Annals of Mathematics and Artificial Intelligence **32:** 179–220, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

Complex probabilistic modeling with recursive relational Bayesian networks

Manfred Jaeger

Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany
E-mail: jaeger@mpi-sb.mpg.de

A number of representation systems have been proposed that extend the purely propositional Bayesian network paradigm with representation tools for some types of first-order probabilistic dependencies. Examples of such systems are dynamic Bayesian networks and systems for knowledge based model construction. We can identify the representation of probabilistic relational models as a common well-defined semantic core of such systems. Recursive relational Bayesian networks (RRBNs) are a framework for the representation of probabilistic relational models. A main design goal for RRBNs is to achieve greatest possible expressiveness with as few elementary syntactic constructs as possible. The advantage of such an approach is that a system based on a small number of elementary constructs will be much more amenable to a thorough mathematical investigation of its semantic and algorithmic properties than a system based on a larger number of high-level constructs. In this paper we show that with RRBNs we have achieved our goal, by showing, first, how to solve within that framework a number of non-trivial representation problems. In the second part of the paper we show how to construct from a RRBN and a specific query, a standard Bayesian network in which the answer to the query can be computed with standard inference algorithms. Here the simplicity of the underlying representation framework greatly facilitates the development of simple algorithms and correctness proofs. As a result we obtain a construction algorithm that even for RRBNs that represent models for complex first-order and statistical dependencies generates standard Bayesian networks of size polynomial in the size of the domain given in a specific application instance.

Keywords: first-order probabilistic representations, knowledge based model construction, Bayesian networks, temporal and relational models

1. Introduction

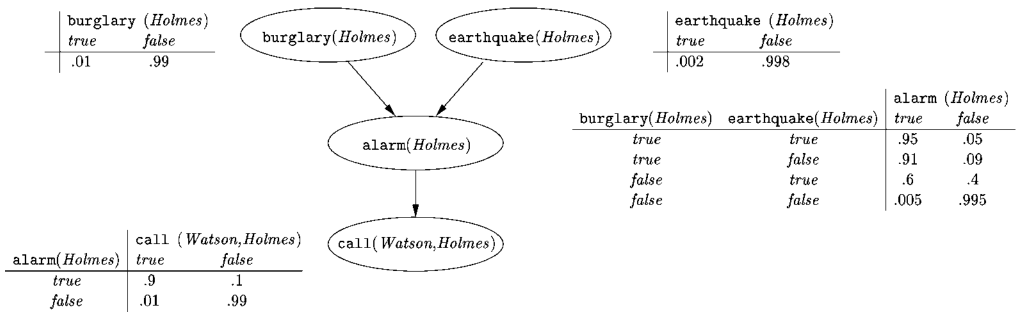
1.1. *From propositional to relational probabilistic models*

Uncertain information can often be modeled as a joint probability distribution of a set $\mathtt{X}_1, \dots, \mathtt{X}_n$ of random variables. An important subclass of such joint distributions is given by the case where each random variable $\mathtt{X}_i$ can only take on finitely many different values. A random variable with k possible values can also be encoded with $\lceil \log(k) \rceil$ *Boolean random variables*, i.e., random variables whose possible values are only *true* and *false*. A joint distribution of random variables with finite ranges, thus, can also be encoded as a joint distribution of Boolean random variables. Such a distribution we call a *probabilistic propositional model* (ppm).

Bayesian networks are the most prominent representation framework for ppms. Figure 1(a) shows a classic example of a Bayesian network representing a joint distribution of the Boolean random variables `burglary`(*Holmes*) (standing for the event that Holmes' home was broken into), `earthquake`(*Holmes*) (there was an earthquake in the area where Holmes lives), `alarm`(*Holmes*) (the alarm at Holmes' house went off), and `call`(*Watson*, *Holmes*) (Holmes received a phone call from his neighbor Watson). The network structure, in conjunction with the conditional probability tables attached to the nodes, specify a joint probability distribution of the network's variables [10,23].

The network in figure 1(a) is adequate for Holmes to evaluate the probability of a burglary at his house when Holmes has exactly one neighbor, Watson. If Holmes has two neighbors, Watson and Gibbon, then he needs to use the ppm represented by the network shown in figure 1(b).

Figures 1(a) and (b) show two distinct Bayesian networks representing two distinct ppms. The two models are very similar, however: the random variables in network (a) are a subset of the random variables in network (b), and the distribution defined by the network (a) is just the marginal on the random variables the two networks have in common of the distribution defined by the network (b). In fact, the similarities between the two models are such that one can hardly speak about two distinct models, but only about



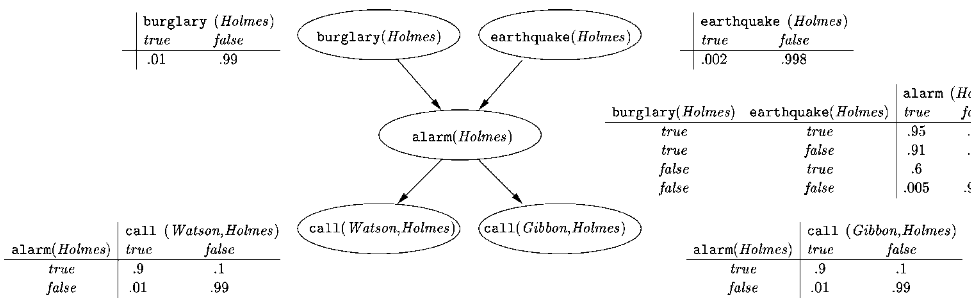
burglary (*Holmes*)	
true	*false*
.01	.99

earthquake (*Holmes*)	
true	*false*
.002	.998

		alarm (*Holmes*)	
burglary(*Holmes*)	earthquake(*Holmes*)	*true*	*false*
true	*true*	.95	.05
true	*false*	.91	.09
false	*true*	.6	.4
false	*false*	.005	.995

	call (*Watson*,*Holmes*)	
alarm(*Holmes*)	*true*	*false*
true	.9	.1
false	.01	.99

(a)



burglary (*Holmes*)	
true	*false*
.01	.99

earthquake (*Holmes*)	
true	*false*
.002	.998

		alarm (*Holmes*)	
burglary(*Holmes*)	earthquake(*Holmes*)	*true*	*false*
true	*true*	.95	.05
true	*false*	.91	.09
false	*true*	.6	.4
false	*false*	.005	.995

	call (*Watson*,*Holmes*)	
alarm(*Holmes*)	*true*	*false*
true	.9	.1
false	.01	.99

	call (*Gibbon*,*Holmes*)	
alarm(*Holmes*)	*true*	*false*
true	.9	.1
false	.01	.99

(b)

Figure 1. Different instances of the same basic model.

two different instances of one basic underlying model. This underlying model, however, no longer can simply be a ppm. Instead, the underlying model must essentially be a blueprint for the construction of individual ppms. To obtain a mathematical formalization of such a blueprint, we first observe that all the random variables in figure 1 are designated by what in predicate logic would be called a ground atomic formula, or, shorter, a *ground atom*: a predicate (or relation) symbol (burglary, ..., call) applied to some constants (*Holmes*, *Watson*). We also see that the random variables appearing in the two ppms of figure 1 are constructed by applying the same relations to the two distinct sets of constants $\{Holmes, Watson\}$ and $\{Holmes, Watson, Gibbon\}$, respectively. The blueprint that underlies the two ppms of figure 1, thus, can be understood as a generic probabilistic model for the relations burglary, ..., call, which can be applied to any (finite) domain of individuals. A (provisional) mathematical definition of such a generic probabilistic model (or blueprint for the construction of ppms) is as follows.

Provisional definition. A *probabilistic relational model* (prm) consists of a set R of relation symbols, and a mapping that assigns to every finite set $D = \{d_1, \dots, d_n\}$ of constants a joint probability distribution of the set

$$\left\{\mathtt{r}(d_{i_1}, \dots, d_{i_l}) \mid \mathtt{r} \in R,\ d_{i_1}, \dots, d_{i_l} \in D\ (l = \text{arity of } \mathtt{r})\right\}$$

of Boolean random variables.

In other words, a prm maps finite domains D to a ppm whose random variables are the ground atoms $\mathtt{r}(d_{i_1}, , \dots, d_{i_l})$ $(d_{i_j} \in D)$. Each such ppm defined by some domain D we call an *instance* of the prm. Comparing these definitions with the networks shown in figure 1, we find that, strictly speaking, the networks do not represent instances of a prm, because they do not contain nodes for all ground atoms that can be formed with the given relation and constant symbols. A full instance of a prm given by $D = \{Holmes, Watson\}$ is represented by the network shown in figure 2 (the probability tables for nodes other than call(*Holmes*, *Holmes*) and call(*Watson*, *Watson*) being as in figure 1). Similarly, figure 1(b) only shows a part of the instance of the underlying prm obtained for $D = \{Holmes, Watson, Gibbon\}$.

The concept of a prm is much more general than perhaps suggested so far by our introductory example. Another class of prms is represented by *dynamic Bayesian net-*

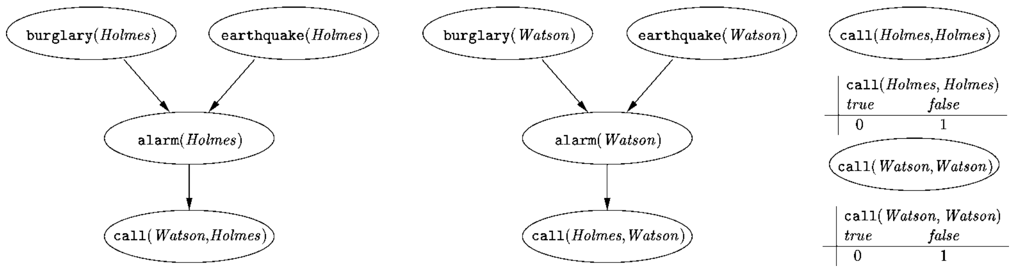
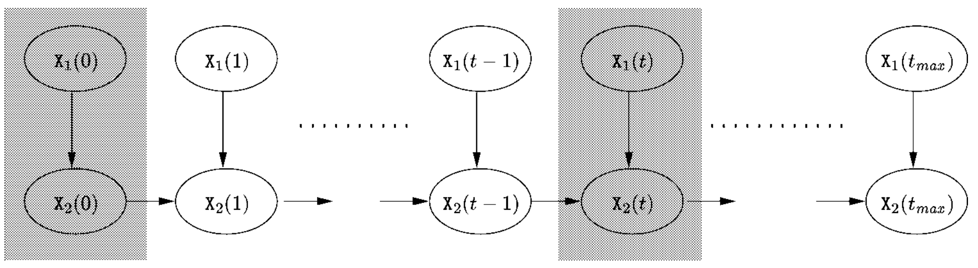


Figure 2. A full instance of a prm for $D = \{Holmes, Watson\}$.



$X_1(0)$ *true*	$X_1(0)$ *false*
.3	.7

$X_1(0)$	$X_2(0)$ *true*	$X_2(0)$ *false*
true	.2	.8
false	.4	.6

$X_1(t)$ *true*	$X_1(t)$ *false*
.3	.7

$X_1(t)$	$X_2(t-1)$	$X_2(t)$ *true*	$X_2(t)$ *false*
true	*true*	.7	.3
true	*false*	.8	.2
false	*true*	.7	.3
false	*false*	.6	.4

Figure 3. Prm represented by dynamic Bayesian network.

works. Dynamic Bayesian networks [7,21] were introduced to model time dependent random variables $X_1, \ldots, X_k$ (which, again, may be assumed to be Boolean). A dynamic Bayesian network is given by a standard Bayesian network that represents the joint distribution of $X_1, \ldots, X_k$ at time $t = 0$, and a standard Bayesian network fragment that represents the joint distribution of $X_1, \ldots, X_k$ for $t \geqslant 1$ conditional on their values at $t - 1$ (only discrete time models are used). In figure 3 the two network fragments defining a dynamic Bayesian network for X_1, X_2 are highlighted. For any $t_{\max} \in \mathbb{N}$ one can concatenate $t_{\max}$ copies of the network fragment for $t \geqslant 1$ with the initial network for $t = 0$, and thereby obtain a standard Bayesian network for the Boolean random variables $X_1(0), X_2(0), \ldots, X_2(t_{\max})$. Thus, a dynamic Bayesian network represents a prm for $X_1, \ldots, X_k$, now interpreted as unary relation symbols. Instances of this prm are given by domains $D = \{0, 1, \ldots, t_{\max}\}$ of time points.

This does not quite fit our provisional definition of a prm, as there it was demanded that a prm defines a ppm for every finite domain D. A dynamic Bayesian network, however, requires that the domain consists of points in time. This domain need not necessarily consist of natural numbers $0, \ldots, t_{\max}$. All that is really needed is that the elements of the domain are totally ordered: when it is given, for instance, that *blue* < *green* < *red*, then the dynamic Bayesian network of figure 3 will also define a joint probability distribution of $X_1(\mathit{blue}), \ldots, X_2(\mathit{red})$.

This leads to a crucial modification of our provisional concept of prms: different instances of the prm, in general, are not given by unstructured domains $D = \{d_1, \ldots, d_n\}$, but by structures $\mathcal{D} = (D, S)$, consisting of a finite domain D, and a set S of *predefined relations* on D.

For a fixed (finite) domain D, and a set R of relation symbols, we denote by $\mathrm{Mod}_D(R)$ the set of all R-structures over D, i.e., the set of all algebraic structures with domain D and interpretations of the symbols $r \in R$.[1] A structure $\mathcal{D} \in \mathrm{Mod}_D(R)$ can be identified with an assignment of truth values *true, false* to all ground atoms $\mathtt{r}(d_{i_1}, \ldots, d_{i_l})$ ($\mathtt{r} \in R$, $d_{i_1}, \ldots, d_{i_l} \in D$, $l =$ arity of $\mathtt{r}$). We also refer to this set of ground atoms as the ground atoms of $\mathrm{Mod}_D(R)$. This also means that a probability distribution over $\mathrm{Mod}_D(R)$ can be identified with a joint probability distribution of the ground atoms of $\mathrm{Mod}_D(R)$ (viewed as Boolean random variables). We can now give the final definition of a prm.

Definition 1.1. Let R, S be two sets of relation symbols. The elements of R are called the *probabilistic relations*; the elements of S are called the *predefined relations*. A probabilistic relational model for R and S is a partial mapping P that assigns to S-structures $\mathcal{D}$ with finite domain D a probability distribution $P(\mathcal{D})$ over $\mathrm{Mod}_D(R)$. In the sequel we write $P_{\mathcal{D}}$ for $P(\mathcal{D})$.

By the equivalence of $P_{\mathcal{D}}$ with a joint probability distribution of the ground atoms of $\mathrm{Mod}_D(R)$, we can still view $P_{\mathcal{D}}$ as a ppm. As before, we call each $P_{\mathcal{D}}$ an instance of the prm. A prm will usually not define $P_{\mathcal{D}}$ for all S-structures $\mathcal{D}$: a dynamic Bayesian network, for instance, can be described as a prm with $S = \{<\}$, such that $P_{\mathcal{D}}$ is defined iff $<$ is a total order on D.

The predefined symbols in S are restricted to relation symbols just as a matter of convenience: this simplifies some of our subsequent definitions. As functions and constants can always be encoded as a relation, this definition really also provides for predefined functions and constants, and we will use them freely in examples.

Probabilistic relational models are a fundamental semantic construct that appear in several places in discrete mathematics and computer science. Random graphs, for instance, are studied from a theoretical point of view in combinatorics, and are applied in average case analysis (see [9] for an overview). Models for random graphs are prms in the sense of definition 1.1 (with $S = \emptyset$ and R containing a single binary (edge) relation). As a semantic model for probabilistic knowledge representation in AI, prms probably were first explicitly used in [14]. The term "probabilistic relational model" itself was introduced by Friedman et al. [8].

To make practical use in AI applications of the semantic notion of prms, two questions have to be answered:

Representation: How can prms be represented within a formal syntactical framework?
Inference: What algorithms can be used to answer queries about a prm?

For ppms the currently favored answers to the corresponding questions are: use Bayesian networks and their inference algorithms. For prms no similarly clear-cut answers have yet emerged. In the following two subsections we will briefly review the

[1] In the following, we will usually not pursue a strict distinction between a (syntactic) symbol and its (semantic) interpretation. Thus, according to context, $d \in D$ is both a constant symbol, and the element of a semantic domain designated by d; $\mathtt{r} \in R$ is both a relation symbol, and a relation defined on some domain.

different answers that have been proposed so far in the literature, and outline our answers, which then are presented in detail in the remainder of this paper.

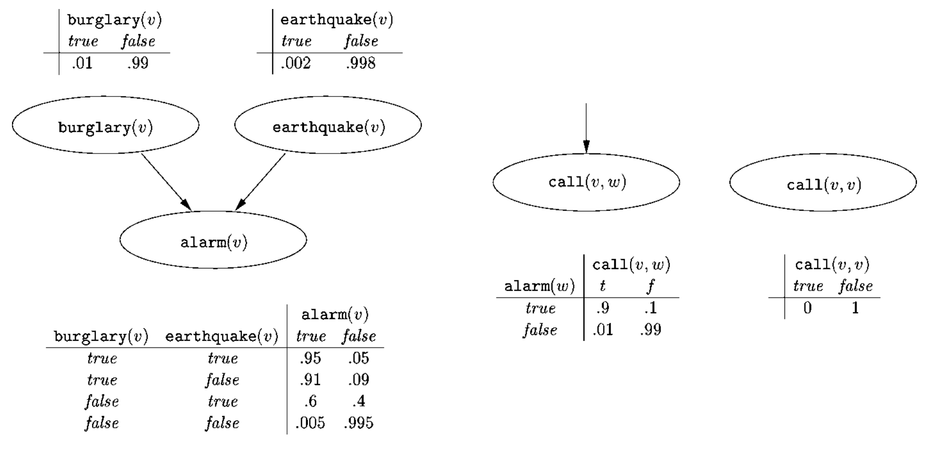
1.2. Representation paradigms

Few systems presented in the literature were developed precisely for the representation of prms in the sense of definition 1.1. Some variations in their individual semantics notwithstanding, the representation of prms nevertheless can be identified as a common core functionality of a number of different systems. In the sequel we discuss these systems only with regard to this core functionality.

1.2.1. Network templates

One possible method for representing prms we already encountered in dynamic Bayesian networks: network templates. By network templates we mean any representation that consists of a number of generic Bayesian network fragments, whose nodes can be instantiated with the ground atoms of a particular domain. Another example of a representation framework that is essentially a template representation are the *probabilistic frame based systems* of Koller and Pfeffer [19].

Figure 4 shows a simple template representation of our introductory prm. The nodes in these templates are labelled with (non-ground) atoms $\mathtt{alarm}(v)$, $\mathtt{call}(v, w), \ldots$, where v, w are logical variables. A finite domain D gives rise to the set of instantiations of these templates by performing all possible substitutions of domain elements (constants) for the variables (where distinct domain elements are to be substituted for distinct variables in a template). The resulting collection of standard Bayesian network fragments can be put together in a unique way to form a single standard Bayesian network. For $D = \{\textit{Holmes}, \textit{Watson}\}$ the templates of figure 4 yield the network of figure 2.



burglary(v) true	burglary(v) false
.01	.99

earthquake(v) true	earthquake(v) false
.002	.998

burglary(v)	earthquake(v)	alarm(v) true	alarm(v) false
true	*true*	.95	.05
true	*false*	.91	.09
false	*true*	.6	.4
false	*false*	.005	.995

alarm(w)	call(v, w) t	call(v, w) f
true	.9	.1
false	.01	.99

call(v, v) true	call(v, v) false
0	1

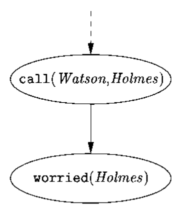
Figure 4. Template representation of introductory prm.

Network templates can only represent a restricted class of prms. To understand their basic limitation, consider the generic node alarm(v) in figure 4. Assume that D is a domain containing *Holmes*. The standard Bayesian network defined by the templates of figure 4 for this domain contains the node alarm(*Holmes*) with parents burglary(*Holmes*) and earthquake(*Holmes*) (cf. figure 2). In particular, the set of parents of alarm(*Holmes*) does not depend on D. Similarly, if D also contains *Watson*, then the node call(*Watson*, *Holmes*) only depends on alarm(*Holmes*), irrespective of the other elements in the domain. This is an intrinsic limitation of template representations: by specifying conditional probability distributions in the templates with probability tables, the structure of the parent sets of nodes is fixed for every instance of the template, and cannot depend on the domain.

However, the set of parents of a ground atom may very well be domain dependent in a realistic model. For example, add an additional relation symbol worried to our vocabulary. The ground atom worried(*Holmes*) stands for the fact that Holmes is worried because one of his neighbors has called and informed him that his alarm bell is ringing. Figure 5 shows the node worried(*Holmes*) in two instances of a prm given by the domains {*Holmes, Watson*} and {*Holmes, Watson, Gibbon*}. The conditional probability tables at worried(*Holmes*) are different in the two instances, which therefore cannot be obtained as instantiations of network templates labelled with fixed conditional probability tables. What is needed to represent complex prms, therefore, are (finitary) representations of the (infinite) classes of conditional probability tables needed in the different instances of the prm.

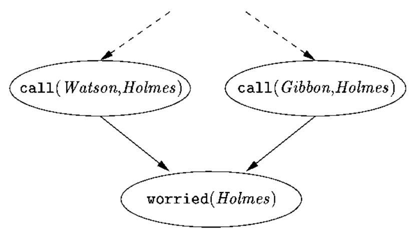
1.2.2. Rule based systems

A number of representation systems have been developed that are based on some form of probabilistic logic programs [5,11,20,24,25]. The most advanced of these systems are the *probabilistic knowledge bases* of Ngo and Haddawy [20]. The exact definitions of syntax and semantics given in [20] are rather involved, so that here we can



call(*Watson, Holmes*)	worried(*Holmes*) *true*	*false*
true	.9	.1
false	.05	.95

(a)



call(*Watson, Holmes*)	call(*Gibbon, Holmes*)	worried(*Holmes*) *true*	*false*
true	*true*	.99	.01
true	*false*	.9	.1
false	*true*	.9	.1
false	*false*	.05	.95

(b)

Figure 5. The worried(*Holmes*) node in two different domains.

only give an impression of the characteristics of Ngo and Haddawy's system by showing how it can be used to represent our example prm. The central syntactic constructs in a probabilistic knowledge base are *probabilistic rules* like

$$P\big(\texttt{worried}(v) = \textit{true} \mid \texttt{call}(w, v) = \textit{true}\big) = 0.9. \tag{1}$$

The variables v, w in this rule are again to be instantiated with the elements of a particular domain. For $D = \{\textit{Holmes}, \textit{Watson}, \textit{Gibbon}\}$ we obtain among the instantiations of the rule $P(\texttt{worried}(\textit{Holmes}) = \textit{true} \mid \texttt{call}(\textit{Watson}, \textit{Holmes}) = \textit{true}) = 0.9$ and $P(\texttt{worried}(\textit{Holmes}) = \textit{true} \mid \texttt{call}(\textit{Gibbon}, \textit{Holmes}) = \textit{true}) = 0.9$. These instantiated rules can be thought of as partial specifications of the conditional probability table of figure 5(b). To define all the probability values needed in the full table, the rule (1) is supplemented with a *combining rule*. By declaring the combining rule

$$\textit{combining-rule}(\texttt{worried}) = \textit{noisy-or}, \tag{2}$$

for instance, one would obtain as the definition of the conditional probability table for the node $\texttt{worried}(\textit{Holmes})$ in a model for the domain $D = \{\textit{Holmes},\ d_1, \ldots, d_k\}$

$$P\big(\texttt{worried}(\textit{Holmes}) = \textit{true} \mid \texttt{call}(d_1, \textit{Holmes}) = \alpha_1, \ldots, \texttt{call}(d_k, \textit{Holmes}) = \alpha_k\big)$$
$$= 1 - (1 - 0.9)^{l(\alpha_1, \ldots, \alpha_k)},$$

where $\alpha_i \in \{\textit{true},\ \textit{false}\}$, and $l(\alpha_1, \ldots, \alpha_k) = |\{i \mid \alpha_i = \textit{true}\}|$. Taken together, (1) and (2) say that any call by a neighbor will cause Holmes to be worried with probability 0.9, and that the effect of calls by several neighbors are independent, so that when l different neighbors call, the probability of Holmes to be worried is $1 - (1 - 0.9)^l$.

For the particular domain $\{\textit{Holmes, Watson, Gibbon}\}$ (1) and (2) will generate the first three lines of the conditional probability table of figure 5(b). The last line, however, will be $P(\texttt{worried}(\textit{Holmes}) = \textit{true} \mid \texttt{call}(\textit{Watson}, \textit{Holmes}) = \textit{false}, \texttt{call}(\textit{Gibbon}, \textit{Holmes}) = \textit{false}) = 0$. If we want to have a "base probability" 0.05 of being worried even when no one has called instead, this would have to be accomplished by a suitable modification of the combining rule.

As this example illustrates, the probabilistic knowledge base approach overcomes the limitation of network templates in that domain dependent dependency structures and probability values can be represented. The probabilistic rules can provide a flexible and intuitive way to specify aspects of a prm in a modular way. In general, however, the probabilistic rules have to be regarded with a great deal of caution. Their intuitiveness is to some extent illusory because a probabilistic rule alone has no declarative semantics: in spite of its suggestive syntax, the rule (1) in itself does not define a conditional probability. What it says about the probabilities in a prm depends on the associated combining rule. The more complicated a prm becomes, which one wants to represent with a probabilistic knowledge base, the more complicated combining rules have to be used, and the less transparent the meaning of the numerical parameters specified in the probabilistic rules will be.

1.2.3. Relational Bayesian networks

Relational Bayesian networks were introduced in [14]. The main tool for the representation of prms used in relational Bayesian networks is the *probability formula*. Probability formulas are functional expressions that define the entries in the conditional probability tables for the different instances of the prm. A suitable probability formula could be used to define, for instance,

$$P(\mathtt{worried}(v) = t) = \begin{cases} 1 - (1 - 0.9)^l & \text{if } |\{w \mid \mathtt{call}(w, v)\}| = l > 1, \\ 0.05 & \text{if } \neg\exists w \mathtt{call}(w, v). \end{cases} \tag{3}$$

This definition then generates the probability tables for $\mathtt{worried}(\mathit{Holmes})$ in figure 5.

The expression on the right hand side of (3) only is a high-level representation of a probability formula. Probability formulas in the strict sense are constructed in a formal syntax that consists of only four elementary construction rules. This reduction of the basic representation syntax to a small number of primitive constructs has two main advantages: first, the semantics of the representation system becomes straightforward and transparent. Second, the representation framework is quite amenable to theoretical investigations of its properties, as many basic questions can be answered by a four-step induction on the construction of probability formulas.

In section 2 we will review syntax and semantics of probability formulas and relational Bayesian networks. These definitions were already given in [14]. In that paper two types of relational Bayesian networks were distinguished: (plain) relational Bayesian networks (RBNs), and recursive relational Bayesian networks (RRBNs). The former can only be used to represent prms without predefined relations ($S = \emptyset$ in definition 1.1), whereas the latter allow predefined relations. In preceding papers [15,16] the focus was on RBNs, as these are significantly simpler than RRBNs, and have a number of interesting theoretical properties, which are not shared by RRBNs. In the present paper, we turn to RRBNs, as they are much more expressive than RBNs. Section 3.1 highlights with some examples the ability of RRBNs to represent complex prms.

1.3. Inference

For any representation of a prm one would like to answer the following questions: given an S-structure $\mathcal{D}$, is $P_{\mathcal{D}}$ defined by the prm? If so, and given ground atoms $\mathtt{r}_0(d_0), \mathtt{r}_1(d_1), \ldots, \mathtt{r}_k(d_k)$ of $\mathrm{Mod}_D(R)$ and $\alpha_i \in \{\mathit{true}, \mathit{false}\}$ $(1 \leqslant i \leqslant k)$, what is the value of

$$P_{\mathcal{D}}(\mathtt{r}_0(d_0) = \mathit{true} \mid \mathtt{r}_1(d_1) = \alpha_1, \ldots, \mathtt{r}_k(d_k) = \alpha_k)? \tag{4}$$

If $P_{\mathcal{D}}$ is defined, being a ppm, it can be represented with a standard Bayesian network. The standard approach to solve the two inference problems therefore is to try to construct a standard Bayesian network representing $P_{\mathcal{D}}$, so that the construction fails iff $P_{\mathcal{D}}$ is undefined. When a standard network representing $P_{\mathcal{D}}$ has been constructed, one can compute probabilities (4) using the standard algorithms. In fact, the connection between representations of prms and the construction of standard Bayesian networks representing instances of the prm originally was so close, that representations of prms were identified with construction rules for standard Bayesian networks. The process of constructing individual standard Bayesian networks as instances of a general specification has become known as *knowledge based model construction* [27].

Given a high-level representation of a prm, it is not immediately clear, however, that one cannot do better than solving inference problems by constructing (possibly very large) standard networks first. In [14] hope was expressed that for RBNs more high-level and efficient inference procedures could be found. Unfortunately, it turned out that the complexity of the given inference problems is inherently so high, that one cannot do better than standard network construction [17].

In this paper, therefore, we will take a closer look at standard network construction as an inference procedure for RRBNs. We develop in section 4 a method for the construction of small standard Bayesian networks representing instances $P_{\mathcal{D}}$. “Small” here means that the size of the standard network only grows polynomially in the size of $\mathcal{D}$. As inference in standard Bayesian networks, although of exponential complexity in the worst case [6], has turned out to be tractable in practice, this is a step towards making the computation of probabilities (4) tractable in many practical cases. Another step in the direction of making these computations feasible is presented in section 5: there it is shown how one can avoid to construct a full network representing $P_{\mathcal{D}}$, by constructing a network directly for the conditional distribution $P_{\mathcal{D}}(\cdot \mid \mathtt{r}_1(d_1) = \alpha_1, \ldots, \mathtt{r}_k(d_k) = \alpha_k)$, and by limiting this construction to a fragment of the full network which is relevant for the node $\mathtt{r}_0(d_0)$.

2. Recursive relational Bayesian networks

In this section we reintroduce syntax and semantics of recursive relational Bayesian networks. Apart from some slight generalizations the definitions given here are the same as in [14].

We begin with fixing some notational conventions. Throughout we will be concerned with two sets R, S of relation symbols, as given in definition 1.1. To mark the distinction between the two sets, we use standard mathematical fonts and symbols ($s, r, \leqslant, \ldots$) for the relations in S, and typewriter font ($\mathtt{X}, \mathtt{r}, \mathtt{alarm}, \ldots$) for the relations in R.

The arity of any relation r is denoted by $|r|$. Throughout, we use boldface $\boldsymbol{v}, \boldsymbol{d}, \ldots$ to denote tuples of variables or domain elements. The length of a tuple $\boldsymbol{v}$ is denoted by $|\boldsymbol{v}|$. Expressions like $\boldsymbol{u} \subseteq \boldsymbol{v}$ and $w \in \boldsymbol{v}$ are to be interpreted by identifying a tuple $\boldsymbol{v} = (v_1, \ldots, v_n)$ with the set $\{v_i \mid 1 \leqslant i \leqslant n\}$ of its components.

As observed in section 1.1, a distribution $P_{\mathcal{D}}$ on $\mathrm{Mod}_D(R)$ can be identified with a joint probability distribution of the ground atoms of $\mathrm{Mod}_D(R)$. When $a_1, \ldots, a_m$ is an enumeration of these ground atoms, then we can factorize the joint probability of the events $a_i = \alpha_i$ ($\alpha_i \in \{false, true\}$, $1 \leqslant i \leqslant m$) in the usual way:

$$P_{\mathcal{D}}(a_1 = \alpha_1, \ldots, a_m = \alpha_m) = \prod_{i=1}^{m} P_{\mathcal{D}}(a_i = \alpha_i \mid a_1 = \alpha_1, \ldots, a_{i-1} = \alpha_{i-1}). \tag{5}$$

This identity holds for all instantiations $\alpha_1, \ldots, \alpha_m$, so that we can also write it as an identity between distributions:

$$P_{\mathcal{D}}(a_1, \ldots, a_m) = \prod_{i=1}^{m} P_{\mathcal{D}}(a_i \mid a_1, \ldots, a_{i-1}). \tag{6}$$

Conditional independencies will often allow us to simplify the conditional distributions $P_{\mathcal{D}}(a_i \mid a_1, \ldots, a_{i-1})$ to conditional distributions $P_{\mathcal{D}}(a_i \mid Pa(a_i))$, where $Pa(a_i)$ is a subset of $\{a_1, \ldots, a_{i-1}\}$. Then

$$P_{\mathcal{D}}(a_1, \ldots, a_m) = \prod_{i=1}^{m} P_{\mathcal{D}}\bigl(a_i \mid Pa(a_i)\bigr). \tag{7}$$

From (7) we derive our strategy for the representation of a prm: for every S-structure $\mathcal{D}$, and every atom a_i of $\mathrm{Mod}_D(R)$, our representation will define a function $P_{\mathcal{D}}(a_i \mid Pa(a_i))$ that maps instantiations (i.e., truth assignments) of a set $Pa(a_i)$ of atoms into [0, 1]. *Probability formulas* are our representation language for functions of this form.

Two key ingredients of probability formulas are *S-constraints* and *combination functions*. S-constraints are the tool employed by probability formulas to make use of predefined relations.

Definition 2.1. Let S be a set of relation symbols. An *S-constraint* is any Boolean combination of atomic formulas that can be constructed from logical variables $u, v, \ldots,$ the identity relation $=$, and relation symbols from S. We write $c(\boldsymbol{v})$ for an S-constraint that only contains variables from the tuple $\boldsymbol{v}$ (but not necessarily all the variables in the $\boldsymbol{v}$). We use τ to denote an arbitrary tautological constraint, e.g., $u = u$.

More succinctly, we can also say that an S-constraint is a quantifier free S-formula (in the sense of first-order logic). Combination functions are the main tool for numerical computations.

Definition 2.2. A *combination function* is any function that maps finite multisets with elements from [0, 1] into [0, 1].

We use braces $\Rbrace$, $\Lbrace$ to denote multisets: if $q_i \in [0, 1]$ for all i from some index set Γ, then $\Lbrace q_i \mid i \in \Gamma \Rbrace$ denotes the multiset that contains $|\{i \in \Gamma \mid q_i = r\}|$ copies of $r \in [0, 1]$. We also use the notation $\Lbrace r_1 : k_1, \ldots, r_n : k_n \Rbrace$ for the multiset that contains k_i copies of the number r_i ($1 \leqslant i \leqslant n$), or explicit enumerations $\Lbrace q_1, \ldots, q_l \Rbrace$. Thus, when $q_1 = q_2 = 0.3$, and $q_3 = 0.8$, then $\{\!|q_i \mid i \in \{1, 2, 3\}|\!\} = \{\!|0.3 : 2, 0.8 : 1|\!\} = \{\!|0.3, 0.3, 0.8|\!\}$. Two important examples of combination functions are

$$\begin{aligned}
&\textit{noisy-or}: && \textit{n-o}\{\!|q_i \mid i \in \Gamma|\!\} := 1 - \prod_{i\in\Gamma}(1 - q_i),\\
&\textit{mean}: && \textit{mean}\{\!|q_i \mid i \in \Gamma|\!\} := \frac{1}{|\Gamma|}\sum_{i\in\Gamma} q_i.
\end{aligned}$$

For technical reasons, combination functions also have to be defined on the empty multiset. For *noisy-or* and *mean* we here employ the conventions $\textit{n-o}\emptyset = \textit{mean}\emptyset := 0$.

Probability formulas, now, are defined with respect to two relational vocabularies R and S.

Definition 2.3. Let R, S be sets of relation symbols. The class of R, *S-probability formulas* is inductively defined as follows.

(i) (Constants). Each $q \in [0, 1]$ is a probability formula.

(ii) (Indicator functions). For each $\mathtt{r} \in R$, and every $|\mathtt{r}|$-tuple $\boldsymbol{v}$ of variables, $\mathtt{r}(\boldsymbol{v})$ is a probability formula.

(iii) (Convex combinations). When F_1, F_2, F_3 are probability formulas, then so is $F_1F_2 + (1 - F_1)F_3$.

(iv) (Combination functions). When $F_1, \ldots, F_k$ are probability formulas, *comb* is any combination function, $\boldsymbol{v}$, $\boldsymbol{w}$ are tuples of variables, and $c(\boldsymbol{v}, \boldsymbol{w})$ is an S-constraint, then

$$\textit{comb}\{\!|F_1, \ldots, F_k \mid \boldsymbol{w}; c(\boldsymbol{v}, \boldsymbol{w})|\!\}$$

is a probability formula.

Special cases of convex combinations are products F_1F_2 (setting $F_3 = 0$) and the inverse $1 - F_1$ of a formula F_1 (setting $F_2 = 0$ and $F_3 = 1$).

Example 2.4. For $R = \{\mathtt{call}\}$ and $S = \emptyset$ an R, S-probability formula is

$$F(v) \equiv \textit{n-o}\{\!|0.9\mathtt{call}(w, v) \mid w; w \neq v|\!\}. \tag{8}$$

It is straightforward to define the set of free variables of a probability formula F: the free variables of $\mathtt{r}(\boldsymbol{v})$ are the variables in the tuple $\boldsymbol{v}$, the free variables of a convex combination are the union of the free variables of F_1, F_2, F_3, and the free variables of a combination function are the union of the free variables of $F_1, \ldots, F_k$ and c, minus the variables in $\boldsymbol{w}$ (thus, a combination function binds the variables $\boldsymbol{w}$). We write $F(\boldsymbol{v})$ for a probability formula whose free variables are among $\boldsymbol{v}$. An important measure for the complexity of a probability formula is its *quantifier depth*:

Definition 2.5. Let $F(\boldsymbol{v})$ be an R, S-probability formula. We define the *quantifier depth* $qd(F) \in \mathbb{N}$ inductively as follows: if F is a constant or an indicator function, then $qd(F) := 0$. If $F = F_1F_2 + (1 - F_1)F_3$ then $qd(F) := \max\{qd(F_i) \mid i = 1, 2, 3\}$. If $F = comb\{\!|F_1, \ldots, F_k \mid \boldsymbol{w}; c(\boldsymbol{v}, \boldsymbol{w})|\!\}$ then $qd(F) := \max\{qd(F_i) \mid i = 1, \ldots, k\} + |\boldsymbol{w}|$.

We now turn to the semantics of probability formulas. Eventually, they will be used to define the probabilities $P_{\mathcal{D}}(a_i \mid Pa(a_i))$ in (7), so we have to say how probability formulas determine such probability values. A probability formula $F(\boldsymbol{v})$ computes probabilities as a function of three inputs: an S-structure $\mathcal{D}$, a tuple $\boldsymbol{d}$ of domain elements from D which is substituted for $\boldsymbol{v}$, and the truth values of the ground R-atoms that are needed to evaluate indicator functions in F. An explicit representation of this set of ground R-atoms is obtained by defining for every symbol $\mathtt{r} \in R$ a first-order formula $pa_{F,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z})$ such that the evaluation of $F(\boldsymbol{d})$ depends on the atom $\mathtt{r}(\boldsymbol{d}')$ just when $pa_{F,\mathtt{r}}(\boldsymbol{d}, \boldsymbol{d}')$ holds.

Definition 2.6. Let $F(\boldsymbol{v})$ be an R, S-probability formula. Let $\mathtt{r} \in R$ and $\boldsymbol{z}$ an $|\mathtt{r}|$-tuple of variables that do not occur in $\boldsymbol{v}$. We define the first-order formula $pa_{F,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z})$ in the vocabulary S by induction on the structure of F:

(i) If $F \equiv q$ then $pa_{F,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z}) = \epsilon$, where ϵ denotes an arbitrary unsatisfiable formula.

(ii) If $F \equiv \tilde{\mathtt{r}}(\boldsymbol{u})$, then $\boldsymbol{u} \subseteq \boldsymbol{v}$, and we define

$$pa_{F,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z}) \equiv \begin{cases} \bigwedge_{i=1}^{|\mathtt{r}|} u_i = z_i & \text{if } \mathtt{r} = \tilde{\mathtt{r}}, \\ \epsilon & \text{otherwise.} \end{cases}$$

(iii) $F \equiv F_1F_2 + (1 - F_1)F_3$ then

$$pa_{F,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z}) \equiv pa_{F_1,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z}) \vee pa_{F_2,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z}) \vee pa_{F_3,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z}).$$

(iv) If $F \equiv comb\{\!|F_1, \ldots, F_k \mid \boldsymbol{w}; c(\boldsymbol{v}, \boldsymbol{w})|\!\}$ then

$$pa_{F,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z}) \equiv \exists \boldsymbol{w}\big(c(\boldsymbol{v}, \boldsymbol{w}) \wedge \big(pa_{F_1,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{z}) \vee \cdots \vee pa_{F_k,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{z})\big)\big).$$

In the case where $\boldsymbol{w} = \emptyset$ this simplifies to

$$pa_{F,\mathtt{r}}(\boldsymbol{v}, \boldsymbol{z}) \equiv c(\boldsymbol{v}) \wedge \big(pa_{F_1,\mathtt{r}}(\boldsymbol{v}_1, \boldsymbol{z}) \vee \cdots \vee pa_{F_k,\mathtt{r}}(\boldsymbol{v}_k, \boldsymbol{z})\big).$$

Example 2.7. For $F(v)$ as in example 2.4, we obtain

$$pa_{F,\mathtt{call}}(v, z_1, z_2) \equiv \exists w\big(w \neq v \wedge (w = z_1 \wedge v = z_2)\big)$$

(which is equivalent to $z_1 \neq v \wedge z_2 = v$). For

$$F'(v\} \equiv 0.4F(v) + 0.6\mathtt{alarm}(v) \tag{9}$$

we obtain $pa_{F',\texttt{call}}(v, z_1, z_2) \equiv z_1 \neq v \wedge z_2 = v$ as before, and $pa_{F',\texttt{alarm}}(v, z_1) \equiv v = z_1$ (in both cases after some simplifications of the result obtained from the recursive definition 2.6).

For a given S-structure $\mathcal{D} \in \text{Mod}_D(S)$, and for a tuple $\boldsymbol{d} \in D^{|\boldsymbol{v}|}$, the formula $pa_{F,\texttt{r}}(\boldsymbol{v}, \boldsymbol{z})$ defines the set of ground $\texttt{r}$-atoms

$$Pa_{\texttt{r}}\big(F(\boldsymbol{d})[\mathcal{D}]\big) := \big\{\texttt{r}(\boldsymbol{e}) | \mathcal{D} \models pa_{F,\texttt{r}}(\boldsymbol{d}, \boldsymbol{e})\big\} \tag{10}$$

and the set of R-atoms

$$Pa\big(F(\boldsymbol{d})[\mathcal{D}]\big) := \bigcup_{\texttt{r} \in R} Pa_{\texttt{r}}\big(F(\boldsymbol{d})[\mathcal{D}]\big). \tag{11}$$

Example 2.8. For $F'(v)$ as given by (9), and $\mathcal{D}$ given by $D = \{Holmes, Watson, Gibbon\}$ (S being empty), we obtain

$$Pa_{\texttt{call}}\big(F'(Holmes)[\mathcal{D}]\big) = \big\{\texttt{call}(Watson, Holmes), \texttt{call}(Gibbon, Holmes)\big\},$$
$$Pa_{\texttt{alarm}}(F'(Holmes)[\mathcal{D}]\big) = \big\{\texttt{alarm}(Holmes)\big\}.$$

Often we will drop the explicit reference to the underlying structure $\mathcal{D}$, and simply write $Pa(F(\boldsymbol{d}))$ for $Pa(F(\boldsymbol{d})[\mathcal{D}])$. Given a truth assignment for all atoms in $Pa(F(\boldsymbol{d}))$, the formula F now defines a probability value for $\boldsymbol{d}$.

Definition 2.9. Let $F(\boldsymbol{v})$ be a R, S-probability formula. Let $\mathcal{D} \in \text{Mod}_D(S)$, $\boldsymbol{d} \in D^{|\boldsymbol{v}|}$, and I an instantiation for all ground atoms in $Pa(F(\boldsymbol{d})[\mathcal{D}])$ with truth values $\{true, false\}$. We then define $F(\boldsymbol{d})[\mathcal{D}, I] \in [0, 1]$ inductively as follows:

(i) $F \equiv q$: $F(\boldsymbol{d})[\mathcal{D}, I] := q$.

(ii) $F \equiv \texttt{r}(\boldsymbol{u})$ (with $\boldsymbol{u} \subseteq \boldsymbol{v}$):

$$F(\boldsymbol{d})[\mathcal{D}, I] := \begin{cases} 1 & \text{if } I(\texttt{r}(\widetilde{\boldsymbol{d}})) = true, \\ 0 & \text{if } I(\texttt{r}(\widetilde{\boldsymbol{d}})) = false, \end{cases}$$

where $\widetilde{\boldsymbol{d}}$ is the tuple of domain elements obtained by substituting for each variable v_i in $\boldsymbol{u}$ the domain element d_i.

(iii) $F \equiv F_1F_2 + (1 - F_1)F_3$:

$$F(\boldsymbol{d})[\mathcal{D}, I] := F_1(\boldsymbol{d})[\mathcal{D}, I]F_2(\boldsymbol{d})[\mathcal{D}, I] + \big(1 - F_1(\boldsymbol{d})[\mathcal{D}, I]\big)F_3(\boldsymbol{d})[\mathcal{D}, I].$$

(iv) $F = comb\{\!|F_1, \ldots, F_k \mid \boldsymbol{w}; c(\boldsymbol{v}, \boldsymbol{w})|\!\}$:

$$F(\boldsymbol{d})[\mathcal{D}, I] := comb\{\!|F_1(\boldsymbol{d}, \boldsymbol{e})[\mathcal{D}, I], \ldots, F_k(\boldsymbol{d}, \boldsymbol{e})[\mathcal{D}, I] \mid \boldsymbol{e} \in D^{|\boldsymbol{w}|}; \mathcal{D} \models c(\boldsymbol{d}, \boldsymbol{e})|\!\}.$$

In the case $\boldsymbol{w} = \emptyset$ this simplifies to

$$F(\boldsymbol{d})[\mathcal{D}, I] := \begin{cases} comb\{\!|F_1(\boldsymbol{d})[\mathcal{D}, I], \ldots, F_k(\boldsymbol{d})[\mathcal{D}, I]|\!\} & \text{if } \mathcal{D} \models c(\boldsymbol{d}), \\ comb\,\emptyset & \text{otherwise.} \end{cases}$$

That $F(\boldsymbol{d})[\mathcal{D}, I]$ is defined when I instantiates $Pa(F(\boldsymbol{d})[\mathcal{D}])$ follows by straightforward induction on the structure of F. Again, we usually drop the reference to $\mathcal{D}$, and write $F(\boldsymbol{d})[I]$ for $F(\boldsymbol{d})[\mathcal{D}, I]$.

Example 2.10. Let F' and $\mathcal{D}$ be as in example 2.8. Let $I(\texttt{call}(\textit{Watson}, \textit{Holmes})) = \textit{true}$, $I(\texttt{call}(\textit{Gibbon}, \textit{Holmes})) = \textit{false}$, and $I(\texttt{alarm}(\textit{Holmes})) = \textit{true}$. Then

$$F'(\textit{Holmes})[\mathcal{D}, I] = 0.4\textit{n-o}\{\!|0, 0.9|\!\} + 0.6 = 0.96.$$

Example 2.11. Consider the formula

$$F(v, w, u) \equiv \textit{n-o}\{\!|0.6\texttt{r}(v, w) \mid \emptyset; w < v|\!\}. \tag{12}$$

For $(d_1, d_2, d_3) \in D^3$ we now have $Pa_{\texttt{r}}(F(d_1, d_2, d_3)[\mathcal{D}]) = \{\texttt{r}(d_1, d_2)\}$ if $\mathcal{D} \models d_2 < d_1$, and $Pa_{\texttt{r}}(F(d_1, d_2, d_3)) = \emptyset$ if $\mathcal{D} \not\models d_2 < d_1$. In the first case, we need for the evaluation of $F(d_1, d_2, d_3)$ an instantiation I of $\texttt{r}(d_1, d_2)$, and obtain

$$F(d_1, d_2, d_3)[\mathcal{D}, I] := \begin{cases} \textit{n-o}\{\!|0.6|\!\} = 1 - 0.4 & \text{if } I(\texttt{r}(d_1, d_2)) = \textit{true}, \\ \textit{n-o}\{\!|0|\!\} = 0 & \text{otherwise.} \end{cases}$$

In the second case we obtain $F(d_1, d_2, d_3)[\mathcal{D}] = \textit{n-o}\,\emptyset = 0$. As the variable u does not actually appear on the right-hand side of (12), the value $F(d_1, d_2, d_3)[\mathcal{D}, I]$ does not depend on d_3.

The following probability formula has a somewhat different flavor:

$$F(v) \equiv \textit{mean}\{\!|\textit{n-o}\{\!|1 \mid \emptyset; v = w|\!\} \mid w; \tau|\!\}. \tag{13}$$

As F does not contain any indicator functions, we have $Pa(F(d)[\mathcal{D}]) = \emptyset$ for all $\mathcal{D}, d$. To see what formula (13) does, first consider the inner subformula $F'(v, w) \equiv \textit{n-o}\{\!|1 \mid \emptyset; v = w|\!\}$. For any $d, d' \in D$ we obtain that $F'(d, d') = 1$ if $d = d'$, and 0 else. To evaluate $F(d)$ we have to evaluate $F'(d, d')$ for every $d' \in D$, and then apply *mean* to the resulting multiset. This, however, will be the multiset $\{\!|1 : 1, 0 : |D| - 1|\!\}$, the mean of which is $1/|D|$. Thus, $F(d)[\mathcal{D}]$ is just the inverse of the domain size (for every d).

To define probability distributions on $\mathrm{Mod}_D(R)$, we now assign to each $\texttt{r} \in R$ exactly one probability formula.

Definition 2.12. Let R, S be sets of relation symbols. A *recursive relational Bayesian network for R with predefined S* is a set

$$\Phi = \left\{F_{\texttt{r}}(v_1, \ldots, v_{|\texttt{r}|}) \mid \texttt{r} \in R\right\}$$

of R, S-probability formulas.

(Non-recursive) relational Bayesian networks (RBNs), which were the main focus of [14–16] are a special case of recursive relational Bayesian networks (RRBNs) defined by the following two restrictions: (1) $S = \emptyset$, so that the constraints $c(\boldsymbol{v}, \boldsymbol{w})$ in combination functions only use the equality relation, (2) the relations in R are arranged in a directed acyclic graph, and each formula $F_{\mathtt{r}}$ only contains indicator functions $\mathtt{s}(\boldsymbol{v})$ for symbols $\mathtt{s}$ that are parents of $\mathtt{r}$ in the graph. Relational Bayesian networks without this restriction were called "recursive" in [14] because the probability for $\mathtt{r}$-atoms may depend recursively on other $\mathtt{r}$-atoms.

The second of the restrictions that distinguish the narrower class of RBNs makes it clear why this form of representation was called a "network". In the generalized form of definition 2.12 this term is no longer a good description of the representation, but was here retained to remain consistent with earlier work.

Definition 2.9 is the cornerstone of the semantics of a RRBN Φ. For the numbers $F_{\mathtt{r}}(\boldsymbol{d})[\mathcal{D}, I]$ to induce a probability distribution $P_{\mathcal{D}}$, it only is required that an appropriate acyclicity condition holds:

Definition 2.13. Let Φ be an R, S-recursive relational Bayesian network, $\mathcal{D}$ an S-structure with finite domain D. Φ and $\mathcal{D}$ induce a *dependency relation* $\prec$ on the atoms of $\mathrm{Mod}_D(R)$ via

$$\mathtt{s}(\boldsymbol{d}') \prec \mathtt{r}(\boldsymbol{d}) \quad \text{iff} \quad \mathtt{s}(\boldsymbol{d}') \in Pa\big(F_{\mathtt{r}}(\boldsymbol{d})[\mathcal{D}]\big).$$

If the relation $\prec$ is acyclic, then a probability distribution $P^{\Phi}_{\mathcal{D}}$ is defined on $\mathrm{Mod}_D(R)$ via

$$P^{\Phi}_{\mathcal{D}} := \prod_{\mathtt{r}\in R} \prod_{\boldsymbol{d}\in D^{|\mathtt{r}|}} P^{\Phi}_{\mathcal{D}}\big(\mathtt{r}(\boldsymbol{d}) \mid Pa(F_{\mathtt{r}}(\boldsymbol{d})[\mathcal{D}])\big),$$

where the conditional distributions on the right are given by

$$P^{\Phi}_{\mathcal{D}}\big(\mathtt{r}(\boldsymbol{d}) = \mathit{true} \mid I\big) := F_{\mathtt{r}}(\boldsymbol{d})[\mathcal{D}, I]$$

for each instantiation I of $Pa(F_{\mathtt{r}}(\boldsymbol{d})[\mathcal{D}])$.

Example 2.14. To illustrate the definitions given so far, we show how to represent as a RRBN the prm given by the dynamic Bayesian network of figure 3.

Let $S = \{zero, s\}$, where $zero$ is a constant, and s a binary relation. The network of figure 3 defines a distribution $P^{*}_{\mathcal{D}}$ on $\mathrm{Mod}_D(\{\mathtt{X}_1, \mathtt{X}_2\})$ for every S-structure $\mathcal{D}$ that interprets $zero$ and s as the minimal element and the successor relation, respectively, of a linearly ordered domain. We have to define a RRBN $\Phi = \{F_{\mathtt{X}_1}, F_{\mathtt{X}_2}\}$, such that $P^{\Phi}_{\mathcal{D}} = P^{*}_{\mathcal{D}}$ for all $\mathcal{D}$ of this form. In the following, $\mathcal{D}$ is an arbitrary such structure. In particular, for every $d \in \mathcal{D}, d \neq zero$, there is a unique element $d - 1 \in D$ with $s(d - 1, d)$.

The definition of $F_{\mathtt{X}_1}$ is simple: just let $F_{\mathtt{X}_1} \equiv 0.3$. For $F_{\mathtt{X}_2}$ we first define a probability formula

$$Z(v) := n\text{-}o\{\!|1 \mid \emptyset; v = zero|\!\}$$

which for every $\mathcal{D}$ and $d \in D$ has the value $Z(d)[\mathcal{D}] = 1$ if $\mathcal{D} \models d = \mathit{zero}$, and $Z(d)[\mathcal{D}] = 0$ else. The probability formula $F_{\mathtt{X}_2}$ then has the structure

$$F_{\mathtt{X}_2} \equiv Z(v)F_0(v) + \bigl(1 - Z(v)\bigr)F_t(v)$$

with subformulas $F_0(v)$ and $F_t(v)$ that determine the probability of $\mathtt{X}_2(d)$ for $d = \mathit{zero}$ and $d \neq \mathit{zero}$, respectively.

The first of these is a straightforward representation of the probability table for $\mathtt{X}_2(0)$ of figure 3:

$$F_0(v) :\equiv 0.2\mathtt{X}_1(v) + 0.4\bigl(1 - \mathtt{X}_1(v)\bigr).$$

For $F_t(v)$ things are slightly more complicated, because here we have to access the truth value of $\mathtt{X}_2(v-1)$. This is done via the subformula

$$T(v) :\equiv \textit{n-o}\{\!|\mathtt{X}_2(w) \mid w; s(w, v)|\!\}.$$

This formula has the following properties: for $d \in D$ we have $\mathit{Pa}(T(d)) = \emptyset$ if $d = \mathit{zero}$, and $\mathit{Pa}(T(d)) = \{\mathtt{X}_2(d-1)\}$ if $d \neq \mathit{zero}$. In the latter case we have

$$T(d)\bigl[\mathtt{X}_2(d-1) \mapsto \mathit{true}\bigr] = 1 \quad \text{and} \quad T(d)\bigl[\mathtt{X}_2(d-1) \mapsto \mathit{false}\bigr] = 0.$$

Thus, $T(d)[I]$ is 1 if $d \neq \mathit{zero}$ and $I(\mathtt{X}_2(d-1)) = \mathit{true}$, and 0 else (the use of *n-o* as a combination function here is quite arbitrary: any combination function with $\mathit{comb}\emptyset = \mathit{comb}\{\!|0|\!\} = 0$ and $\mathit{comb}\{\!|1|\!\} = 1$ would do). With $T(v)$ we can now represent the conditional probability table for $\mathtt{X}_2(t)$ in the formula $F_t(v)$ as

$$F_t(v) :\equiv T(v)0.7 + \bigl(1 - T(v)\bigr)\bigl(\mathtt{X}_1(v)0.8 + \bigl(1 - \mathtt{X}_1(v)\bigr)0.6\bigr).$$

$F_t(v)$ is an illustration how probability formulas naturally encode *context-specific independence* [4]: the fact that $\mathtt{X}_2(d)$ only depends on $\mathtt{X}_1(d)$ if $I(\mathtt{X}_2(d-1)) = \mathit{false}$ is directly reflected in the structure of the formula $F_t(v)$. In section 5 it will be shown how this structural aspect of probability formulas helps us to develop methods for more efficient inference.

3. Representations

We illustrate the expressive power of RRBNs from two perspectives: in section 3.1 from a practical perspective by showing how to solve realistic non-trivial representation problems, in section 3.2 from a theoretical perspective by showing that, in principle, every probabilistic relational model can be represented by a RRBN.

3.1. Practical modelling

We use as a running example the following scenario: a bicycle race of k participants is to be monitored over $t_{\max}$ time steps. We are interested in making probability assessments of the following kind: given that at time t participant a is ahead of b, what is the probability that at time $t+3$ b is ahead of a? Given that a finishes among the first 10% of participants, what is the probability that he will be qualified for the next race?

It is not our goal to construct a whole coherent RRBN for this scenario. We will only use individual aspects of this toy example to illustrate how RRBNs solve some very general modelling problems that are relevant in many different application domains. The temporal aspect of our example makes it related to dynamic Bayesian networks (and, in fact, it is partly inspired by the application of dynamic Bayesian networks for traffic monitoring [13]). It goes beyond dynamic Bayesian networks, however, in that at each time step we are dealing with a random relational structure over a domain of k participants.

3.1.1. Sorted domains

The domains of structures $\mathcal{D}$ for which $P_{\mathcal{D}}$ is to be defined consist of time points $0, 1, \ldots, t_{\max}$, and participants $p_1, \ldots, p_k$. Atoms whose probabilities are to be assessed will be of the form $\texttt{ahead}(t, p_2, p_7)$ or $\texttt{in\_shape}(p_3)$. The probabilities for these atoms must obviously be defined in a way that distinguishes time point arguments from participant arguments. To do this with RRBNs we assume that predefined unary relations T, P are given whose interpretations in $\mathcal{D}$ are just the time points and participants, respectively. A probability formula that says that a participant is in shape with probability 0.7 (and a time point is in shape with probability 0) then is

$$F_{\texttt{in\_shape}}(v) \equiv 0.7\textit{n-o}\{\!|1 \mid \emptyset;\, Pv|\!\}. \tag{14}$$

Again, the use of *noisy-or* here is quite arbitrary, as any combination function with $\textit{comb}\{\!|1|\!\} = 1$ and $\textit{comb}\,\emptyset = 0$ would do.

Once we have ascertained that certain constructs are expressible in the strict syntax of probability formulas, we can introduce appropriate syntactic abbreviations to increase the readability of the formulas. As a first convention, we abbreviate the formula $\textit{n-o}\{\!|1 \mid \emptyset;\, c(\boldsymbol{v})|\!\}$ ($c(\boldsymbol{v})$ an arbitrary S-constraint) by $c(\boldsymbol{v})$ (the *indicator* of c: $c(\boldsymbol{d})[\mathcal{D}] = 1$ if $\mathcal{D} \models c(\boldsymbol{d})$, and $c(\boldsymbol{d})[\mathcal{D}] = 0$ if $\mathcal{D} \not\models c(\boldsymbol{d})$). Formula (14) then simplifies to

$$F_{\texttt{in\_shape}}(v) \equiv 0.7Pv. \tag{15}$$

Sort constraints on variables can also be encoded even more conveniently by obvious conventions on the variable names, so that (15) becomes even simpler

$$F_{\texttt{in\_shape}}(p) \equiv 0.7.$$

3.1.2. First-order conditioning

Once we have defined a random relation $\texttt{ahead}(t, p_1, p_2)$ standing for p_1 to be ahead of p_2 at time t, we may next want to represent the relation $\texttt{leader}(t, p)$ standing for p leading the field at time t. The relation $\texttt{leader}$ is deterministically defined in terms of the relation $\texttt{ahead}$: $\texttt{leader}(t, p) \Leftrightarrow \neg\exists p'\texttt{ahead}(t, p', p)$. This can be done in a probability formula as follows:

$$F_{\texttt{leader}}(t, p) \equiv 1 - \textit{n-o}\{\!|\texttt{ahead}(t, p', p) \mid p';\, \tau|\!\}.$$

The subformula $n\text{-}o\{\!|\,\mathtt{ahead}(t, p', p) \mid p'; \tau\,|\!\}$ evaluates to 1 if $\mathtt{ahead}(t, p', p)$ holds for at least one participant p', and to 0 else. Thus it is the indicator of the formula $\exists p' \mathtt{ahead}(t, p', p)$. This is inverted by $1 - \cdots$, obtaining the indicator for $\neg\exists p' \mathtt{ahead}(t, p', p)$.

More generally, by using the noisy-or construct for existential quantification, the construct $1 - \cdots$ for negation, multiplication for conjunction, indicators of definition 2.3(ii) for atomic R-formulas, and indicators $c(\boldsymbol{v})$ (as discussed above) for atomic S-formulas, we can define for any first-order formula $\phi(\boldsymbol{v})$ over $R \cup S$ a probability formula $F_\phi(\boldsymbol{v})$, such that in all S-structures $\mathcal{D}$, for all $\boldsymbol{d} \in D^{|\boldsymbol{v}|}$, and for all instantiations I of the atomic R-formulas on which the truth value of $\phi(\boldsymbol{d})$ depends, we have

$$F_\phi(\boldsymbol{d})[I] = \begin{cases} 1 & \text{if } (\mathcal{D}, I) \models \phi(\boldsymbol{d}), \\ 0 & \text{else.} \end{cases}$$

Extending the conventions adopted above for S-constraints, we simply use $\phi(\boldsymbol{v})$ to denote the probability formula F_ϕ. We can then write, for instance,

$$F(t, p) \equiv 0.7\big(\neg\exists p' \mathtt{ahead}(t, p', p)\big) + 0.1\big(\exists p' \mathtt{ahead}(t, p', p)\big),$$

which evaluates to 0.7 if p is the leader at time t, and to 0.1 else. Note that $F_{\neg\phi} = 1 - F_\phi$, so that formulas of the form $\phi(\boldsymbol{v})F_2 + \neg\phi(\boldsymbol{v})F_3$ are a special form of convex combinations in the sense of definition 2.3.

3.1.3. Proportions

One of the primary concerns of probabilistic extensions of first-order logic in AI [1, 12] was to provide a framework for reasoning with statistical probabilities, where (on finite domains) the statistical probability of some property usually is identified with the proportion of domain elements that have the property. To reason with such proportions Bacchus [1] introduces the syntactic construct

$$[\phi(\boldsymbol{v}, \boldsymbol{w}) \mid \psi(\boldsymbol{v}, \boldsymbol{w})]_{\boldsymbol{w}} \tag{16}$$

to represent the proportion of tuples $\boldsymbol{w}$ that satisfy the formula $\phi(\boldsymbol{v}, \boldsymbol{w})$ among those $\boldsymbol{w}$ that satisfy $\psi(\boldsymbol{v}, \boldsymbol{w})$ (in other words, the conditional statistical probability for $\boldsymbol{w}$ of $\phi(\boldsymbol{v}, \boldsymbol{w})$ given $\psi(\boldsymbol{v}, \boldsymbol{w})$; this conditional probability is parameterized by the free variables $\boldsymbol{v}$).

Proportions are a very useful tool for our model representation task: consider, for instance, the relation $\mathtt{qualified}(p)$ representing that p qualifies for the next race. The probability of $\mathtt{qualified}(p)$ will usually depend on the relative position in which p finishes, i.e., on the proportion of p' with $\mathtt{ahead}(t_{\max}, p', p)$ (assuming here that the race is over at time $t_{\max}$, and the relation $\mathtt{ahead}(t_{\max}, \cdot, \cdot)$ stores the order of finishing). This proportion can be represented with the probability formula

$$F(p) \equiv \mathit{mean}\{\!|\,\mathtt{ahead}(t_{\max}, p', p) \mid p'; \tau\,|\!\}.$$

We could now define

$$F_{\texttt{qualified}}(p) \equiv 1 - F(p),$$

making the probability of qualifying inversely proportional to the relative position of finishing.

More generally, for any S-constraint $c(\boldsymbol{v}, \boldsymbol{w})$, and any R-formula $\phi(\boldsymbol{v}, \boldsymbol{w})$, we have the probability formula

$$[\phi(\boldsymbol{v}, \boldsymbol{w}) \mid c(\boldsymbol{v}, \boldsymbol{w})]_{\boldsymbol{w}} :\equiv mean\{\!|\phi(\boldsymbol{v}, \boldsymbol{w}) \mid \boldsymbol{w}; c(\boldsymbol{v}, \boldsymbol{w})|\!\}. \tag{17}$$

For $\boldsymbol{d} \in D^{|w|}$ then $[\phi(\boldsymbol{d}, \boldsymbol{w}) \mid c(\boldsymbol{d}, \boldsymbol{w})]_{\boldsymbol{w}}$ evaluates to the conditional statistical probability for $\boldsymbol{w}$ of $\phi(\boldsymbol{d}, \boldsymbol{w})$ given $c(\boldsymbol{d}, \boldsymbol{w})$.

This construct is asymmetric in that the first argument of the formula (17) is required to be an R-formula, and the second a (quantifier-free) S-formula. We could generalize the construct to also allow R-formulas $\psi(\boldsymbol{v}, \boldsymbol{w})$ as conditioning propositions. The representation of such a conditional probability $[\phi(\boldsymbol{v}, \boldsymbol{w}) \mid \psi(\boldsymbol{v}, \boldsymbol{w})]_{\boldsymbol{w}}$, however, seems to require combination functions for which our subsequent results in section 4 are not applicable. For this reason we do not pursue such more general constructs for statistical probabilities here.

3.1.4. Functional relations

So far we have not addressed the key modelling problem in our example: how to model the random relation ahead, in particular, how to maintain ahead(t, p, p') over time, i.e., how to define ahead$(t, \cdot, \cdot)$ conditional on ahead$(t-1, \cdot, \cdot)$ so that ahead$(t, \cdot, \cdot)$ is guaranteed to be an order relation on P provided that ahead$(t-1, \cdot, \cdot)$ is such an order relation, and the transition from $t-1$ to t models a random change in the order.

Our solution to this modelling problem illustrates the use of functional relations, and how to encode them with probability formulas (a (partial) functional relation from D^k to D^l is a relation r$(\boldsymbol{v}, \boldsymbol{w})$ on D^{k+l} such that for all $\boldsymbol{d} \in D^k$ there exists at most one $\boldsymbol{d}' \in D^l$ with r$(\boldsymbol{d}, \boldsymbol{d}')$).

Our model for the transition from ahead$(t-1, \cdot, \cdot)$ to ahead$(t, \cdot, \cdot)$ is based on the assumption that the time units are so small that in the interval $[t-1, t]$ at most one event of the form "participant p passes participant p'" takes place. The model then consists of the random selection of one participant p, and advancing him by one position in the order ahead.

A random selection is the special case of a random functional relation with $k = 0$, $l = 1$, i.e., a unary relation select for which select(p) holds for at most one p. To define a random selection with probability formulas, we assume that the domain elements of sort P, too, are equipped with some predefined strict total order $<_P$. In the sequel we take this order to correspond with the naming of the elements, i.e., $p_1 <_P p_2 <_P \cdots <_P p_k$. Intuitively, we can define a random selection by going through the elements of P in the order $<_P$, and for each p_i assign select(p_i) the conditional probability $1/(k-i+1)$ given that no $p_j <_P p_i$ has already been selected, and conditional probability 0 else. Then for all p_i:

$$\begin{aligned}
P\big(\texttt{select}(p_i)=1\big) &= \big(1-P\big(\texttt{select}(p_1)=1\big)\big)\big(1-P(\texttt{select}(p_2)=1)\big)\cdots \\
&\quad \big(1-P\big(\texttt{select}(p_{i-i})=1\big)\big)P\big(\texttt{select}(p_i)=1\big) \\
&= \left(1-\frac{1}{k}\right)\left(1-\frac{1}{k-1}\right)\cdots\left(1-\frac{1}{k-i+2}\right)\frac{1}{k-i+1} \\
&= \frac{1}{k}.
\end{aligned}$$

As in our model one selection takes place at every point in time t, the relation `select` needs to have a second argument of sort T. Its probabilistic model now is encoded in the probability formula

$$F_{\texttt{select}}(t,p) \equiv \neg\exists p'\big(p' <_P p \wedge \texttt{select}(t,p')\big) mean\{\!|\, p=\tilde{p} \mid \tilde{p};\, p <_P \tilde{p} \vee p=\tilde{p} \,|\!\}. \tag{18}$$

The factor $mean\{\!| \ldots |\!\}$ here counts the elements $\tilde{p}$ with $p \leqslant_P \tilde{p}$ in a similar way as (13) counted domain elements. For $p = p_i$ it thus returns the value $1/(k-i+1)$. This factor is preceded by a logical expression that by the conventions introduced above stands for a formula that evaluates to 0 or 1 according to whether $\texttt{select}(t,p')$ is true for some $p' <_P p$.

Once the relation $\texttt{select}(t-1,\cdot)$ has been determined, the relation $\texttt{ahead}(t,\cdot,\cdot)$ can be defined for $t > 0$ by a purely logical probability formula:

$$\begin{aligned}
F_{\texttt{ahead},t}(t,p,p') \equiv\; &\big(\texttt{select}(t-1,p) \wedge in\_front(t-1,p',p)\big) \\
&\vee \big(\neg\big(\texttt{select}(t-1,p) \wedge in\_front(t-1,p',p)\big) \\
&\wedge \texttt{ahead}(t-1,p,p')\big).
\end{aligned}$$

Here $in\_front(t-1,p',p)$ is an abbreviation for a subformula that says that at time $t-1$ p' was directly in front of p. A full probabilistic model for `ahead` can then be given by a formula

$$F_{\texttt{ahead}}(t,p,p') \equiv (t=0)F_{\texttt{ahead},0}(p,p') + (t>0)F_{\texttt{ahead},t}(t,p,p'),$$

where $F_{\texttt{ahead},0}(p,p')$ is a formula that generates a random initial order. This can be done with similar functional constructs as the `select` relation used in $F_{\texttt{ahead},t}$.

The way a functional relation is defined by (18) easily generalizes to the case where a functional relation $\texttt{r}(\boldsymbol{v},\boldsymbol{w})$ with the following properties is to be defined: the domain of `r` is defined by an R,S-formula $\phi(\boldsymbol{v})$, the range of `r` is defined by an S-constraint $c(\boldsymbol{w})$, and for $\boldsymbol{d}$ with $\phi(\boldsymbol{d})$ all $\boldsymbol{d}'$ with $c(\boldsymbol{d}')$ are equally likely to be the unique value for which $\texttt{r}(\boldsymbol{d},\boldsymbol{d}')$ is true. A probability formula that defines such a functional `r` then is:

$$\begin{aligned}
&F_{\texttt{r}}(\boldsymbol{v},\boldsymbol{w}) \\
&\quad\equiv \phi(\boldsymbol{v})\neg\exists\boldsymbol{w}'(\boldsymbol{w}' < \boldsymbol{w} \wedge \texttt{r}(\boldsymbol{v},\boldsymbol{w}'))mean\{\!|\,\boldsymbol{w}=\tilde{\boldsymbol{w}} \mid \tilde{\boldsymbol{w}};\,(\boldsymbol{w} < \tilde{\boldsymbol{w}} \vee \boldsymbol{w}=\tilde{\boldsymbol{w}}) \wedge c(\boldsymbol{w})\,|\!\}
\end{aligned}$$

(the relation $\boldsymbol{w}' < \boldsymbol{w}$ now is some predefined total order on D^l).

As in the case of proportion formulas above, we here have an asymmetry in that the domain of a functional relation can be defined by an arbitrary R, S-formula, its range only with a S-constraint. As above, this could be generalized with the aid of suitable combination functions other than *mean* and *noisy-or*.

It should be pointed out that the functional relations we here have considered are fundamentally different from random functions with a predefined set of possible values, which is fixed for all domains (this is the concept of a random function to be found, e.g., in [19,20]). This is a much simpler case of a random function (or functional relation), and can easily be handled in the RRBN framework by adding to the predefined relations S a constant symbol for each of the possible function values.

3.2. A completeness theorem

The theorem proved in this section says that basically every prm can be represented by a RRBN. Some qualifications are necessary however. The first qualification is that RRBNs can only represent prms that are *compatible with isomorphisms*. Loosely speaking, a prm is compatible with isomorphisms if for any two isomorphic S-structures $\mathcal{D}, \mathcal{D}'$ the two distributions $P_{\mathcal{D}}$ and $P_{\mathcal{D}'}$ also are isomorphic. The precise condition is as follows: when $\mathcal{D}, \mathcal{D}'$ are S-structures with domains D and D', $\mathcal{C}$ and $\mathcal{C}'$ are R-structures over the same domains D and D', respectively, and $i : D \rightarrow D'$ is both an isomorphism from $\mathcal{D}$ to $\mathcal{D}'$ and from $\mathcal{C}$ to $\mathcal{C}'$, then $P_{\mathcal{D}}(\mathcal{C}) = P_{\mathcal{D}'}(\mathcal{C}')$ The second qualification is that we require a predefined total order $<$ on the domain. Finally, we shall only consider prms that are *total for ordered structures*, i.e., for every $\mathcal{D} \in \mathrm{Mod}_D(S)$ in which $<$ is interpreted as a strict linear order on D, we have that $P_{\mathcal{D}}$ is defined. While RRBNs can also represent a large class of prms that are not total in this sense, an exact statement of what types of partial prms can be represented would introduce additional technicalities that we here want to avoid.

Theorem 3.1. Let R, S be disjoint relational vocabularies, with $< \in S$ a binary relation symbol. Let P be a prm that is compatible with isomorphisms and total for ordered structures. There exists an R, S-RRBN Φ with $P^{\Phi}_{\mathcal{D}} = P_{\mathcal{D}}$ for all ordered $\mathcal{D}$.

Proof. To simplify the proof we assume that all relations in $R \cup S \setminus \{<\}$ have the same arity $q \geqslant 1$. The proof of the general case is basically the same, it only requires a larger load of notation.

Let $\mathcal{D} \in \mathrm{Mod}_D(S)$ with $<$ interpreted as a linear order on D. Let $|D| = n$. Let $R = \{\mathtt{r}_1, \ldots, \mathtt{r}_k\}$, $S \setminus \{<\} = \{s_1, \ldots, s_l\}$. A strict linear order $\prec$ is defined on the atoms of $\mathrm{Mod}_D(R)$ by: $\mathtt{r}_i(\boldsymbol{d}) \prec \mathtt{r}_j(\boldsymbol{d}')$ iff $i < j$ or $i = j$ and $\boldsymbol{d}$ precedes $\boldsymbol{d}'$ in the lexicographic order induced by $\leqslant$ on q-tuples of D. Let $a_1, \ldots, a_m$ be the enumeration of the atoms of $\mathrm{Mod}_D(R)$ according to the order $\prec$. The distribution $P_{\mathcal{D}}$ then is given by the conditional distributions $P_{\mathcal{D}}(a_i \mid a_1, \ldots, a_{i-1})$ $(1 \leqslant i \leqslant m$, cf. (6)). The straightforward approach now is to assign to the relation $\mathtt{r}_j \in R$ a probability formula $F_{\mathtt{r}_j}(\boldsymbol{v})$, such that for the atom $a_i = \mathtt{r}_j(\boldsymbol{d})$ we get $Pa(F_{\mathtt{r}_j}(\boldsymbol{d})) = \{a_1, \dots, a_{i-1}\}$, and for an instantiation I of $a_1, \dots, a_{i-1}$

$$F_{\mathtt{r}_j}(\boldsymbol{d})[\mathcal{D}, I] = P_{\mathcal{D}}\bigl(\mathtt{r}_j(\boldsymbol{d}) = \mathit{true} \mid I\bigr). \tag{19}$$

The right-hand side of (19) is a number in [0, 1] that depends on $\mathcal{D}$, I and $\boldsymbol{d}$. The plan, now, is to ruthlessly exploit the generality of the definition of combination functions, which allows us to define a combination function $\mathit{comb}_{P,j}$ that encodes the function P in such a manner that

$$F_{\mathtt{r}_j}(\boldsymbol{d})[\mathcal{D}, I] = \mathit{comb}_{P,j}(\mathcal{D}, I, \boldsymbol{d}) = P_{\mathcal{D}}\bigl(\mathtt{r}_j(\boldsymbol{d}) = \mathit{true} \mid I\bigr). \tag{20}$$

The problem now is that the middle term in (20) is not really meaningful, as a combination function is applied to a structured argument $(\mathcal{D}, I, \boldsymbol{d})$, not a [0, 1]-multiset. The main part of the proof of the theorem therefore will be to show how to code $(\mathcal{D}, I, \boldsymbol{d})$ as a multiset. This encoding, in turn, has to be done with probability formulas.

Essentially, $(\mathcal{D}, I, \boldsymbol{d})$ can be represented by an $(l + j) \times n^q$-matrix

$$X = (x_{\gamma,\delta})_{1 \leqslant \gamma \leqslant l+j, 1 \leqslant \delta \leqslant n^q}$$

whose rows are the characteristic vectors of the instantiations of the relations in $S \cup \{\mathtt{r}_1, \dots, \mathtt{r}_j\}$ given by $\mathcal{D}$ and I. The relation $\mathtt{r}_j$, however, is only partially instantiated by I: denoting by $\langle \boldsymbol{d} \rangle$ the index of $\boldsymbol{d}$ in the lexicographic order of D^q, we have that I instantiates the first $\langle \boldsymbol{d} \rangle - 1$ $\mathtt{r}_j$-atoms only. The precise definition of X now is given by

$$x_{\gamma,\delta} := \begin{cases} 1 & \text{if } 1 \leqslant \gamma \leqslant l, \text{ and } \mathcal{D} \models s_\gamma(\boldsymbol{d}') \text{ for } \boldsymbol{d}' \text{ with } \langle \boldsymbol{d}' \rangle = \delta, \\ 1 & \text{if } l + 1 \leqslant \gamma \leqslant l + j - 1, \text{ and } I(\mathtt{r}_\gamma(\boldsymbol{d}')) = 1 \text{ for } \boldsymbol{d}' \text{ with } \langle \boldsymbol{d}' \rangle = \delta, \\ 1 & \text{if } \gamma = l + j, \delta < \langle \boldsymbol{d} \rangle, \text{ and } I(\mathtt{r}_j(\boldsymbol{d}')) = 1 \text{ for } \boldsymbol{d}' \text{ with } \langle \boldsymbol{d}' \rangle = \delta, \\ 0 & \text{else.} \end{cases}$$

The first l rows of X code $\mathcal{D}$ only up to isomorphism, which is why from this point onward we cannot distinguish isomorphic S-structures. X can be turned into a multiset M_X via

$$M_X := \left\{\!\!\left| \frac{x_{\gamma,\delta}}{(\gamma - 1)n^q + \delta} \;\middle|\; 1 \leqslant \gamma \leqslant l + j,\ 1 \leqslant \delta \leqslant n^q \right|\!\!\right\}.$$

From M_X the matrix X can be reconstructed when the parameters $(l + j)$ and q are given: M_X contains a total number of $(l + j)n^q$ elements, which determines n, and thus the dimensions of X. The entry $x_{\gamma,\delta}$ is 1 iff M_X contains the number $1/(\gamma - 1)n^q + \delta$.

X, respectively M_X, does not fully encode the tuple $(\mathcal{D}, I, \boldsymbol{d})$, as it does not identify $\boldsymbol{d}$, or, equivalently, the domain of atoms on which I is defined. For this we also need to know the parameter $\langle \boldsymbol{d} \rangle$. We integrate $\langle \boldsymbol{d} \rangle$ into the multiset encoding by modifying M_X slightly to obtain

$$M(\mathcal{D}, I, \boldsymbol{d}) := \left\{\!\!\left| \left(\frac{1}{3}\right)^{\langle \boldsymbol{d} \rangle}, \frac{x_{\gamma,\delta}}{2((\gamma - 1)n^q + \delta)} \;\middle|\; 1 \leqslant \gamma \leqslant l + j,\ 1 \leqslant \delta \leqslant n^q \right|\!\!\right\}.$$

Here the original entries of M_X have all been divided by 2, so that the nonzero ones among them now are of the form $1/z$, z even, and can thereby be distinguished from the encoding $(1/3)^{\langle \boldsymbol{d} \rangle}$ of $\langle \boldsymbol{d} \rangle$. Thus, from $M(\mathcal{D}, I, \boldsymbol{d})$ both X and $\langle \boldsymbol{d} \rangle$, and hence $(\mathcal{D}, I, \boldsymbol{d})$ can be recovered.

We next define probability formulas whose evaluations will generate the elements of $M(\mathcal{D}, I, \boldsymbol{d})$. In particular, we will use a formula $F_{\langle\rangle}$ with

$$F_{\langle\rangle}(\boldsymbol{d})[\mathcal{D}, I] = \left(\frac{1}{3}\right)^{\langle \boldsymbol{d} \rangle},$$

formulas $F_\gamma(w_1, \ldots, w_q)$ for $1 \leqslant \gamma \leqslant l + j - 1$ with

$$F_\gamma(\boldsymbol{d}')[\mathcal{D}, I] = \frac{x_{\gamma, \langle \boldsymbol{d}' \rangle}}{2((\gamma - 1)n^q + \langle \boldsymbol{d}' \rangle)} \quad (\boldsymbol{d}' \in D^q),$$

and a formula $F_{l+j}(v_1, \ldots, v_q, w_1, \ldots, w_q)$ with

$$F_{l+j}(\boldsymbol{d}, \boldsymbol{d}')[\mathcal{D}, I] = \frac{x_{l+j, \langle \boldsymbol{d}' \rangle}}{2((l + j - 1)n^q + \langle \boldsymbol{d}' \rangle)} \quad (\boldsymbol{d}' \in D^q)$$

(note that the right-hand side of this equation depends on $\boldsymbol{d}$ via the definition of the $x_{l+j,\delta}$). Given such $F_{\langle\rangle}$ and F_γ we put

$$F_{\mathrm{r}_j}(\boldsymbol{v}) \equiv comb_{P,j}\{\!|F_{\langle\rangle}(\boldsymbol{v}), F_1(\boldsymbol{w}), \ldots, F_{l+j-1}(\boldsymbol{w}), F_{l+j}(\boldsymbol{v}, \boldsymbol{w}) \mid \boldsymbol{w}; \tau |\!\}. \tag{21}$$

For $\boldsymbol{d} \in D^q$ and I then

$$\begin{aligned}&\{\!|F_{\langle\rangle}(\boldsymbol{d})[\mathcal{D}, I], F_1(\boldsymbol{d}')[\mathcal{D}, I], \ldots, F_{l+j-1}(\boldsymbol{d}')[\mathcal{D}, I], F_{l+j}(\boldsymbol{d}, \boldsymbol{d}')[\mathcal{D}, I] \mid \boldsymbol{d}' \in D^q |\!\}\\ &\quad = M(\mathcal{D}, I, \boldsymbol{d}).\end{aligned}$$

We can now define for a multiset M

$$comb_{P,j} M := \begin{cases} P_{\mathcal{D}}(\mathrm{r}_j(\boldsymbol{d}) = true \mid I) & \text{if } M = M(\mathcal{D}, I, \boldsymbol{d}),\\ 1 & \text{otherwise.}\end{cases}$$

By the compatibility of P with isomorphisms, the value $P_{\mathcal{D}}(\mathrm{r}_j(\boldsymbol{d}) = true \mid I)$ only depends on the encoding $M(\mathcal{D}, I, \boldsymbol{d})$, so that $comb_{P,j}$ is well-defined. The choice of the value 1 in the else-case of this definition is arbitrary, as $comb_{P,j}$ will never need to be evaluated for multisets not encoding structures $(\mathcal{D}, I, \boldsymbol{d})$.

For F_{r_j} defined by (21) we then obtain the desired equality (19), and we are done.

It remains to show how to define $F_{\langle\rangle}$ and the F_γ. This can be done by using some additional tailor-made combination functions. For $F_{\langle\rangle}$ we use the combination function

$$power_{1/3} M := \begin{cases} \left(\frac{1}{3}\right)^k & \text{if } M = \{\!|1 : k|\!\}\ (k \in \mathbb{N} \cup \{0\}),\\ 1 & \text{otherwise,}\end{cases}$$

and put

$$F_{()}(\boldsymbol{v}) :\equiv power_{1/3}\{\!|1 \mid \boldsymbol{w};\, \boldsymbol{w} < \boldsymbol{v}|\!\}.$$

For $\gamma = 1, \ldots, l$ F_γ is defined as

$$F_\gamma(\boldsymbol{w}) :\equiv s_\gamma(\boldsymbol{w}) comb_\gamma\{\!|\tilde{\boldsymbol{w}} \leqslant \boldsymbol{w} \mid \tilde{\boldsymbol{w}};\, \emptyset|\!\}$$

with

$$comb_\gamma M :\equiv \begin{cases} \dfrac{1}{2((\gamma-1)(k+m)+k)} & \text{if } M = \{\!|1 : k, 0 : m|\!\}\ (k \geqslant 1,\ m \geqslant 0), \\ 0 & \text{else.} \end{cases}$$

The definition of F_γ for $\gamma = l+1, \ldots, l+j-1$ is similar. The definition of F_{l+j}, too, follows the same pattern, only that now the evaluation of r_j-atoms that are not instantiated by I must be prevented:

$$F_{l+j}(\boldsymbol{v}, \boldsymbol{w}) :\equiv n\text{-}o\{\!|\mathrm{r}_j(\boldsymbol{w}) \mid \emptyset;\, \boldsymbol{w} < \boldsymbol{v}|\!\} comb_{l+j}\{\!|\tilde{\boldsymbol{w}} \leqslant \boldsymbol{w} \mid \tilde{\boldsymbol{w}}, \emptyset|\!\}. \qquad \square$$

In the proof of the theorem the representation of the prm P by a RRBN has essentially been accomplished by encoding P with the very peculiar combination functions $comb_{P,j}$. This is rather orthogonal to the approach taken in section 3.1, where we have attempted to represent complex models with only a very limited supply of elementary combination functions, so that the representation is really in the structure of the probability formulas. This latter approach is clearly meant for probability formulas, and thus the general construction given in the proof of theorem 3.1 does not provide a pattern for the solution of actual modelling problems. Nevertheless, the result shows that the framework of probability formulas and combination functions, in principle, is general enough to cope with practically all modelling problems.

4. Inference

The inference problem is the following: given an R, S-RRBN Φ, and an S-structure $\mathcal{D}$, is $P^\Phi_{\mathcal{D}}$ defined? If $P^\Phi_{\mathcal{D}}$ is defined, and given an instantiation

$$\mathrm{r}_0(\boldsymbol{d}_0) = \alpha_0, \quad \mathrm{r}_1(\boldsymbol{d}_1) = \alpha_1, \quad \ldots, \quad \mathrm{r}_l(\boldsymbol{d}_l) = \alpha_l$$
$$\big(\alpha_i \in \{false, true\},\ \mathrm{r}_i \in R,\ \boldsymbol{d}_i \in D^{|\mathrm{r}_i|}\big),$$

of some atoms of $\mathrm{Mod}_D(R)$, what is the conditional probability

$$P^\Phi_{\mathcal{D}}\big(\mathrm{r}_0(\boldsymbol{d}_0) = \alpha_0 \mid \mathrm{r}_1(\boldsymbol{d}_1) = \alpha_1, \ldots, \mathrm{r}_l(\boldsymbol{d}_l) = \alpha_l\big)? \tag{22}$$

The solution to the first problem is straightforward: by definition 2.13, $P^\Phi_{\mathcal{D}}$ is defined iff the dependency relation $\prec$ on the atoms of $\mathrm{Mod}_D(R)$ is acyclic. It is convenient to view the relation $\prec$ as the edge relation in *a dependency graph*, defined as follows.

Definition 4.1. Let Φ be an R, S-RRBN, $\mathcal{D}$ an S-structure. The directed graph $G^{\Phi}_{\mathcal{D}}$ is defined as follows: the nodes of $G^{\Phi}_{\mathcal{D}}$ are the atoms of $\mathrm{Mod}_D(R)$; two nodes $\mathtt{r}(\boldsymbol{d})$ and $\mathtt{s}(\boldsymbol{d}')$ are joined by a directed edge from $\mathtt{s}(\boldsymbol{d}')$ to $\mathtt{r}(\boldsymbol{d})$ iff $\mathtt{s}(\boldsymbol{d}') \prec \mathtt{r}(\boldsymbol{d})$.

Given Φ and $\mathcal{D}$, the acyclicity of $G^{\Phi}_{\mathcal{D}}$ can be checked in time polynomial in $|D|$: for all $\mathtt{r}, \mathtt{s} \in R$ we generate the formulas $pa_{F_{\mathtt{r}},\mathtt{s}}$ according to definition 2.6 (in time independent of $|D|$). Then $G^{\Phi}_{\mathcal{D}}$ is created by checking for all pairs of atoms $\mathtt{r}(\boldsymbol{d})$, $\mathtt{s}(\boldsymbol{d}')$ whether $pa_{F_{\mathtt{r}},\mathtt{s}}(\boldsymbol{d}, \boldsymbol{d}')$ holds in $\mathcal{D}$. This check can be done naively in time $\mathrm{O}(|D|^{q}l)$, where q is the quantifier depth of $F_{\mathtt{r}}$, and l is the number of atoms in $pa_{F_{\mathtt{r}},\mathtt{s}}$ (using the slight idealization that we can check in constant time whether a ground S-atom holds in $\mathcal{D}$). Thus, we can create $G^{\Phi}_{\mathcal{D}}$ in time polynomial in $|D|$, and then test for acyclicity, again in polynomial time.

When $G^{\Phi}_{\mathcal{D}}$ turns out to be acyclic, we can use it as the underlying directed acyclic graph of a Bayesian network for $P^{\Phi}_{\mathcal{D}}$: to represent $P^{\Phi}_{\mathcal{D}}$ it is only needed to label each node $\mathtt{r}(\boldsymbol{d})$ in the graph with the probability formula $F_{\mathtt{r}}(\boldsymbol{d})$ (recall definition 2.13: $F_{\mathtt{r}}(\boldsymbol{d})$ induces a function that maps instantiations I of $Pa(F_{\mathtt{r}}(\boldsymbol{d}))$ to the conditional probability value $P^{\Phi}_{\mathcal{D}}(\mathtt{r}(\boldsymbol{d}) = \mathit{true} \mid I)$).

Example 4.2. Let $R = \{\mathtt{r}, \mathtt{s}\}$, $S = \emptyset$. Let Φ be the R-RRBN given by

$$\begin{aligned} F_{\mathtt{s}}(v, w) &\equiv 0.7, \\ F_{\mathtt{r}}(v) &\equiv \textit{n-o}\{\!|0.4\mathtt{s}(v, w) \mid w;\, w \neq v|\!\}. \end{aligned}$$

Let $D = \{d_1, \ldots, d_5\}$. One connected component of the graph $G^{\Phi}_{\mathcal{D}}$ is shown in figure 6. Labeling the nodes $\mathtt{s}(d_1, d_i)$ with 0.7, and the node $\mathtt{r}(d_1)$ with $\textit{n-o}\{\!|0.4\mathtt{s}(d_1, w) \mid w;\, w \neq d_1|\!\}$ turns the graph into a Bayesian network with conditional probability tables represented by probability formulas.

Generating a standard Bayesian network with the conditional probability distributions represented by probability formulas does not yet solve our inference problem, because standard inference algorithms for Bayesian networks require explicit table representations. There are several approaches one can take to overcome this problem: the first is simply to expand the probability formula representation to a full table representation by computing the value $F_{\mathtt{r}}(\boldsymbol{d})[I]$ for each instantiation I of $Pa(F_{\mathtt{r}}(\boldsymbol{d}))$. This, however, will usually lead to representations with a size exponential in the size of the

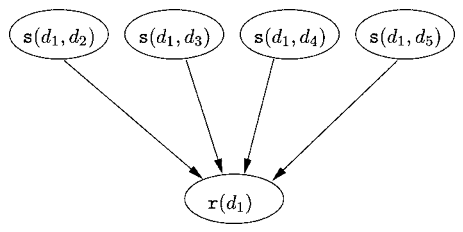


Figure 6. The first network representation of $P^{\Phi}_{\mathcal{D}}$.

domain: in example 4.2, for instance, this approach will lead to conditional probability tables for atoms $\mathtt{r}(d_i)$ of size $2^{|D|-1}$.

A second approach one could take is to develop a calculus that allows us to perform operations directly on probability formulas which are performed by standard inference algorithms on conditional probability tables. The basic operations here needed are multiplication and marginalization. Multiplication does not cause any problems, but marginalizing an atom $\mathtt{s}(\boldsymbol{d}') \in Pa(F_{\mathtt{r}}(\boldsymbol{d}))$ out of $F_{\mathtt{r}}(\boldsymbol{d})$ leads to a function on instantiations of $Pa(F_{\mathtt{r}}(\boldsymbol{d})) \setminus \{\mathtt{s}(\boldsymbol{d}')\}$ that is not again compactly representable by a probability formula.

We will here pursue a third approach, which promises to lead to tractable inference at least in many cases where the intractability caused by the construction of exponentially large conditional probability tables in the first approach does not reflect any inherent complexity of the inference problem. The basic idea behind the approach is to take the graph $G^{\Phi}_{\mathcal{D}}$ labelled with the probability formulas $F_{\mathtt{r}}(\boldsymbol{d})$, and to decompose it by adding additional variables, to form a graph $DG^{\Phi}_{\mathcal{D}}$ with the following properties: every atom of $\mathrm{Mod}_D(R)$ is a node in $DG^{\Phi}_{\mathcal{D}}$, the number of nodes in $DG^{\Phi}_{\mathcal{D}}$ is polynomial in the number of nodes in $G^{\Phi}_{\mathcal{D}}$, every node in $DG^{\Phi}_{\mathcal{D}}$ has at most 3 parents, and the nodes in $DG^{\Phi}_{\mathcal{D}}$ are labelled with conditional probability tables so that the (marginal) probability distribution induced by $DG^{\Phi}_{\mathcal{D}}$ on the atoms of $\mathrm{Mod}_D(R)$ again is $P^{\Phi}_{\mathcal{D}}$. The remainder of this section is devoted to develop a construction technique for such a network $DG^{\Phi}_{\mathcal{D}}$.

To motivate our approach for the construction of $DG^{\Phi}_{\mathcal{D}}$, consider figure 7. It shows the decomposition of the network in figure 6, which is here directly obtained by inserting an explicit representation of the causal model that originally motivated the *noisy-or* combination function (cf. [23, figure 4.20]). In detail, we have added a layer of variables $\mathtt{X}(d_i)$, which are "noisy" versions of the variables $\mathtt{s}(d_1, d_i)$ $(i = 2, \ldots, 5)$: the conditional probability of $\mathtt{X}(d_i) = \mathit{true}$ is 0.4 if $\mathtt{s}(d_1, d_i) = \mathit{true}$, and 0 else. The conditional probability of $\mathtt{r}_1(d_1)$ then becomes the deterministic *or* of the four variables $\mathtt{X}(d_2), \ldots, \mathtt{X}(d_5)$. A direct representation of this deterministic *or* as a conditional probability table for $\mathtt{r}(\boldsymbol{d})$ would still be exponential in size, but, by a standard construction, we can decompose the *or* with 4 inputs using two auxiliary "*or*-nodes" Z_1, Z_2, so that

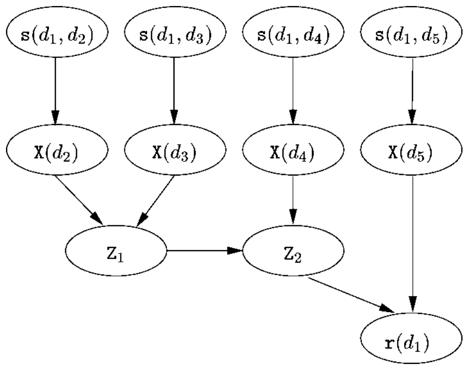


Figure 7. Decomposition of the network in figure 6.

all nodes in the resulting subnetwork only have 2 parents. One readily verifies that the conditional probability of $\mathtt{r}(d_1)$ given $\mathtt{s}(d_1, d_2), \ldots \mathtt{s}(d_1, d_5)$ in the network of figure 7 is the same as in the network of figure 6 (indeed, *noisy-or* has just been defined for this to be the case).

Our general strategy for decomposing the network $G^{\Phi}_{\mathcal{D}}$ now can be outlined as follows: given a node $\mathtt{r}(\boldsymbol{d})$ in $G^{\Phi}_{\mathcal{D}}$, which is labelled with a probability formula $F_{\mathtt{r}}(\boldsymbol{d}) = comb\{\!|F(\boldsymbol{d}, \boldsymbol{w}) \mid \boldsymbol{w}; c(\boldsymbol{d}, \boldsymbol{w})|\!\}$, proceed as follows:

(D1) For each $\boldsymbol{d}'$ with $c(\boldsymbol{d}, \boldsymbol{d}')$ create a new node $\mathtt{X}(\boldsymbol{d}')$ with parents $Pa(F(\boldsymbol{d}, \boldsymbol{d}'))$ labelled with $F(\boldsymbol{d}, \boldsymbol{d}')$.

(D2) Make the nodes $\mathtt{X}(\boldsymbol{d}')$ the parent set of $\mathtt{r}(\boldsymbol{d})$.

(D3) Insert a "computation network" for the computation of the conditional probability of $\mathtt{r}(\boldsymbol{d})$ given $\{\mathtt{X}(\boldsymbol{d}') \mid c(\boldsymbol{d}, \boldsymbol{d}')\}$ between $\mathtt{r}(\boldsymbol{d})$ and its parents, such that in the end $\mathtt{r}(\boldsymbol{d})$ and each of the added auxiliary variables have at most two parents.

Only for certain combination functions will such a decomposition be possible. Our first goal, therefore, will be to determine the class of these combination functions. For the following definitions and results some standard terminology from probability theory is required: a binary random variable X with $P(X = 1) = p$ is called $B(p)$-*distributed.* If X_i is $B(p_i)$-distributed $(1 \leqslant i \leqslant k)$, and the $X_1, \ldots, X_k$, are independent, then the joint distribution of the X_i is denoted by $B(p_1) \otimes \cdots \otimes B(p_k)$. When $f(x_1, \ldots, x_k)$ is a real-valued function, then $E[f(X_1, \ldots, X_k)]$ denotes the expected value of $f(X_1, \ldots, X_k)$. To make explicit that this expectation is with respect to a $B(p_1) \otimes \cdots \otimes B(p_k)$-distribution of the X_i we also write $E_{p_1,\ldots,p_k}[f(X_1, \ldots, X_k)]$.

Definition 4.3. A combination function *comb* is called an *expectation* if for all $n \geqslant 1$, for all $p_1, \ldots, p_n \in [0, 1]$, and for $B(p_1) \otimes \cdots \otimes B(p_n)$-distributed random variables $X_1, \ldots, X_n$ it holds that

$$E\big[comb\{\!|X_1, \ldots, X_n|\!\}\big] = comb\{\!|p_1, \ldots, p_n|\!\}. \tag{23}$$

A simple "syntactic" characterization of expectations can be based on the following definition.

Definition 4.4. A combination function is called *multilinear* if for all $n \geqslant 1$, and for all $i_1, \ldots, i_n \in \{0, 1\}$ there exists $\alpha_{i_1,\ldots,i_n} \in \mathbb{R}$, such that for all $p_1, \ldots, p_n \in [0, 1]$

$$comb\{\!|p_1, \ldots, p_n|\!\} = \sum_{(i_1,\ldots,i_n)\in\{0,1\}^n} \alpha_{i_1,\ldots,i_n} p_1^{i_1} \cdots p_n^{i_n}. \tag{24}$$

Theorem 4.5. A combination function is an expectation iff it is multilinear.

Proof. Assume first that *comb* is multilinear, and let $X_1, \ldots, X_n$ be independent $B(p_i)$-distributed random variables. Noting that $E[X^i] = E[X]^i$ for all integrable random variables X and $i \in \{0, 1\}$, we then obtain

$$\begin{aligned} E\big[comb\{\!|X_1, \ldots, X_n|\!\}\big] &= \sum_{(i_1,\ldots,i_n)} \alpha_{i_1,\ldots,i_n} E[X_1]^{i_1} \cdots E[X_n]^{i_n} \\ &= \sum_{(i_1,\ldots,i_n)} \alpha_{i_1,\ldots,i_n} p_1^{i_1} \cdots p_n^{i_n} \\ &= comb\{\!|p_1, \ldots, p_n|\!\}. \end{aligned}$$

For the converse direction, assume that *comb* is an expectation. We prove that *comb* is multilinear by induction on n. For $n = 1$ we have

$$E[comb\{\!|X_1|\!\}] = comb\{\!|1|\!\}p_1 + comb\{\!|0|\!\}(1 - p_1),$$

which is multilinear with $\alpha_0 = comb\{\!|0|\!\}$ and $\alpha_1 = comb\{\!|1|\!\} - comb\{\!|0|\!\}$.

Now let $n \geqslant 1$. Then for $B(p_1) \otimes \cdots \otimes B(p_n)$-distributed $X_1, \ldots, X_n$

$$\begin{aligned} &E\big[comb\{\!|X_1, \ldots, X_n|\!\}\big] \\ &\quad = E\big[comb\{\!|X_1, \ldots, X_n|\!\} \mid X_n = 0\big](1 - p_n) + E\big[comb\{\!|X_1, \ldots, X_n|\!\} \mid X_n = 1\big]p_n \\ &\quad = E\big[comb_0\{\!|X_1, \ldots, X_{n-1}|\!\}\big](1 - p_n) + E\big[comb_1\{\!|X_1, \ldots, X_{n-1}|\!\}\big]p_n \qquad (25) \end{aligned}$$

for combination functions $comb_0$ and $comb_1$ that satisfy

$$comb_i\{\!|X_1, \ldots, X_{n-1}|\!\} = comb\{\!|X_1, \ldots, X_{n-1}, i|\!\} \quad (i = 0, 1), \qquad (26)$$

and expectations now taken over the joint distribution of $X_1, \ldots, X_{n-1}$.

As *comb* is an expectation, the right-hand side of (25) is equal to $comb\{\!|p_1, \ldots, p_n|\!\}$, which therefore is of the form

$$comb_0^*\{\!|p_1, \ldots, p_{n-1}|\!\}(1 - p_n) + comb_1^*\{\!|p_1, \ldots, p_{n-1}|\!\}p_n, \qquad (27)$$

where

$$comb_i^*\{\!|p_1, \ldots, p_{n-1}|\!\} = E_{p_1,\ldots,p_{n-1}}\big[comb_i\{\!|X_1, \ldots, X_{n-1}|\!\}\big] \quad (i = 0, 1). \qquad (28)$$

We now need to show that the $comb_i^*$ are multilinear functions of $p_1, \ldots, p_{n-1}$, which then makes (27) multilinear in $p_1, \ldots, p_n$. To infer the multilinearity of the $comb_i^*$ from (28) and the induction hypothesis, we have to verify that

$$comb_i^* M = comb_i M \qquad (29)$$

for multisets M of the form $M = \{\!|1 : k, 0 : n - k - 1|\!\}$ $(0 \leqslant k \leqslant n - 1)$. Note that the $comb_i$ only are determined by (26) for arguments of this form, whereas the $comb_i^*$ are determined by (28) for all multisets of $n - 1$ elements from $[0, 1]$.

To show (29), let $0 \leqslant k \leqslant n - 1$, and put $p_1 = \cdots = p_k = 1$, $p_{k+1} = \cdots = p_{n-1} = 0$. For $B(p_i)$-distributed random variables X_i, (25) then becomes

$$comb_0\{\!|p_1, \ldots, p_{n-1}|\!\}(1 - p_n) + comb_1\{\!|p_1, \ldots, p_{n-1}|\!\}p_n \qquad (30)$$

which has to be equal to (27) for all $p_n \in [0, 1]$. But now (29) follows by letting $p_n = 1$, respectively $p_n = 0$.

Thus, combining (28) and (29) we obtain

$$comb_i^*\{\!|p_1, \ldots, p_{n-1}|\!\} = E_{p_1,\ldots,p_{n-1}}\big[comb_i^*\{\!|X_1, \ldots, X_{n-1}|\!\}\big]$$

for all $p_1, \ldots, p_{n-1}$, so that by induction hypothesis the $comb_i^*$ are multilinear for multisets of $n-1$ elements, and thus *comb* is multilinear for multisets of n elements. □

Example 4.6. *Noisy-or* and *mean* are multilinear, and hence are expectations.

For combination functions that are expectations we can now give a general definition of the first two decomposition steps (D1) and (D2), and show that the probability distribution on the nodes of the original network is not affected by the decomposition.

Definition 4.7. Let $\mathcal{D}$ be an S-structure, $\boldsymbol{d}$ a tuple of domain elements from D, and

$$F(\boldsymbol{d}) \equiv comb\{\!|F_1(\boldsymbol{d}, \boldsymbol{w}), \ldots, F_k(\boldsymbol{d}, \boldsymbol{w}) \mid \boldsymbol{w}; c(\boldsymbol{d}, \boldsymbol{w})|\!\}$$

an R, S-probability formula. Let $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_m$ be an enumeration of $\{\boldsymbol{d}' \in D^{|\boldsymbol{w}|} \mid \mathcal{D} \models c(\boldsymbol{d}, \boldsymbol{d}')\}$. Let G be a Bayesian network, $\mathtt{A}$ a node in G whose parents are $Pa(F(\boldsymbol{d}))$, and which is labelled with $F(\boldsymbol{d})$. We define a new network

$$G' = decompose1(G, \mathtt{A})$$

as follows (cf. figure 8):

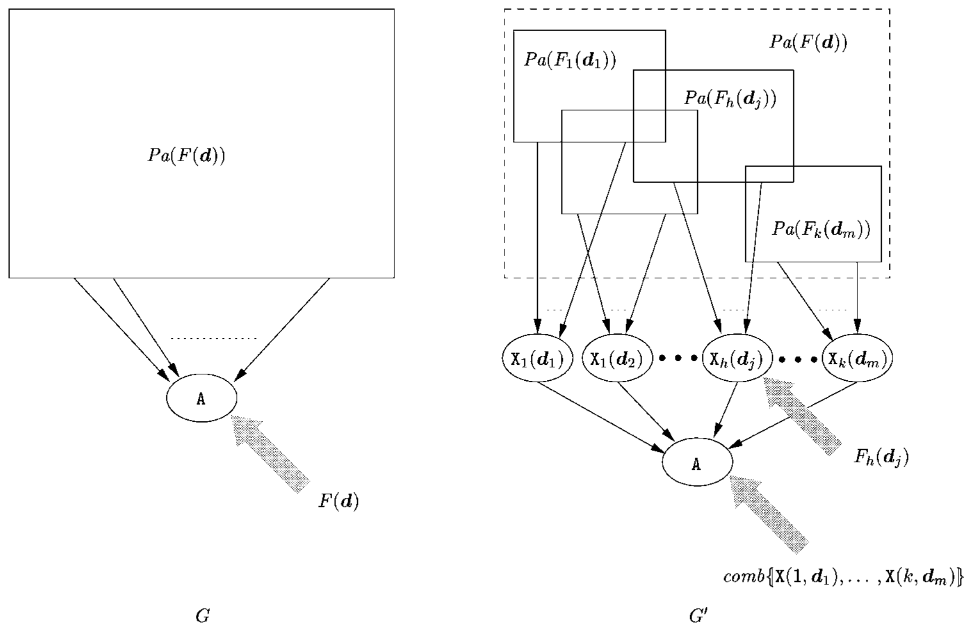

$Pa(F(\boldsymbol{d}))$
A
$F(\boldsymbol{d})$
G
$Pa(F(\boldsymbol{d}))$
$Pa(F_1(\boldsymbol{d}_1))$
$Pa(F_h(\boldsymbol{d}_j))$
$Pa(F_k(\boldsymbol{d}_m))$
$\mathtt{X}_1(\boldsymbol{d}_1)$
$\mathtt{X}_1(\boldsymbol{d}_2)$
$\mathtt{X}_h(\boldsymbol{d}_j)$
$\mathtt{X}_k(\boldsymbol{d}_m)$
$F_h(\boldsymbol{d}_j)$
A
$comb\{\mathtt{X}(1, \boldsymbol{d}_1), \ldots, \mathtt{X}(k, \boldsymbol{d}_m)\}$
G'

Figure 8. The first decomposition step.

(1) For $h = 1, \dots, k$ and $j = 1, \dots, m$ let $\mathtt{X}_h(\boldsymbol{d}_j)$ be a new Boolean random variable.

(2) Create a new graph by removing all edges between $Pa(F(\boldsymbol{d}))$ and $\mathtt{A}$, inserting an edge from each node in $Pa(F_h(\boldsymbol{d}, \boldsymbol{d}_j))$ to $\mathtt{X}_h(\boldsymbol{d}_j)$, and inserting an edge from each node $\mathtt{X}_h(\boldsymbol{d}_j)$ to $\mathtt{A}$.

(3) Label each node $\mathtt{X}_h(\boldsymbol{d}_j)$ with $F_h(\boldsymbol{d}, \boldsymbol{d}_j)$, and $\mathtt{A}$ with

$$comb\{\!|\mathtt{X}_1(\boldsymbol{d}_1), \mathtt{X}_1(\boldsymbol{d}_2), \dots, \mathtt{X}_h(\boldsymbol{d}_j), \dots, \mathtt{X}_k(\boldsymbol{d}_m)|\!\}.$$

Theorem 4.8. Let P and P' be the probability distributions defined by G and $G' = decompose1(G, \mathtt{A})$, respectively, on the random variables of G. If *comb* is an expectation, then $P = P'$.

Proof. It suffices to show that for each instantiation I of $Pa(F(\boldsymbol{d}))$ we have

$$P'(A = true \mid I) = P(A = true \mid I). \tag{31}$$

To show (31) we first write

$$\begin{aligned} P'(A = true \mid I) &= \sum_{I'} P'\big(A = true \mid I'\big) P\big(I' \mid I\big) \\ &= \sum_{I'} comb\{\!|i'_{h,j} \mid 1 \leqslant h \leqslant k,\ 1 \leqslant j \leqslant m|\!\} P\big(I' \mid I\big) \end{aligned} \tag{32}$$

where I' varies over all instantiations of the new variables $\mathtt{X}_h(\boldsymbol{d}_j)$, and

$$i'_{h,j} = \begin{cases} 0 & \text{if } I'(\mathtt{X}_h(\boldsymbol{d}_j)) = \mathit{false}, \\ 1 & \text{if } I'(\mathtt{X}_h(\boldsymbol{d}_j)) = \mathit{true}. \end{cases} \tag{33}$$

As the variables $\mathtt{X}_h(\boldsymbol{d}_j)$ are conditionally independent given $Pa(F(\boldsymbol{d}))$, we have that conditional on I they are $B(F_1(\boldsymbol{d}, \boldsymbol{d}_1)[I]) \otimes \cdots \otimes B(F_k(\boldsymbol{d}, \boldsymbol{d}_m)[I])$-distributed. The right hand side of (32) thus is equal to

$$E_{F_1(\boldsymbol{d},\boldsymbol{d}_1)[I],\dots,F_k(\boldsymbol{d},\boldsymbol{d}_m)[I]}\big[comb\{\!|\mathtt{X}_1(\boldsymbol{d}_1), \dots, \mathtt{X}_k(\boldsymbol{d}_m)|\!\}\big].$$

As *comb* is an expectation, this is equal to

$$comb\{\!|F_1(\boldsymbol{d}, \boldsymbol{d}_1)[I], \dots, F_k(\boldsymbol{d}, \boldsymbol{d}_m\}[I]|\!\} = P(\mathtt{A} = true \mid I). \qquad \square$$

To complete the decomposition for node $\mathtt{A}$ we have to insert a suitable computation network for its conditional probability distribution $comb\{\!|\mathtt{X}_1(\boldsymbol{d}_1), \dots, \mathtt{X}_k(\boldsymbol{d}_m)|\!\}$. That this is always possible is the content of the next theorem.

Theorem 4.9. Let $\mathcal{D}$, $\boldsymbol{d}$, G and $\mathtt{A}$ be as in definition 4.7. Assume that in the construction $G' = decompose1(G, \mathtt{A})$ new variables $\mathtt{X}_1(\boldsymbol{d}_1), \dots, \mathtt{X}_k(\boldsymbol{d}_m)$ have been inserted. There exists a network $G'' = decompose2(G')$ with the following properties:

(1) The nodes of G'' are the nodes of G' together with l new nodes $\mathtt{Z}_1, \dots, \mathtt{Z}_l$ where $l \leqslant (km)^6 + km$.

(2) $\mathtt{A}$ and the $\mathtt{Z}_i$ have at most two parents each, these are all among the $\mathtt{X}_h(\boldsymbol{d}_j)$ and the $\mathtt{Z}_i$.

(3) $\mathtt{A}$ and the new $\mathtt{Z}_i$ are labelled with conditional probability tables, so that $P' = P''$, where P' is the distribution defined by G', and P'' is the distribution defined by G'' restricted to the nodes of G'.

Proof. We show that we can insert nodes $\mathtt{Z}_1, \ldots, \mathtt{Z}_l$ as stated in the theorem such that for every instantiation I' of $\mathtt{X}_1(\boldsymbol{d}_1), \ldots, \mathtt{X}_k(\boldsymbol{d}_m)$ we obtain $P''(\mathtt{A} = \mathit{true} \mid I') = P'(\mathtt{A} = \mathit{true} \mid I')$. The following construction is illustrated in figure 9.

Let $i'_{h,j}$ be defined by (33). The value of $P'(\mathtt{A} = \mathit{true} \mid I') = \mathit{comb}\{\!|i'_{h,j} \mid 1 \leqslant h \leqslant k,\ 1 \leqslant j \leqslant m|\!\}$ only depends on the number of entries $i'_{h,j}$ equal to 1. Denote the event that this number is t by $\#I' = t$. We introduce $km + 1$ new variables $\mathtt{Z}_0, \ldots, \mathtt{Z}_{km}$ that are to have the following property:

$$P''\big(\mathtt{Z}_t = \mathit{true} \mid \#I' = t\big) = \mathit{comb}\{\!|1 : t, 0 : km - t|\!\}, \tag{34}$$

$$P''\big(\mathtt{Z}_t = \mathit{true} \mid \#I' \neq t\big) = 0. \tag{35}$$

Then the probability given I' that at least (and also: at most) one of the nodes $\mathtt{Z}_t$ is 1 is equal to $P'(\mathtt{A} = \mathit{true} \mid I')$. Connecting $\mathtt{A}$ to the variables $\mathtt{Z}_t$ by a decomposed deterministic *or* therefore yields $P''(\mathtt{A} = 1 \mid I) = P'(\mathtt{A} = 1 \mid I)$. It thus remains to show that we can obtain (34) and (35) for the $\mathtt{Z}_t$. For this we appeal to results on the formula complexity of symmetric Boolean functions [3,18] which imply that the functions

$$S^t_{km} : \{0,1\}^{km} \to \{0,1\},$$
$$(x_1, \ldots, x_{km}) \mapsto \begin{cases} 1 & \text{if } \sum_{j=1}^{km} x_j = t, \\ 0 & \text{if } \sum_{j=1}^{km} x_j \neq t \end{cases}$$

can be represented by a Boolean circuit with at most $(km)^5$ *and, or* and *not* gates. This circuit can be used as a subnetwork with input nodes $\mathtt{X}_1(\boldsymbol{d}_1), \ldots, \mathtt{X}_k(\boldsymbol{d}_m)$, output node $\mathtt{Z}_t$ and internal nodes $\mathtt{Z}_{t,g}$ $(1 \leqslant g \leqslant (km)^5)$. The conditional probability tables of the $\mathtt{Z}_{t,g}$ are deterministic *and*, *or* and *not*, as determined by the Boolean circuit. The logical function of the output node $\mathtt{Z}_t$ is modified by changing probability values 1 to $\mathit{comb}\{\!|1 : t, 0 : mk{-}t|\!\}$ (leaving 0-entries unchanged). Then (34) and (35) hold for the $\mathtt{Z}_t$.

To define the $\mathtt{Z}_t$ we have added km subnetworks representing Boolean circuits of size at most $(km)^5$, i.e., a total of at most $(km)^6$ new nodes. Another km nodes are added by the final *or* over the $\mathtt{Z}_t$. All nodes between $\mathtt{A}$ and the $\mathtt{X}_h(\boldsymbol{d}_j)$ have two parents at most. □

For natural combination functions one can usually obtain computation networks that are smaller and simpler than those obtained from the general construction of the proof of theorem 4.9. In the case of *noisy-or* a straightforward generalization of the construction shown in figure 7 yields computation networks with only $2n - 1$ new nodes when applied to a node with n parents. Moreover, the inserted network here has a tree

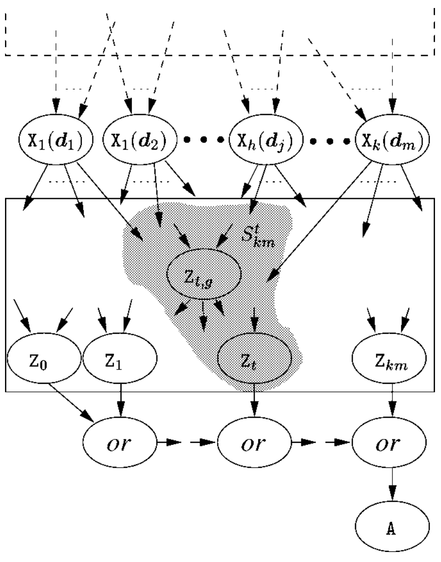


Figure 9. The second decomposition step.

structure and therefore does not introduce any new cycles into the network. We now show that an equally small and structurally simple computation network exists for *mean*.

For this let A be a node in a network G with parents $\{X_1, \ldots, X_n\}$, labelled with $mean\{|X_1, \ldots, X_n|\}$. We define $decompose2(G, A)$ by introducing new variables $Y_1, \ldots, Y_n$ connected to $X_1, \ldots, X_n$ and A as shown in figure 10. The conditional probability tables for Y_1, Y_h $(1 \leqslant h \leqslant n)$ and A are:

X_1	$P(Y_1 = true)$
false	0
true	1

Y_{h-1}	X_h	$P(Y_h = true)$
false	*false*	0
false	*true*	$1/h$
true	*false*	$(h-1)/h$
true	*true*	1

Y_n	$P(A = true)$
false	0
true	1

To show that the resulting network is a correct computation network for *mean*, we have to check that for an instantiation I of $X_1, \ldots, X_n$ that instantiates exactly k of the X_i to *true* we obtain

$$P(A = true \mid I) = \frac{k}{n} \quad (0 \leqslant k \leqslant n). \tag{36}$$

We show (36) by induction on the number n of input nodes. For $n = 1$ the instantiation of A is just a deterministic copy of the instantiation of X_1, so that (36) holds. Now assume that (36) is satisfied for $n-1$ inputs. Let I be an instantiation of $X_1, \ldots, X_n$, and I' its restriction to $X_1, \ldots, X_{n-1}$. Let k and k' be the number of nodes instantiated to *true* by

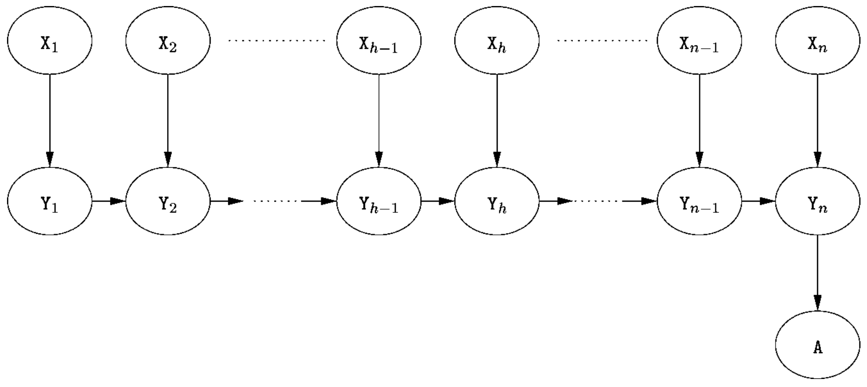


Figure 10. Computation network for *mean*.

I and I', respectively ($k' \in \{k-1, k\}$). By induction hypothesis then

$$P\bigl(\mathtt{Y}_{n-1} = \mathit{true} \mid I'\bigr) = \frac{k'}{n-1}. \tag{37}$$

Now (36) can be verified by straightforward computations, separately for the two cases $k' = k-1$ and $k' = k$. In the case $k' = k-1$, i.e., $I(\mathtt{X}_n) = \mathit{true}$ we obtain

$$\begin{aligned}P(\mathtt{A} = \mathit{true} \mid I) &= P(\mathtt{Y}_n = \mathit{true} \mid I)\\ &= P(\mathtt{Y}_n = \mathit{true} \mid \mathtt{Y}_{n-1} = \mathit{true}, \mathtt{X}_n = \mathit{true})P\bigl(\mathtt{Y}_{n-1} = \mathit{true} \mid I'\bigr)\\ &\quad + P(\mathtt{Y}_n = \mathit{true} \mid \mathtt{Y}_{n-1} = \mathit{false}, \mathtt{X}_n = \mathit{true})P\bigl(\mathtt{Y}_{n-1} = \mathit{false} \mid I'\bigr)\\ &= \frac{k-1}{n-1} + \frac{1}{n}\left(1 - \frac{k-1}{n-1}\right) = \frac{k}{n}.\end{aligned}$$

The case $k' = k$ is similar.

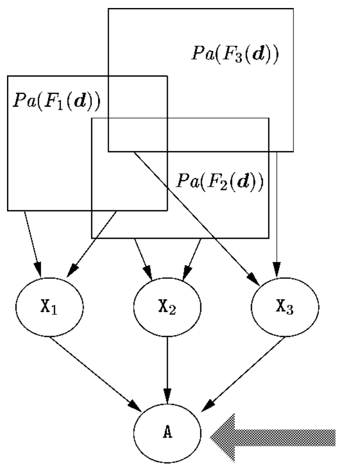
Thus, just as for *noisy-or*, we obtain a tree structured computation network that only introduces n new nodes. Putting the two decomposition steps together, we now define:

Definition 4.10. Let $\mathcal{D}, \boldsymbol{d}, G$ and $\mathtt{A}$ be as in definition 4.7, with *noisy-or* or *mean* the combination function of $F(\boldsymbol{d})$. Then

$$\mathit{decompose}(G, \mathtt{A}) := \mathit{decompose2}\bigl(\mathit{decompose1}(G, \mathtt{A}), \mathtt{A}\bigr).$$

The definition of $\mathit{decompose}(G, \mathtt{A})$ may be extended to other multilinear combination functions for which an effective construction of computation networks has been defined.

The definition of $\mathit{decompose}(G, \mathtt{A})$ for nodes $\mathtt{A}$ labelled with *noisy-or* or *mean* combination functions is the cornerstone of the transformation from the network $G^{\Phi}_{\mathcal{D}}$ to the network $DG^{\Phi}_{\mathcal{D}}$. To fully define the transformation, we also have to explain how to decompose nodes labelled with convex combinations.


$Pa(F_3(\boldsymbol{d}))$
$Pa(F_1(\boldsymbol{d}))$
$Pa(F_2(\boldsymbol{d}))$
X1
X2
X3
A

$\mathtt{X}_1$	$\mathtt{X}_2$	$\mathtt{X}_3$	$P(\mathtt{A} = true)$
false	*false*	*false*	0
false	*false*	*true*	1
false	*true*	*false*	0
false	*true*	*true*	1
true	*false*	*false*	0
true	*false*	*true*	0
true	*true*	*false*	1
true	*true*	*true*	1

Figure 11. Decomposition of a convex combination.

Definition 4.11. Let $\mathcal{D}$, $\boldsymbol{d}$, G, $\mathtt{A}$ be as in definition 4.7, with

$$F(\boldsymbol{d}) \equiv F_1(\boldsymbol{d})F_2(\boldsymbol{d}) + \bigl(1 - F_1(\boldsymbol{d})\bigr)F_3(\boldsymbol{d}).$$

We define a new network $G' = decompose(G, \mathtt{A})$ as follows (cf. figure 11):

(1) Let $\mathtt{X}_1$, $\mathtt{X}_2$, $\mathtt{X}_3$ be new Boolean random variables.

(2) Create a new graph by removing all edges between $Pa(F(\boldsymbol{d}))$ and $\mathtt{A}$, inserting an edge from each node in $Pa(F_i(\boldsymbol{d}))$ to $\mathtt{X}_i$, and inserting an edge from each node $\mathtt{X}_i$, to $\mathtt{A}$.

(3) Label each node $\mathtt{X}_i$ with $F_i(\boldsymbol{d})$, and $\mathtt{A}$ with the deterministic table shown in figure 11.

As a convex combination also is multilinear in its three probability formula arguments, the same reasoning as for multilinear combination function shows that the decomposition of definition 4.11 preserves the probability distribution defined by the network.

The decomposition given in definition 4.11 is very similar to a decomposition given by Boutilier et al. [4] for the utilization of context specific independence in probabilistic inference. The similarity between the two decompositions is no coincidence, as the one in [4] was designed to be particularly useful when conditional probability tables are represented by (decision-) trees, and nested convex combinations are the probability formula equivalent of a tree representation.

Calling a probability formula *basic* if it is either a constant or an indicator function, and *complex* else, we can now define $DG^{\Phi}_{\mathcal{D}}$ via the algorithm *decompose-network* given in table 1.

It is readily verified that the algorithm terminates, and that the structure of the output network does not depend on the selection procedure for the node $\mathtt{A}$ to be decomposed next. It also is clear that the probability distribution defined on the atoms of $\mathrm{Mod}_D(R)$

Table 1
Algorithm *decompose-network*.

```
Input: Network $G^{\Phi}_{\mathcal{D}}$ defined by RRBN $\Phi$ that uses noisy-or and mean as the only combination functions
Initialize: $G := G^{\Phi}_{\mathcal{D}}$;
while $G$ contains node $\mathtt{A}$ labelled with complex probability formula $F(\boldsymbol{d})$
   $G := decompose(G, \mathtt{A})$
end while;
$DG^{\Phi}_{\mathcal{D}} := G$
```

is the same in $DG^{\Phi}_{\mathcal{D}}$ as in $G^{\Phi}_{\mathcal{D}}$, and that each node in $DG^{\Phi}_{\mathcal{D}}$ has at most three parents. It remains to show that the size of $DG^{\Phi}_{\mathcal{D}}$ is polynomial in the size of D.

Theorem 4.12. Let Φ be an R, S-RRBN defined by probability formulas that contain *noisy-or* and *mean* as the only combination functions. Let $\mathcal{D}$ be a S-structure with $|D| = n$. The number of nodes in $DG^{\Phi}_{\mathcal{D}}$ as a function of n is $\mathrm{O}(n^{k+q})$, where k is the maximum arity of relation symbols in R, and q is the maximal quantifier depth of probability formulas in Φ.

The proof is almost immediate by noting that $G^{\Phi}_{\mathcal{D}}$ contains at most n^k nodes, and that the iterative decomposition of a node $\mathtt{A}$ in $G^{\Phi}_{\mathcal{D}}$ labelled with $F(\boldsymbol{d})$ generates $\mathrm{O}(n^q)$ new nodes.

We now have shown that for an interesting class of RRBNs we can construct for inference standard Bayesian networks of size polynomial in the size of the underlying domain. This, of course, does not guarantee that inference will be polynomial in the domain size (and in view of the results mentioned in section 1.3 it is very unlikely that such a guarantee could be given for any inference technique). For the very simple example 4.2 the result of the decomposition always is a singly connected network, so that here the complexity of inference is actually linear in the domain size. An interesting question for future work is to investigate under what conditions the good behavior of this example generalizes, i.e., under what conditions will the $DG^{\Phi}_{\mathcal{D}}$ have graph theoretic properties that guarantee probabilistic inference algorithms to run in time polynomial in $|D|$?

5. Context

The network $DG^{\Phi}_{\mathcal{D}}$ constructed in the previous section is a complete representation of the distribution $P^{\Phi}_{\mathcal{D}}$; it can be used to compute the answer to any query (22). For a particular query, $DG^{\Phi}_{\mathcal{D}}$ usually contains a lot of information which is not relevant for the computation of the answer, and the query can also be answered by computations in a small subnetwork of $DG^{\Phi}_{\mathcal{D}}$. This observation is true not only for networks of the type $DG^{\Phi}_{\mathcal{D}}$, but for any (large) standard Bayesian network, and has led to the development of *pruning* techniques [2] that in a preprocessing step generate a subnetwork which is sufficient for the computation of a given query. In knowledge based model construction (cf. section 1.3) the simplifications possible through pruning usually are taken into account already during the construction of the standard Bayesian network, which works by an incremental construction starting from the nodes $\mathtt{r}_i(\boldsymbol{d}_j)$ appearing in the query [5,20]. Analogous query dependent incremental construction techniques can also be used in the RRBN framework to generate only a relevant subnetwork of $G^{\Phi}_{\mathcal{D}}$, and hence of $DG^{\Phi}_{\mathcal{D}}$.

Example 5.1. Let S contain a constant *LA* and two unary relations C and P, denoting objects of sort "city" and "person", respectively. A RRBN for the random relations $R = \{\mathtt{burglary}, \mathtt{alarm}, \mathtt{lives\_in}, \mathtt{earthquake}\}$ then is given by

$$\begin{aligned}
F_{\mathtt{lives\_in}}(p,c) &\equiv 0.05(c = LA) + 0.001(\neg c = LA),\\
F_{\mathtt{burglary}}(p) &\equiv 0.01,\\
F_{\mathtt{earthquake}}(c) &\equiv 0.01(c = LA),\\
F_{\mathtt{alarm}}(p) &\equiv \mathtt{lives\_in}(p, LA)\textit{n-o}\{\!|0.9\mathtt{burglary}(p),\\
&\qquad\qquad 0.2\mathtt{earthquake}(LA) \mid \emptyset; \tau |\!\}\\
&\quad + (\neg\mathtt{lives\_in}(p, LA))0.9\mathtt{burglary}(p).
\end{aligned}$$

These formulas use some of the simplifications introduced in section 3.1, especially implicit sort constraints by the naming of variables. The formula $F_{\mathtt{lives\_in}}(p,c)$ simply (and somewhat unrealistically) says that person p lives in city c with probability 0.05 if $c = LA$, and with probability 0.001 else. A more sensible formula would have to make `lives_in` a functional relation using the constructs discussed in section 3.1.

Let the given domain contain a person *Holmes*, and assume that the query is

$$\begin{aligned}
&P\big(\mathtt{burglary}(Holmes) = true \mid\\
&\quad \mathtt{lives\_in}(Holmes, LA) = false, \mathtt{alarm}(Holmes) = true\big) = ?
\end{aligned}$$

Then the relevant subnetwork of $G^{\Phi}_{\mathcal{D}}$ obtained by an incremental construction starting with the nodes in the query is shown in figure 12.

The subnetwork of $G^{\Phi}_{\mathcal{D}}$ constructed in this example is not as small as it might be: the node $\mathtt{earthquake}(LA)$ and its link to $\mathtt{alarm}(Holmes)$ is relevant only when $\mathtt{lives\_in}(Holmes, LA)$ is true. This illustrates a basic limitation of network reduction by pruning (and the corresponding incremental construction methods): with these techniques we cannot take into account the values to which ground atoms are instantiated in the evidence – only the fact which atoms are instantiated has an influence on the con-

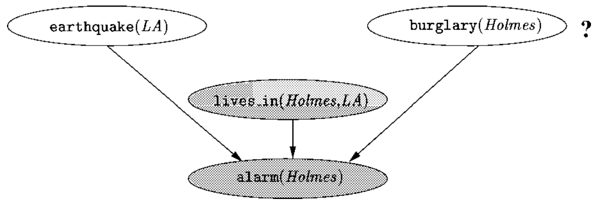


Figure 12. Network reduction with value-insensitive pruning.

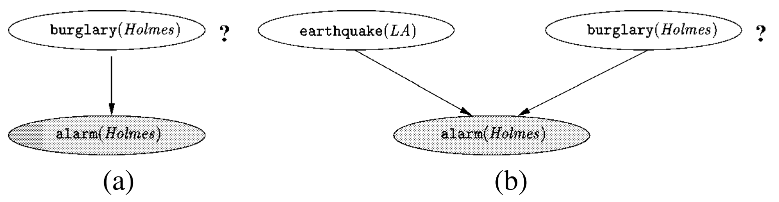


Figure 13. Value-sensitive network reduction.

structed subnetwork. A much better use of the given evidence, however, would be to only construct the network in figure 13(a) when it is given that lives_in(*Holmes*, *LA*) is false, and the network in figure 13(b) when it is given that lives_in(*Holmes*, *LA*) is true.

Techniques that try to utilize given instantiations of random variables to particular values for simplification of model representations and inference have become associated with the word *context*. In the *context sensitive knowledge bases* of Ngo and Haddawy [20] probabilistic rules (1) are annotated with constraints that are expressed by a designated set of *context relations*. Our example (which is taken almost directly from [20]), would be encoded in a context sensitive knowledge base by making lives_in a designated context relation, and by entering two separate rules for alarm into the knowledge base – one for the case of lives_in(·, *LA*) being false, and one for it being true. Disadvantage of this approach is that it introduces an often not very natural distinction between context relations and probabilistic relations, and that it requires that all atoms over the context relations are assigned truth values before probabilistic inference can begin.

Boutilier et al. [4] define *context specific independence* of random variables, a concept that initially only leads to more compact representations of conditional probability tables. By decomposing nodes according to a similar rule as given in our definition 4.11, however, this compact representation can also be used to enable value-sensitive network reductions.

In the following we will show how by a simple extension of the *decompose-network* algorithm of section 4, a construction algorithm for standard Bayesian networks is obtained, which, in the context of given evidence, generates smaller networks in a manner that is sensitive to the values of instantiated atoms. In particular, in the case of our introductory example, only the small networks in figure 13 will be constructed. Our approach relies on the following standard network simplification, which also is the basis for cutset conditioning algorithms [22,26].

Definition 5.2. Let N be a standard Bayesian network with nodes labelled with standard conditional probability tables. Let I be an instantiation of a subset $\mathtt{E}_1, \ldots, \mathtt{E}_k$ of nodes. The network N' obtained by *conditioning N on I* is defined by deleting all outgoing edges from instantiated nodes and by conditioning all conditional probability tables on I. A conditional probability table for a node $\mathtt{X}$ in N with parents $Pa(\mathtt{X})$ is conditioned on I by making it a table that depends on $Pa(\mathtt{X}) \setminus \{\mathtt{E}_1, \ldots, \mathtt{E}_k\}$ only, and assigning to every instantiation I^* of these nodes the original table entry for the instantiation $I^* \cup I$.

The conditional distribution given I is the same in N and N', so that N' can be used for the computation of conditional probabilities given I.

When a conditional probability distribution is represented by a probability formula, rather than a table, it sometimes can be conditioned on evidence simply by substituting 0 and 1 for the instantiated ground atoms. The probability formula at the node alarm(*Holmes*) in the network of figure 12, for instance, is

$$\begin{aligned}&\texttt{lives\_in}(\mathit{Holmes}, \mathit{LA})\,\textit{n-o}\{\!|0.9\texttt{burglary}(\mathit{Holmes}), 0.2\texttt{earthquake}(\mathit{LA}) \mid \emptyset; \tau|\!\}\\ &\quad + \big(\neg\texttt{lives\_in}(\mathit{Holmes}, \mathit{LA})\big)0.9\texttt{burglary}(\mathit{Holmes}). \qquad (38)\end{aligned}$$

It can be conditioned on lives_in(*Holmes*, *LA*) $=$ *false* by setting the indicator lives_in(*Holmes*, *LA*) in (38) to 0, which leads to the much simpler formula

$$0.9\texttt{burglary}(\mathit{Holmes}). \qquad (39)$$

Thus, conditioning the probability formula at alarm(*Holmes*) on the evidence not only removes the link from lives_in(*Holmes*, *LA*) to alarm(*Holmes*), but we also obtain a representation of the new conditional distribution at alarm(*Holmes*) that shows that this distribution no longer depends on earthquake(*LA*).

The probability formula (38) can be directly conditioned on instantiations of lives_in(*Holmes*, *LA*), because its dependency on this ground atom is given explicitly through occurrences of (ground) indicator variables lives_in(*Holmes*, *LA*) in the formula, which can be replaced with 0 or 1. In general, however, the dependence of a probability formula $F(\boldsymbol{d})$ on some ground atom $\texttt{r}(\boldsymbol{d})$ can also be implicit, in that $\texttt{r}(\boldsymbol{d}) \in \mathit{Pa}(F(\boldsymbol{d}))$, but $F(\boldsymbol{d})$ does not contain the indicator $\texttt{r}(\boldsymbol{d})$. An example is the probability formula

$$F_{\texttt{r}}(d_1) \equiv \textit{n-o}\{\!|0.4\texttt{s}(d_1, w) \mid w; w \neq d_1|\!\}, \qquad (40)$$

at the node $\texttt{r}(d_1)$ of the network in figure 6. This formula depends on $\texttt{s}(d_1, d_4)$, for instance, but it cannot be conditioned on an instantiation $\texttt{s}(d_1, d_4) = \alpha$ simply by substituting 0 or 1. This problem has disappeared in the decomposed version of the network shown in figure 7. All probability formulas in this network depend explicitly on their parent nodes. This is true in general: successive applications of the *decompose* operator eliminate implicit dependencies on parent nodes.

A general strategy for the construction of decomposed networks that are simplified in the most effective way by conditioning on given instantiations of ground atoms is to substitute values 0 or 1 for ground indicator variables as soon as they appear explicitly in the iterative construction of $DG^{\Phi}_{\mathcal{D}}$. This substitution is formally defined as follows.

Definition 5.3. Let $F(\boldsymbol{v})$ be an R, S-probability formula, $\boldsymbol{d}$ a tuple of domain elements, I an instantiation of some ground R-atoms. We define the probability formula $F(\boldsymbol{d})_{|I}$ obtained by *conditioning* $F(\boldsymbol{d})$ *on* I as follows:

(i) $F(\boldsymbol{d}) = q$, then $F(\boldsymbol{d})_{|I} \equiv q$.

(ii) If $F(\boldsymbol{d}) = \mathtt{r}(\boldsymbol{d})$ then $F(\boldsymbol{d})_{|I} \equiv 0$ if $I(\mathtt{r}(\boldsymbol{d})) = \mathit{false}$, $F(\boldsymbol{d})_{|I} \equiv 1$ if $I(\mathtt{r}(\boldsymbol{d})) = \mathit{true}$, and $F(\boldsymbol{d})_{|I} \equiv \mathtt{r}(\boldsymbol{d})$ if $\mathtt{r}(\boldsymbol{d})$ is not instantiated by I.

(iii) If $F(\boldsymbol{d}) \equiv F_1(\boldsymbol{d})F_2(\boldsymbol{d}) + (1 - F_1(\boldsymbol{d}))F_3(\boldsymbol{d})$, then

$$F(\boldsymbol{d})_{|I} \equiv F_1(\boldsymbol{d})_{|I}F_2(\boldsymbol{d})_{|I} + (1 - F_1(\boldsymbol{d})_{|I})F_3(\boldsymbol{d})_{|I}$$

(when one of the $F_i(\boldsymbol{d})_{|I}$ is 0 or 1, then this expression is simplified according to the rules $0F \rightsquigarrow 0$, $F + 0 \rightsquigarrow F$).

(iv) If $F(\boldsymbol{d}) \equiv \mathit{comb}\{\!|F_1(\boldsymbol{d}, \boldsymbol{w}), \ldots, F_k(\boldsymbol{d}, \boldsymbol{w}) \mid \boldsymbol{w}; c(\boldsymbol{d}, \boldsymbol{w})|\!\}$, then

$$F(\boldsymbol{d})_{|I} \equiv \mathit{comb}\{\!|F_1(\boldsymbol{d}, \boldsymbol{w})_{|I}, \ldots, F_k(\boldsymbol{d}, \boldsymbol{w})_{|I} \mid \boldsymbol{w}; c(\boldsymbol{d}, \boldsymbol{w})|\!\}.$$

We can now integrate the conditioning operation of definition 5.3 into the *decompose-network* algorithm of table 1 by preceding the decomposition step $G :=$ *decompose*$(G, \mathtt{A})$ in the while-loop with the command

$$\textit{replace } F(\boldsymbol{d}) \textit{ with } F(\boldsymbol{d})_{|I}.$$

The final result returned by this modified algorithm we denote with $DG^{\Phi}_{\mathcal{D}|I}$. We record in a theorem:

Theorem 5.4. Let I be an instantiation of some atoms of $\mathrm{Mod}_D(R)$. Let P, P' be the probability distributions defined by $DG^{\Phi}_{\mathcal{D}}$ and $DG^{\Phi}_{\mathcal{D}|I}$, respectively, on the atoms of $\mathrm{Mod}_D(R)$. Then $P(\cdot \mid I) = P'(\cdot \mid I)$.

The proof follows immediately from the fact that in each execution of the while-loop in the amended decomposition algorithm neither command changes the conditional distribution given I in the current network G.

Note that it is really essential to condition probability formulas on the evidence as soon as instantiated atoms appear explicitly: in our introductory example 5.1 for instance, conditioning (38) on `lives_in`(*Holmes*, *LA*) = *false* before decomposing the node `alarm`(*Holmes*) leads to the simplified formula (39), and subsequent decomposition to the small network of figure 13(a). Decomposing `alarm`(*Holmes*) first, and conditioning on `lives_in`(*Holmes*, *LA*) = *false* later, on the other hand, will no longer lead to the elimination of the node `earthquake`(*LA*).

6. Conclusion

In recent years several frameworks for probabilistic modeling and inference have been proposed that aim to integrate first-order logic representation constructs into the Bayesian network paradigm. Probabilistic relational models can be regarded as a well-defined common semantic core of these systems.

Probability formulas and recursive Relational Bayesian networks are one framework for probabilistic relational model representation. In this paper we have shown how RRBNs can be used to represent complex prms, and how the structure of probability formulas can be exploited to construct standard Bayesian networks that represent just the part of a probabilistic model that is relevant for a particular query.

Key design goal of RRBNs was to base the representation framework on a small number of elementary constructs. This has been achieved mostly by reducing the representation language to the four simple construction rules for probability formulas. When, furthermore, the combination function permitted in the construction of probability formulas are limited to a small set of elementary functions, then the whole representation framework of RRBNs consists of a small number of elementary constructs. We have singled out *noisy-or* and *mean* as very useful elementary combination functions, and have seen that they alone allow us to deal with a variety of interesting modelling problems.

The price we pay for the simplicity of the representation framework is a rather rigid language, in which even relatively simple representations become hard to read. This, however, is not a major problem, as one can always introduce high-level constructs that are definable in terms of the given basic ones. Experience tells us that this is the most fruitful approach: given a certain type of semantic construct for which a formal representation language is needed, one should identify small sets of elementary constructs with which the problem can be solved in principle. Once such a set of elementary constructs has been obtained, one can always introduce secondary, defined constructs that can be used to achieve better readability of complex representations. Investigations of basic semantic or algorithmic questions, however, are much easier to conduct when only the few underlying elementary constructs have to be taken into account. This is just why the theory of computability is based on simple representation languages for algorithms – Turing machine tables, recursive functions – not directly on high-level programming languages. Mathematical logic, too, has only been so successful because it reduced the rich language of mathematics to the few simple syntax rules of predicate logic. The experiences gained in these foundational mathematical fields should not be ignored in the lesser enterprise of studying the mathematics of probabilistic relational models.

References

[1] F. Bacchus, *Representing and Reasoning with Probabilistic Knowledge* (MIT Press, Cambridge, MA, 1990).

[2] M. Baker and T.E. Boult, Pruning Bayesian networks for efficient computation, in: *Uncertainty in Artificial Intelligence*, Vol. 6, eds. P.P. Bonissone, M. Henrion, L.N. Kanal and J.F. Lemmer (Elsevier Science, Amsterdam, 1991) pp. 225–232.

[3] R.B. Boppana and M. Sipser, The complexity of finite functions, in: *Handbook of Theoretical Computer Science* (Elsevier Science, Amsterdam, 1990).

[4] C. Boutilier, N. Friedman, M. Goldszmidt and D. Koller, Context-specific independence in Bayesian networks, in: *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, Portland, Oregon (1996) pp. 115–123.

[5] J.S. Breese, Construction of belief and decision networks, Computational Intelligence 8(4) (1992) 624–647.

[6] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artificial Intelligence 42 (1990) 393–405.

[7] P. Dagum, A. Galper and E. Horvitz, Dynamic network models for forecasting, in: *Proceedings of the 8th Annual Conference on Uncertainty in Artificial Intelligence (UAI-92)* (Morgan Kaufmann, San Francisco, CA, 1992) pp. 41–48.

[8] N. Friedman, L. Getoor, D. Koller and A. Pfeffer, Learning probabilistic relational models, in: *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)* (1999).

[9] A. Frieze and C. McDiarmid, Algorithmic theory of random graphs, Random Structures & Algorithms 10 (1997) 5–42.

[10] F.V. Jensen, *An Introduction to Bayesian Networks* (UCL Press, 1996).

[11] P. Haddawy, Generating Bayesian networks from probability logic knowledge bases, in: *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)* (1994) pp. 262–269.

[12] J.Y. Halpern, An analysis of first-order logics of probability, Artificial Intelligence 46 (1990) 311–350.

[13] T. Huang, D. Koller, J. Malik, G. Ogasawara, B. Rao, S. Russell and J. Weber, Automatic symbolic traffic scene analysis using belief networks, in: *Proceedings of the 12th National Conference on Artificial Intelligence* (1994) pp. 966–972.

[14] M. Jaeger, Relational Bayesian networks, in: *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)* (1997).

[15] M. Jaeger, Convergence results for relational Bayesian networks, in: *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS-98)* (1998).

[16] M. Jaeger, Reasoning about infinite random structures with relational Bayesian networks, in: *Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR'98)* (1998) pp. 570–581.

[17] M. Jaeger, On the complexity of inference about probabilistic relational models, Artificial Intelligence 117 (2000) 297–308.

[18] V.M. Khrapchenko, The complexity of the realization of symmetrical functions by formulae, Mathematical Notes of the Academy of Sciences of the USSR 11(1) (1972) 70–76.

[19] D. Koller and A. Pfeffer, Probabilistic frame-based systems, in: *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)* (1998) pp. 580–587.

[20] L. Ngo and P. Haddawy, Answering queries from context-sensitive probabilistic knowledge bases, Theoretical Computer Science 171 (1997) 147–177.

[21] A.E. Nicholson and J.M. Brady, Dynamic belief networks for discrete monitoring, IEEE Systems, Man and Cybernetics 24(11) (1994) 1593–1610.

[22] J. Pearl, Fusion, propagation and structuring in belief networks, Artificial Intelligence 29 (1986) 241–288.

[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, The Morgan Kaufmann Series in Representation and Reasoning (Morgan Kaufmann, San Mateo, CA, 2nd revised edition, 1988).

[24] D. Poole, Probabilistic horn abduction and Bayesian networks, Artificial Intelligence 64 (1993) 81–129.

[25] A. Saffiotti and E. Umkehrer, Inference-driven construction of valuation systems from first-order clauses, IEEE Transactions on Systems, Man, and Cybernetics 24(11) (1994) 1611–1624.

[26] H.J. Suermondt and G.F. Cooper, Probabilistic inference in multiply connected belief networks using loop cutsets, International Journal of Approximate Reasoning 4 (1990) 283–306.

[27] M.P. Wellman, J.S. Breese and R.P. Goldman, From knowledge bases to decision models, The Knowledge Engineering Review 7(1) (1992) 35–53.