# A Dynamic Approach to Probabilistic Inference using Bayesian Networks

**Michael C. Horsch** and **David Poole**[*]
Department of Computer Science
University of British Columbia
Vancouver, British Columbia
Canada
email: horsch@cs.ubc.ca, poole@cs.ubc.ca

## Abstract

In this paper we present a framework for dynamically constructing Bayesian networks. We introduce the notion of a background knowledge base of *schemata*, which is a collection of parameterized conditional probability statements. These schemata explicitly separate the general knowledge of properties an individual may have from the specific knowledge of particular individuals that may have these properties. Knowledge of individuals can be combined with this background knowledge to create Bayesian networks, which can then be used in any propagation scheme.

We discuss the theory and assumptions necessary for the implementation of dynamic Bayesian networks, and indicate where our approach may be useful.

## 1 A dynamic approach to using Bayesian networks

Bayesian networks are often used in expert systems [1], and decision analysis [8, 3], where a network is engineered to perform a highly specialized analysis task.

In these applications, the network often implicitly combines general knowledge with specific knowledge. For example, a Bayesian network with an arc as in Figure 1 refers to a specific individual (a house or a tree or dinner or whatever), exhibiting a somewhat generalized property (fire causes smoke).

The dynamic approach presented in this paper is motivated by the observation that a knowledge engineer who is modelling a domain has expertise in that domain (or is able to consult with an expert), but may not be able to anticipate the individuals which are to be included in the model.

This paper presents a framework in which probabilistic information is written in a knowledge base of parameterized *schemata*, explicitly separating the general knowledge of the properties an individual may have, from the specific knowledge of the particular individual which may have these properties. When the system is supplied with knowledge of the individuals that are being modelled, an automatic process combines the parameterized information in the knowledge base with the individuals, creating a Bayesian network.

By separating properties from individuals the knowledge engineer can write a knowledge base which is independent of the individuals; the system user can tell the system which individuals to consider, because she can make this observation at run time. Thus the system user doesn't have to be an expert in the domain to create an appropriate network. In an advanced application, it is conceivable that an automated process would be able to make the observations as part of its operation, constructing and adjusting a probability model as it interacts with its domain.

## 2 Bayesian Networks

A Bayesian network is a directed acyclic graph which represents in graphical form the joint probability distribution, and the statistical independence assumptions, for a set of random variables.

A node in the graph represents a variable, and an

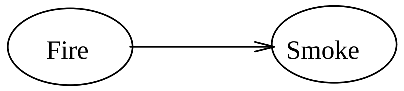

Figure 1: A simple arc implicitly representing an individual.

[*]This research is supported in part by NSERC grant #OGPOO44121.

arc indicates that the node at the head of the arc is directly dependent on, or *conditioned by* the node at the tail. The collection of arcs directed to a variable give the independence assumptions for the dependent variable. Associated with this collection is a prior probability distribution, or *contingency table*, which quantifies the effects of observing events for the conditioning variables on the probability of the dependent variable.

The graphical representation is used in various ways to calculate posterior joint probabilities. Pearl [5] uses the arcs to propagate causal and diagnostic support values throughout a singly–connected network. Lauritzen and Spiegelhalter [4] perform evidence absorption and propagation by constructing a triangulated graph based on the Bayesian network that models the domain knowledge. Shachter [8] uses the arcs to perform node reduction on the network.[1] Poole and Neufeld [6] implement an axiomatization of probability theory in Prolog which uses the arcs of the network to calculate probabilities using "reasoning by cases" with the conditioning variables. Our current implementation uses the work of Poole and Neufeld, but is not dependent on it.

## 3 Representation Issues

Creating networks automatically raises several issues which we discuss in this section. First, we need to represent our background knowledge in a way which preserves a coherent joint distribution defined by Bayesian networks, and which also facilitates its use in arbitrary situations. Furthermore, there are situations in which using the background knowledge may lead to ambiguity, and our approach must also deal with this.

Before we get into our discussion, a short section dealing with syntactic conventions will help make things clearer.

### 3.1 Some syntax

In this section we clarify the distinction between our background knowledge and Bayesian networks.

In our Bayesian networks, random variables are propositions written like ground Prolog terms. For example, the random variable *flies(e127)* could represent the proposition that individual *e127* flies.

A *schema* is part of the background knowledge base, and describes qualitatively the direct dependencies of a random variable. In this paper, a schema is stated declaratively in sans serif font.

[1]Shachter's influence diagrams contain decision and test nodes, as well as other devices not being considered in this paper. We are only looking at probabilistic nodes at present.

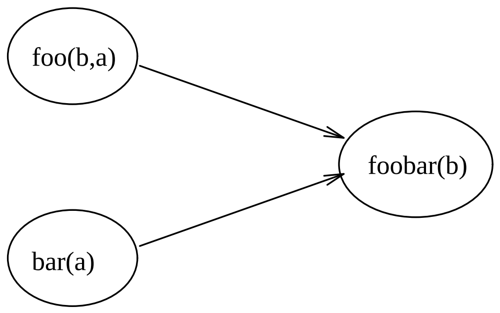

Figure 2: A simple instantiation of a schema.

A schema can be defined constructively: a parameterized atom is written as a Prolog term with parameters capitalized. A schema is represented by the following:

$$\mathsf{b} \quad \longleftarrow \quad \mathsf{a}_1, \ldots, \mathsf{a}_\mathsf{n}$$

where the arrow $\longleftarrow$ indicates that the parameterized atom, $b$, on the left side is directly dependent on the parameterized atoms, $a_i$, on the right.

An instantiation of a parameter is the substitution of a parameter by a constant representing an individual. We indicate this in our examples by listing the individuals in a set, as in $\{\mathsf{e127}\}$. Instantiating a parameterized atom creates a proposition that can be used in a Bayesian network.

A schema is instantiated when all parameters have been instantiated, and an instantiated schema becomes a part of the Bayesian network, indicating a directed arc from the instantiation of each $a_i$ and the instantiation of $b$.

For example, the schema:

$$\mathsf{foobar(X)} \quad \longleftarrow \quad \mathsf{foo(X,a), bar(a)}$$

with $\{\mathsf{b}\}$ instantiating the parameter $\mathsf{X}$, creates the network[2] shown in Figure 2.

The schemata in the background knowledge base are isolated pieces of information, whereas a Bayesian network is a single entity. An interpretation is that a Bayesian network defines a joint distribution, whereas the schemata in a knowledge base define conditional probability factors of which instances are used to compute a joint probability distribution.

[2]In our examples, Bayesian networks are pictured as encircled node labels connected by directed arrows, as in Figure 2.

A schema is incomplete without the contingency table which quantifies the connection between the dependent variable and its conditioning parents.[3] These tables have the following form, and follow the same syntactic assumptions as the symbolic schemata:

$$\begin{aligned}
\mathrm{P}(foobar(X)|foo(X,a),bar(a)) &= 0.95\\
\mathrm{P}(foobar(X)|foo(X,a),\neg bar(a)) &= 0.666\\
\mathrm{P}(foobar(X)|\neg foo(X,a),bar(a)) &= 0.25\\
\mathrm{P}(foobar(X)|\neg foo(X,a),\neg bar(a)) &= 0.15
\end{aligned}$$

By writing a parameterized schema, we are licensing a construction process to instantiate the parameters in the schema for any individual. This requires that the knowledge we provide as schemata must be written generally.

The parameters in the schema and the contingent probability information should not be interpreted with any kind of implicit quantification. Rather, they are intended to apply to a "typical" individual. Furthermore, when this parameterized information is used, the parameters should be replaced with constants which do not convey any intrinsic information. By designing schemata carefully, we can avoid the "Ace of Spades" problem [7].

## 4 Dynamic Instantiation

A general schema with parameters can be instantiated for more than one individual. There are three cases:

### 4.1 Unique schemata

These are schema which have no parameters, or which have the same parameters occurring on both sides of the arc. For example:

$$\mathsf{c(X)} \longleftarrow \mathsf{a(X), b}$$

For the constants {x,y} instantiating the parameter X, the network building process puts two arcs in our network, as in Figure 3.

### 4.2 Right–multiple schemata

In the second case, there are parameters on the right side of the schema which do not occur on the left side, as in c(Y) ⟵ a,b. (Note that, in addition, there may be parameters which occur on both sides of the arc). Instantiating the parameter Y in the above schema with values {x, y, z} creates the network shown in Figure 4.

[3] In our examples, we do not provide the table explicitly, we assume its existence implicitly.

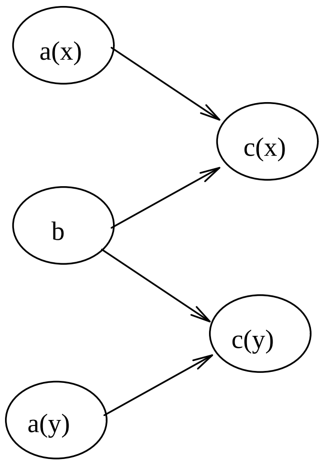

Figure 3: The network created by instantiating the parameterized unique schemata from Section 4.1.

**Example 1:** Pearl [5, Chapter 2] presents an extended example, about burglar alarms and testimonies from several people, as motivation for the use of Bayesian networks in an automated reasoning system. The following schemata can specify this problem:

$$\begin{aligned}
\mathsf{alarm\_sound} &\longleftarrow \mathsf{burglary, earthquake}\\
\mathsf{news\_report} &\longleftarrow \mathsf{earthquake}\\
\mathsf{testimony(X)} &\longleftarrow \mathsf{alarm\_sound}\\
\mathsf{call(Y)} &\longleftarrow \mathsf{alarm\_sound}
\end{aligned}$$

Note the use of right–multiple schemata. This allows us to specify multiple cases for testimonies from different people. In particular, we could tell the system that we are investigating the hypothesis that a burglary occurred based on testimonies from Dr Watson and Mrs Gibbons. □

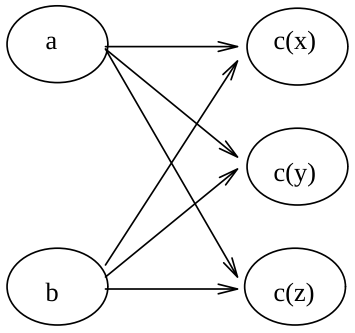

Figure 4: The network created by instantiating the parameterized right–multiple schema from Section 4.2.

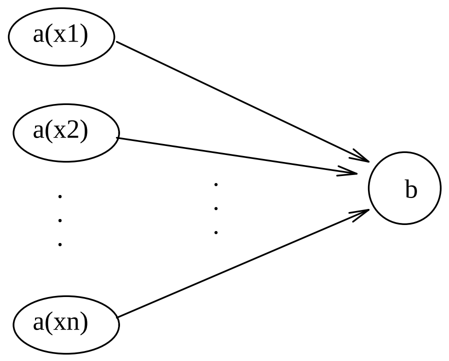

Figure 5: The network created by instantiating the parameterized left–multiple schema from Section 4.3.

### 4.3 Left–multiple schemata

The remaining case is characterized by parameters occurring on the left side of the schema which do not occur on the right side. For example:

$$\mathsf{b} \quad \longleftarrow \quad \mathsf{a(X)}$$

Instantiating the parameter $\mathsf{X}$ with a number of individuals $\{\mathsf{x1}, \ldots, \mathsf{xn}\}$ creates a network structure as shown in Figure 5. The conditional probability for $\mathsf{b}$ must be stated in terms of multiple instances of the node $\mathsf{a(X)}$. To use the multiple instantiation of a left–multiple schema, we need to be able to provide contingencies for any arbitrary number of individuals. This is problematic because it is not reasonable to provide this kind of information (in terms of contingency tables, for example), as background knowledge: we would require a table for every possible set of individuals. To be useful, the contingency tables for left–multiple schemata must allow for any number of instantiations.

It would be representationally restrictive to disallow the use of left–multiple schemata, because sometimes we do want to allow an unspecified number of variables to condition a dependent variable. For example, we may want to include in our knowledge base the idea that any number of people may call the fire department to report a fire.

We provide two mechanisms, called existential and universal combination, which specify how an unknown number of possible conditioning variables affects another variable. These are based on the *Canonical Models of Multicausal Interactions* suggested by Pearl [5].

**Existential Combination**

Consider the following example: A person who smells smoke may set off a fire alarm, and the sound of the fire alarm may cause others to leave the building.

This situation shows the usefulness of an existential combination over individuals in the problem. We want to combine the effects of each individual into a single variable which relates the likelihood that at least one of the individuals satisfied the condition. The following schema represents this notion syntactically:

$$\mathsf{b} \quad \longleftarrow \quad \exists \mathsf{X} \in \mathsf{type} \cdot \mathsf{a(X)}.$$

The existential schema serves as a short–hand notation for a dynamic Or–rule structure. For example, when it is given that $\{\mathsf{x1}, \mathsf{x2}, \ldots, \mathsf{xn}\}$ are members of the set $\mathsf{type}$, the schema above expands into a network structure as in Figure 6, where the contingency table for $\exists\ \mathsf{X} \in \mathsf{type} \cdot \mathsf{a(X)}$ is unity if any of $\mathsf{a(X)}$ is known to be true, and zero if they are all false. The Or–rule is indicated in the diagram by the circular arc on the inputs to the existential node.

This schema requires a contingency table such as:

$$\mathrm{P}(b|\exists X \in \mathit{type} \cdot a(X)) = 0.7665$$
$$\mathrm{P}(b|\neg\exists X \in \mathit{type} \cdot a(X)) = 0.0332$$

which describes the effect on $\mathsf{b}$ for the case where at least one known individual satisfies $\mathsf{a(X)}$, and the case where no known individual does (these numbers are arbitrary).

**Example 2:** The example about fire alarms, given at the beginning of this section, can be represented by the following background knowledge base:

$$\begin{array}{rcl}
\mathsf{smells\_smoke(X)} & \longleftarrow & \mathsf{fire} \\
\mathsf{sets\_off\_alarm(X)} & \longleftarrow & \mathsf{smells\_smoke(X)} \\
\mathsf{alarm\_sounds} & \longleftarrow & \exists \mathsf{Y} \in \mathsf{person} \cdot \mathsf{sets\_off\_alarm(Y)} \\
\mathsf{leaves\_building(Z)} & \longleftarrow & \mathsf{alarm\_sounds}
\end{array}$$

Suppose we are given that *john* and *mary* are the only known members of the set *person.* This information creates the network shown in Figure 7. □

There are several points which should be made:

1. The variable $\exists\ \mathsf{X} \in \mathsf{type} \cdot \mathsf{a(X)}$ is propositional, and when it is true, it should be interpreted as the fact that some known individual satisfies the proposition $\mathsf{a(X)}$. When this existential combination is false, the interpretation should

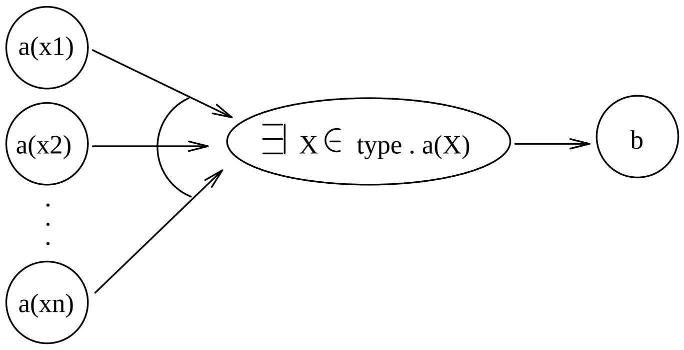

Figure 6: The instantiation of the existential combination node from Section 4.3.

be that no known individual satisfies the proposition $\mathsf{a(X)}$, although there may be an individual satisfying the proposition who is unknown to the system.

2. The proposition $\exists\ \mathsf{X} \in \mathsf{type} \cdot \mathsf{a(X)}$ can be considered a schema on it's own, acting like a variable which is disjunctively influenced by $\mathsf{a(c)}$ for every constant $\mathsf{c}$ in the set $\mathsf{type}$. The intermediate Bayesian arcs are not written as part of the knowledge base.

3. The set $\mathsf{type}$, from which the individuals for this schema are taken, serves two purposes: to constrain the applicability of the individuals to the schema, and to facilitate the dynamic specification of individuals. In this way, all and only those individuals who are known to satisfy the type requirement will be included in this combination.

**Universal Combination**

Universal combination can be used in situations for which all individuals of a certain type must satisfy a condition to affect a consequent. For example, a board of directors meeting can begin only when all the members who are going to come are present.

We represent this idea with a schema with the following syntax:

$$\mathsf{b} \quad \longleftarrow \quad \forall \mathsf{X} \in \mathsf{type} \cdot \mathsf{a(X)}$$

The discussion concerning the existential combination applies here, with the only exception being the interpretation of the schema. The variable $\forall\ \mathsf{X} \in \mathsf{type} \cdot \mathsf{a(X)}$ is true if $\mathsf{a(X)}$ is true for every member of the set $\mathsf{type}$. We treat the combination as a dynamic And–rule over all members of the set $\mathsf{type}$. The And–rule contingency table is unity if all the members satisfy the condition $\mathsf{a(X)}$, and zero otherwise.

**Example 3:** A board meeting for a large corporation requires the presence of all board members, and the meeting may result in some actions, say, buying out a smaller company. A board member may attend the meeting, depending on her reliability and state of health.

$$\begin{array}{rcl} \mathsf{meeting} & \longleftarrow & \forall \mathsf{X} \in \mathsf{board\_members} \cdot \mathsf{present(X)} \\ \mathsf{buy\_out} & \longleftarrow & \mathsf{meeting} \\ \mathsf{present(X)} & \longleftarrow & \mathsf{sick(X)}, \mathsf{reliable(X)} \end{array}$$

We note that at the time the schemata are written, it is not important to know how many board members there may be. However, we must know exactly who is a member on the board before the network can be created. This information is supplied at run time, and the construction process can construct the appropriate network. Evidence concerning the reliability and health of any board member can then be submitted to the network, providing appropriate conditioning for queries. □

## 5 Creating Bayesian networks

To create a Bayesian network, the schemata in our knowledge base must be combined with the individuals the process knows about. The process by which this is accomplished in our implementation has been kept simple, and syntactic.

Every schema in the knowledge base is instantiated by substituting ground atoms representing the individuals known to be part of the model. Each instantiation of a schema is an arc in the Bayesian network. The contingency table associated with each schema is also instantiated, providing the prior probability information we require.

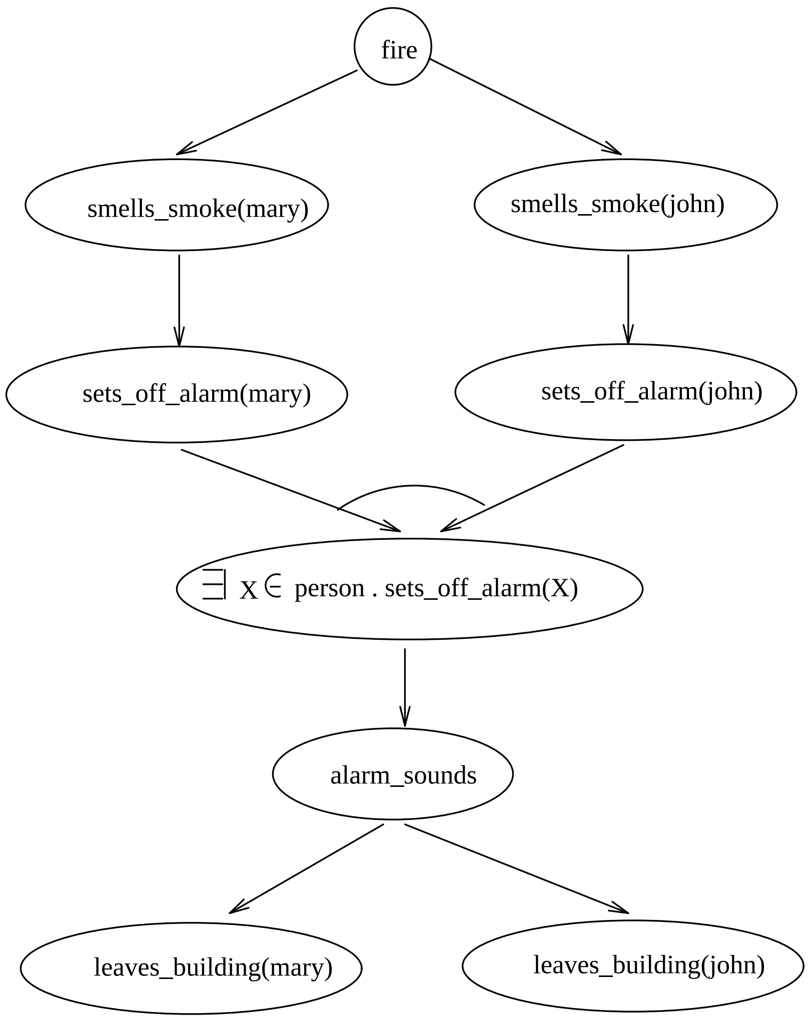

Figure 7: The Bayesian network created for Example arabiceg1.

This procedure creates the collection of Bayesian arcs we use as the Bayesian network for our model.

## 6 Current and Future Work

These proceedings report simultaneous and independent research on building Bayesian networks dynamically. Goldman and Charniak [2] use a sophisticated rule base to construct Bayesian networks using more 'semantic' knowledge. As well, their approach handles cases where arcs are added to a network which has already absorbed evidence.

We have outlined only two ways in which the effects of an unknown number of individuals can be combined. These could be extended to include other types of combination.

This paper describes how knowledge of individuals can be used to create a Bayesian network. There are times when we want to use observations about the individuals to add arcs to the network. For instance, observing the topology of an electronic circuit should help structure the Bayesian network which models the uncertainty in diagnosing faults. These observations have a unusual property: they can be used to structure a network if they are observed to be true, and yield no exploitable structural information if they are observed false. We are looking at how we might implement this idea.

The process, described in this paper, which creates the Bayesian network from the knowledge base is very simple, and many improvements could be made. One way to limit the size of our networks is to delay the creation of the network until all the evidence available has been observed. This would allow the exploitation of conditional independence at the level of the structure of the network, making the network smaller and easier to evaluate. This idea is not exploited by many systems (for example, Lauritzen and Spiegelhalter [4] mention the idea, but argue against it).

The ideas in this paper have been implemented and tested using toy problems. Currently, larger applications are being built in more sophisticated domains, including circuit diagnosis and the interpretation of sketch maps.

## 7 Conclusions

In this paper we have presented a dynamic approach to the use of Bayesian networks which separates background and specific knowledge. Bayesian networks are created by combining parameterized schemata with the knowledge of individuals in the model.

Our approach to the use of Bayesian networks has been from the viewpoint of providing an automated tool for probabilistic reasoning, and is useful for modelling problems for domains in which many details of the problem cannot be exhaustively anticipated by the designer of the probabilistic model. Such domains include expert systems, diagnosis, story understanding, natural language processing and others.

An implementation of this approach has been completed in Prolog.

## References

[1] S Andreassen, M. Woldbye, B. Falck, and S.K. Andersen. MUNIN—A Causal Probabilistic Network for Interpretation of Electromyographic Findings. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 366–372, 1987.

[2] Robert P. Goldman and Eugene Charniak. Dynamic construction of belief networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 90–97, 1990.

[3] R.A. Howard and J.E. Matheson, editors. *Readings on the Principles and Applications of Decision Analysis.* Strategic Decisions Group, CA, 1984.

[4] S. L. Lauritzen and D. J. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems. *J. R. Statist Soc B*, 50(2):157–224, 1988.

[5] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Reasoning.* Morgan Kaufmann Publishers, Los Altos, 1988.

[6] David Poole and Eric Neufeld. Sound probabilistic inference in prolog: An executible specification of Bayesian networks. 1989.

[7] J. K. Schubert. Cheeseman: A travesty of truth. *Computational Intelligence*, 4(1):118–121, 1988.

[8] Ross D. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 36(4):589–604, 1988.