the rule and the third argument is an instantiation of the body of the rule. The last argument states the probability for that specific

instantiation. Our representation language also provides a far more expressive mechanism for specifying CPTs. It utilizes the first-order power of the language to describe the CPTs functionally. For example, in the case of the above mentioned many_rule in the AIDS knowledge base, if it is known that X was already infected in the previous time step, we do not need to take the values of the other parents into consideration. Our language allows such a functional representation by stating the probability statement as a Prolog routine:

```
prob(aids,Aids_X_T,[Aids_X_Q,Aids_Y_Q,Contact_X_Y_Q]) :-
   % <Here comes the computation>.
```

In particular, we could use such a statement to represent CPTs as decision trees, as in Fig. 2. This allows us to model situations where, depending on the values which some of the random variables in the rule body take on, the head becomes independent of parts of the rule body. If we know, for example, that a person X was already infected with the HIV virus at time T-1, he will continue to be infected at time T. The HIV-infection status of other persons and their possible contacts with X are not relevant to the value of this variable. Similarly, the HIV-infection status of another person Y is not relevant if there was no contact between X and Y. The decision-tree representation allows us to model such situations concisely, and can also be used to prune the networks resulting from the knowledge-based model construction process (see below).

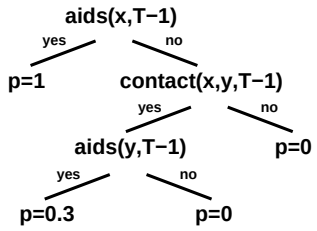

Fig. 2. CPT as a decision tree

We can also use the ability to specify CPTs functionally in other situations. In the traffic domain, for example, we model the random variables describing the location and the speed of a car as normally distributed random variables over the real numbers. The formalism of probability statements is able to capture this:

```
prob(hlocation,Head,Body,P) :- P is gaussian(U,V),
   % <computation for U and V>.
```

When constructing the propositional belief network, our implementation could express these variables as continuous-valued nodes. The inference on the resulting networks could be executed by stochastic simulation algorithms, since these are capable of doing inference in networks containing both discrete and continuous variables. In certain cases, the approach of [9] could also be used. Alternatively (although this is not yet implemented in our system), we could automatically discretize these nodes according to the user's specification.

4 Constructing the Models

Our network construction algorithm extends that of Haddawy and Krieger [6] to deal with procedural descriptions of CPTs and with modular specification of multiple influences on a node. In particular, our implementation allows more than one rule with the same head, and can also handle rules that have free variables in the body given the instantiated head. These variables are instantiated at run-time with all domain elements.

Roughly speaking, the algorithm works as follows. As in the algorithm of [6], it backward chains from the query and from the evidence. Each ground fact generated in this process forms a node A in the network. The parents of A are generated from the bodies of all the rules whose head match this ground fact. When the body of such a rule has free variables not instantiated in the head, these are instantiated with all possible domain elements. For each relevant rule, and for each instantiation of the variables, the algorithm computes the probabilities defined by each rule separately. Since these might be described procedurally (e.g., using a decision tree), this might require some computation. The influences from the different rules or rule instantiations are then combined using the combination function specified by the user. Our current implementation allows noisy-or, noisy-and, min, and max but this list is easily extendable. This completes the construction of the neighborhood of the node A (A, its parents, and the associated CPT). The backward chaining process then continues on the set of ground facts corresponding to A's parents. Finally, after the network is constructed, the d-separation criterion is used to prune the irrelevant parts of the network.

In Haddawy and Krieger's algorithm, the restrictions placed on the language guaranteed that the prior probability distribution is uniquely determined. Because of the use of combination rules and because of the use of time parameters to enforce acyclicity, this unique prior probability distribution property still holds for our extensions. Therefore the extended algorithm is sound and complete with respect to the distributions defined by our knowledge bases. (This is an easy extension of Haddawy and Krieger's theorem.)

In our implementation, we used two application domains as our main examples. The first domain models the propagation of infectious diseases; we used AIDS as our example. The second domain is a simple partial model of traffic behavior on highways; it emphasizes the random variable that a lane change is safe. For details of this second knowledge base, we refer to [4].

The first-order belief network structure for the AIDS knowledge base was given above in Fig. 1. As described in the previous section, our variables in this domain are `aids(X,T)` and `contact(X,Y,T)`. The probabilities for the `aids` rule are as specified in Fig. 2. The probability of `contact(X,Y,T)` is taken to be 0.1.

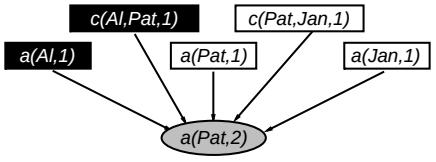

Fig. 3. Simple query for the AIDS knowledge base

Fig. 3 shows a flexible dynamic belief network produced by our algorithm from this knowledge base for a particular situation of interest. We mark evidence nodes by rectangles: black rectangles denote nodes that are set to true, and white rectangles nodes that are set to false. The query is denoted by a gray oval. In this example, we assume that the domain consists of three people: Al, Pat, and Jan. At time 1, Al is infected but Pat and Jan are not. Moreover we know that at this time point Al and Pat had contact with each other, but besides that there was no other contact. If we ask for the probability that Pat is infected at time 2, we get Fig. 3.[1] If we run this network on the HUGIN system, we obtain that the probability of `aids(Pat,2)` is 0.3.

The network changes considerably if we get the additional evidence that at time 3, Pat was diagnosed with the AIDS virus. We now have a significantly larger network for the same query, because other aspects of the problem become relevant. For example, Pat might have been infected via contact with Jan at time 2, so that Jan's HIV status (and its causes) becomes relevant. The network generated by our system is shown in Fig. 4. The resulting probability is 0.935.

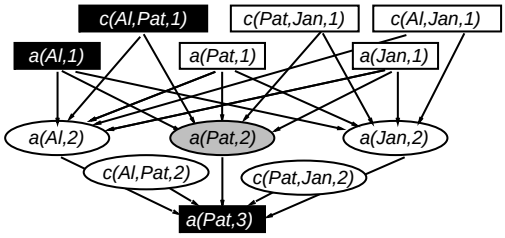

Fig. 4. Network for the same query with one additional piece of evidence

We mentioned in Sect. 3 that our language allows us to represent CPTs as decision trees. Although this is not yet implemented, the decision-tree representation can be used to prune the constructed belief network considerably. For example, we know that if an individual is infected at time `T-1`, he is also infected at time `T` independently of any other event. This would allow us to prune all the edges leading to the node `aids(Al,2)` except for the one from `aids(Al,1)`. Similarly, the fact that Jan had no contact with anyone at time 1 would allow us to prune some of the edges leading into `aids(Jan,2)`. Fig. 5 shows the network that would result from such a process. Since our system provides the infrastructure for the decision-tree representation, the implementation of the pruning routine should be straightforward.

[1] Currently, the knowledge base does not encode the fact the the `contact` relation is irreflexive and symmetric. Therefore, the network constructed by the algorithm would also contain the node `contact(Pat,Pat)`, which we have eliminated for clarity. A similar phenomenon occurs in Fig.s 4 and 5, where we have also eliminated the duplicate nodes resulting from symmetry (e.g., `contact(Pat,Al)`).

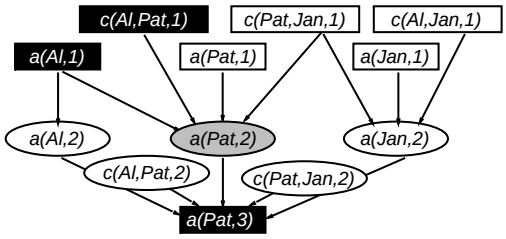

Fig. 5. The network obtained after pruning using the decision-tree representation

5 Conclusions and Further Work

In this paper we introduced a Prolog-based language for representing first-order dynamic belief networks. As our examples illustrate, this language is powerful enough to capture complex dynamic domains. In particular, it allows us to modularize the influences on an event and to combine them in different ways. It also allows us to use the first-order power of the language in representing the probabilities associated with an event. For example, we can easily provide a functional representation of the probabilities in the CPTs, which, in particular, allows us to easily include continuous random variables. We also described the implementation of an algorithm that takes knowledge bases in this format as input together with a query, evidence, and fixed domain and generates propositional dynamic belief networks. In contrast to standard dynamic belief networks, those resulting from our algorithm have a flexible structure that is adapted to the specific situation.

This ability to create flexible dynamic belief networks should have many other applications. In dynamic situations, we would often like to extend our current model further into the future, while "forgetting" things about the past. In the context of standard dynamic belief networks, new time slices are simply added to the existing network, while earlier time slices are marginalized into the newer ones. This process is called *roll-up*. It would be interesting to define a similar procedure for flexible dynamic belief networks, which would use our approach to generate new time slices with a structure that depends on the current situation. Such a flexible roll-up procedure would allow us to use our system in dynamic environments over time and not only for a given single situation.

We may also wish to model some variables more often than others. For example, it is probably not useful to update our weather variable five times a second, whereas such granularity is required when modeling vehicle behavior. The power of our representation language makes this very easy to accomplish. We can simply modify the rule corresponding to the weather variable so that the variable only exists in certain time slices (e.g., once every ten minutes), and update the rules for those variables that depend on the weather to refer to the latest copy of that variable. We could use the expressive power of the language to specify the required CPTs functionally, e.g., based on the time elapsed since the variable was last updated. This would allow us to treat frequency with which a particular variable is modeled as a parameter, and to change it without modifying the knowledge base. This may allow us to dynamically change this frequency, as required by the situation. For example, it is more important to provide accurate predictions for

a nearby car that is changing lanes than for a distant car that is going at a steady rate. This idea can be combined with a flexible roll-up algorithm that creates new time slices as needed, thereby producing a very powerful paradigm for producing very compact flexible dynamic belief networks, where only the relevant variables are represented.

Acknowledgements We would like to thank Stuart Russell for many useful comments, and Timothy Huang and Keiji Kanazawa for helping us with information about the traffic domain. This research was supported by the German Fulbright Commission, the Studienstiftung des deutschen Volkes, a University of California President's Postdoctoral Fellowship, and an NSF Experimental Science Postdoctoral Fellowship.

References

1. S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen. HUGIN—a shell for building Bayesian belief universes for expert systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, volume 2, pages 1080–1085, 1989.
2. Fahiem Bacchus. Using first order probability logic for the construction of bayesian networks. *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, 1993.
3. John S. Breese. Construction of belief and decision networks. *Computational Intelligence*, 1992.
4. S. Glesner. Constructing flexible dynamic belief networks from first-order probabilistic knowledge bases. Master's thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1994.
5. R. P. Goldman and E. Charniak. Dynamic construction of belief networks. *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 90–97, July 1990.
6. P. Haddawy and R. A. Krieger. Principled construction of minimal bayesian networks from probability logic knowledge bases. Submitted to Journal of AI Research.
7. J. Y. Halpern. An analysis of first order logics of probability. *Artificial Intelligence*, 46:311–350, May 1991.
8. M. C. Horsch and D. Poole. A dynamic approach to probabilistic inference using bayesian networks. *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 155–161, July 1990.
9. S. L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108, December 1992.
10. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
11. M. P. Wellman, J. S. Breese, and R. P. Goldman. From knowledge bases to decision models. *The Knowledge Engineering Review*, 7(1):35–53, November 1992.