entropy

MDPI

Article

On the Coherence of Probabilistic Relational Formalisms

Glauber De Bona and Fabio G. Cozman *

Escola Politécnica, Universidade de São Paulo, São Paulo 05508-010, Brazil; glauber.bona@usp.br
* Correspondence: fgcozman@usp.br

Received: 22 February 2018; Accepted: 24 March 2018; Published: 27 March 2018

Abstract: There are several formalisms that enhance Bayesian networks by including relations amongst individuals as modeling primitives. For instance, Probabilistic Relational Models (PRMs) use diagrams and relational databases to represent repetitive Bayesian networks, while Relational Bayesian Networks (RBNs) employ first-order probability formulas with the same purpose. We examine the *coherence checking* problem for those formalisms; that is, the problem of guaranteeing that any grounding of a well-formed set of sentences does produce a valid Bayesian network. This is a novel version of de Finetti's problem of coherence checking for probabilistic assessments. We show how to reduce the coherence checking problem in relational Bayesian networks to a validity problem in first-order logic augmented with a transitive closure operator and how to combine this logic-based approach with faster, but incomplete algorithms.

Keywords: relational Bayesian networks; probabilistic relational models; coherence checking

1. Introduction

Most statistical models are couched so as to guarantee that they specify a single probability measure. For instance, suppose we have N independent biased coins, so that heads has probability p for each one of them. Then, the probability of a particular configuration of all coins is exactly $p^n(1-p)^{N-n}$, where n is the number of heads in the configuration. Using de Finetti's terminology, we can say that the probabilistic assessments and independence assumptions are *coherent* as they are satisfied by a probability distribution [1]. The study of coherence and its consequences has influenced the foundations of probability and statistics, serving as a subjectivist basis for probability theory [2,3], as a broad prescription for statistical practice [4,5] and generally as a bedrock for decision-making and inference [6–8].

In this paper, we examine the coherence checking problem for probabilistic models that enhance Bayesian networks with relations and first-order formulas: more precisely, we introduce techniques that allow one to check whether a given relational Bayesian network, or a given probabilistic relational model is guaranteed to specify a probability distribution. Note that "standard" Bayesian networks are, given some intuitive assumptions, guaranteed to be coherent [9–11]. The challenge here is to handle models that enlarge Bayesian networks with significant elements of first-order logic; we do so by resorting to logical inference itself as much as possible. In the remainder of this section, we explain the motivation for this study and the basic terminology concerning it, and at the end of this section, we state our goals and our approach in more detail.

To recap, a Bayesian network consists of a directed acyclic graph, where each node is a random variable, and a joint probability distribution over those variables, such that the distribution and the graph satisfy a Markov condition: each random variable is independent of its non-descendants given its parents. (In a directed acyclic graph, node X is a *parent* of node Y if there is an edge from X to Y. The set of parents of node X is denoted $\mathtt{Pa}(X)$. Similarly, we define the children of a node, the descendants of a node, and so on.)

If all random variables $X_1, \ldots, X_n$ in a Bayesian network are categorical, then the Markov condition implies a factorization:

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i | \mathtt{Pa}(X_i) = \pi_i), \tag{1}$$

where π_i is the projection of $\{X_1 = x_1, \ldots, X_n = x_n\}$ on $\mathtt{Pa}(X_i)$.

Typically, one specifies a Bayesian network by writing down the random variables $X_1, \ldots, X_n$, drawing the directed acyclic graph, and then settling on probability values $P(X_i = x_i | \mathtt{Pa}(X_i) = \pi_i)$, for each X_i, each x_i and each π_i. By following this methodological guideline, one obtains the promised coherence: a unique joint distribution is given by Expression (1).

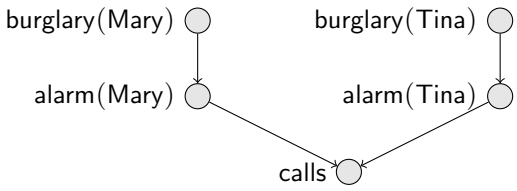
The following example introduces some useful notation and terminology.

Example 1. *Consider two neighbors,* Mary *and* Tina. *The probability that a house is burglarized is* 0.001 *in their town. The alarm of a house rings with probability* 0.9 *given that the house is burglarized and with probability* 0.01 *given the house is not burglarized. Finally, if either alarm rings, the police are called. This little story, completed with some assumptions of independence, is conveyed by the Bayesian network in Figure 1, where* $\mathsf{burglary}(x)$ *means that the house of* x *(either* Mary *or* Tina*) is burglarized; similarly* $\mathsf{alarm}(x)$ *means that the alarm of* x*'s house rings; and finally* calls *just means that the police are called by someone.*

*In this paper, every random variable is binary with values zero and one, the former meaning "*false*" and the latter meaning "*true*". Furthermore, we often write* $P(X)$*, where* X *is a random variable, to mean the event* $\{X = 1\}$*, and we often write* $P(\neg X)$ *to mean the event* $\{X = 0\}$.

Note also that we use, whenever appropriate, logical expressions with random variable names, such as $\mathsf{alarm}(\mathsf{Mary}) \vee \mathsf{burglary}(\mathsf{Tina})$ *to mean the disjunction of the proposition stating that* $\mathsf{alarm}(\mathsf{Mary})$ *is* true *and the proposition that* $\mathsf{burglary}(\mathsf{Tina})$ *is* true*. A random variable name has a dual use as a proposition name.*

From the Bayesian network in Figure 1, we compute $P(\mathsf{alarm}(\mathsf{Mary})) = 0.9 \times 0.001 + 0.01 \times 0.999 = 0.01899$ *and* $P(\mathsf{calls}) = 0.01899 + 0.01899 - (0.01899)^2$.



$P(\mathsf{burglary}(x)) = 0.001$

$P(\mathsf{alarm}(x)|\mathsf{burglary}(x)) = 0.9$

$P(\mathsf{alarm}(x)|\neg\mathsf{burglary}(x)) = 0.01$

$\mathsf{calls} = \mathsf{alarm}(\mathsf{Mary}) \vee \mathsf{alarm}(\mathsf{Tina})$

Figure 1. Bayesian network modeling the burglary-alarm-call scenario with Mary and Tina. In the probabilistic assessments (right), the logical variable x stands for Mary and for Tina.

Here are some interesting scenarios that enhance the previous example:

Example 2. *(Scenario 1) Consider that now we have three people,* Mary, Tina *and* John*, all neighbors. We can easily imagine an enlarged Bayesian network, with two added nodes related to* John*, and a modified definition where* $\mathsf{calls} = \mathsf{alarm}(\mathsf{Mary}) \vee \mathsf{alarm}(\mathsf{Tina}) \vee \mathsf{alarm}(\mathsf{John})$.

(Scenario 2) It is also straightforward to expand our Bayesian network to accommodate n *individuals* $a_1, a_2, \ldots, a_n$*, all neighbors. We may even be interested in reasoning about* calls *without any commitments to a fixed* n*, where* calls *is a disjunction over all instances of* $\mathsf{alarm}(x)$*. For instance, we have that* $P(\neg\mathsf{calls}) = (1 - 0.01899)^n$*; hence, the probability of a call to the police will be larger than half for a city with more than 36 inhabitants. No single Bayesian network allows this sort of "aggregate" inference.*

(Scenario 3) Consider a slightly different situation with three people, where: Mary *and* Tina *are neighbors* Tina *and* John *are neighbors, but* Mary *and* John *are not neighbors. Suppose also that each person may call*

the police, depending on neighboring alarms. This new situation is codified into the Bayesian network given in Figure 2.

(Scenario 4) Suppose we want to extend Scenario 3 to a town with n people. Without knowing which pairs are neighbors, there is no way we can predict in advance the structure of the resulting Bayesian network. However, we can reason about the possible networks: for instance, we know that each set of n people produces a valid Bayesian network, without any cycles amongst random variables.

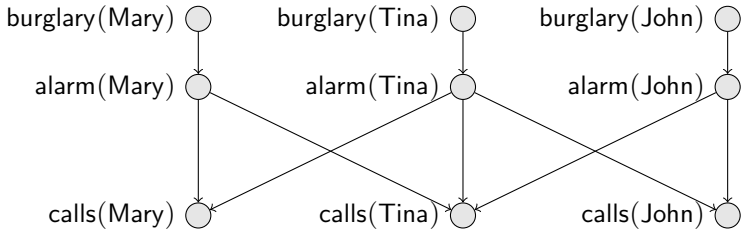


Figure 2. Bayesian network modeling Scenario 3 in Example 2. Probabilistic assessments are just as in Figure 1, except that, for each x, calls(x) is the disjunction of its corresponding parents.

There are many other scenarios where probabilistic modeling must handle repetitive patterns such as the ones described in the previous examples, for instance in social network analysis or in processing data in the semantic web [12–14]. The need to handle such "very structured" scenarios has led to varied formalisms that extend Bayesian networks with the help of predicates and quantifiers, relational databases, loops and even recursion [15]. Thus, instead of dealing with a random variable X at a time, we deal with *parameterized random variables* [16]. We write $X(x)$ to refer to a parameterized random variable that yields a random variable for each fixed x in a domain; if we consider individuals a and b in a domain, we obtain random variables $X(a)$ and $X(b)$.

Plates offer a popular scheme to manipulate parameterized random variables [17]. A plate is a set of parameterized random variables that share a logical variable, meaning that they are indexed by elements of the same domain. A plate is usually drawn as a rectangle (associated with a domain) containing parameterized random variables. Figure 3 shows simple plate models for the burglary-alarm-call scenario described in Scenario 2 of Example 2.

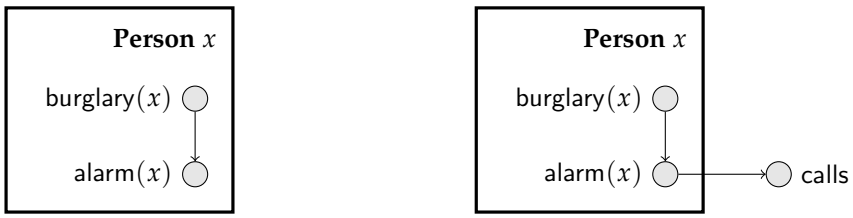


Figure 3. Plate models for Scenario 2 of Example 2; that is, for the burglary-alarm-call scenario where there is a single random variable calls. Left: A partial plate model (without the calls random variable), indicating that parameterized random variables burglary(x) and alarm(x) must be replicated for each person x; the domain consists of the set of persons as marked in the top of the plate. Note that each parameterized random variable must be associated with probabilistic assessments; in this case, the relevant ones from Figure 1. Right: A plate model that extends the one on the left by including the random variable calls.

Plates appeared with the BUGS package, to facilitate the specification of hierarchical models, and have been successful in applications [18]. One restriction of the original plate models in the BUGS package is that a parameterized random variable could not have children outside of its enclosing plate. However, in practice, many plate models violate this restriction. Figure 3 depicts a partial plate

model that satisfies the restriction of the original BUGS package (left), and the plate model that violates it (right). Note that as long as the graph consisting of parameterized random variables is acyclic, we know that every Bayesian network generated from the plate model is indeed consistent.

Several other combinations of parameterized random variables and graph-theoretical representations have been proposed, often grouped under the loose term "Probabilistic Relational Model (PRM)" [10,19,20]. Using PRMs, one can associate parameterized random variables with domains, impose constraints on domains and even represent limited forms of recursion [19,21]. A detailed description of PRMs is given in Section 4; for now, it suffices to say that a PRM is specified as a set of "classes" (each class is a set of individuals), where each class is associated with a set of parameterized random variables and additionally by a relational database that gives the relations amongst individuals in classes. The plate model in Figure 3 (left) can be viewed as a diagrammatic representation for a minimalistic PRM, where we have a class **Person** containing parameterized random variables. Note that such a minimalistic PRM with a single class **Person** cannot encode Scenario 4 in Example 2, as in that scenario, we have pairs of interacting individuals.

Suppose that we want a PRM to represent Scenario 4 in Example 2. Now, the class **Person** must include parameterized random variables burglary, alarm and calls. The challenge is how to indicate which **Person**s are parents of a particular calls(x). To do so, one possibility is to introduce another class, say **Neighborhood**, where each element of **Neighborhood** refers to two elements of **Person**. In Section 4 we show how the resulting PRM can be specified textually; for now, we want to point out that finding a diagrammatic representing this PRM is not an obvious matter. Using the scheme suggested by Getoor et al. [19], we might draw the diagram in Figure 4. There, we have a class **Person**, a class **Neighborhood** and a "shadow" class **Person** that just indicates the presence of a second **Person** in any **Neighborhood** pair. Dealing with all possible PRMs indeed requires a very complex diagrammatic language, where conditional edges and recursion can be expressed [21].

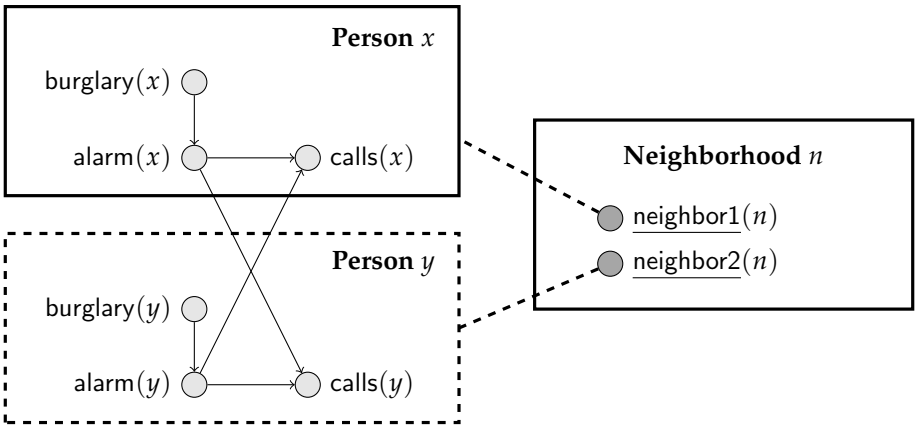


Figure 4. A Probabilistic Relational Model (PRM) for Scenario 4 in Example 2, using a diagrammatic scheme suggested by Getoor et al. [19]. A textual description of this PRM is presented in Section 4.

Instead of resorting to diagrams, one may instead focus just on textual languages to specify repetitive Bayesian networks. A very solid formalism that follows this strategy is Jaeger's Relational Bayesian Networks (RBNs) [22,23]. In RBNs, relations within domains are specified using a first-order syntax as input, returning an output that can be seen as a typical Bayesian network. For instance, using syntax that will be explained later (Section 2), one can describe Scenario 4 in Example 2 with the following RBN:

```
burglary(x) = 0.001;
alarm(x) = 0.9 * burglary(x) + 0.01 * (1-burglary(x));
calls(x) = NoisyOR { alarm(y) | y; neighbor(x,y) };
```

One problem that surfaces when we want to use an expressive formalism, such as RBNs or PRMs, is whether a particular model is guaranteed to always produce consistent Bayesian networks. Consider a simple example [19].

Example 3. *Suppose we are modeling genetic relationships using the parameterized random variable* $\mathsf{gene}(x)$*, for any person* x*. Now, the genetic features of* x *depend on the genetic features of the mother and the father of* x*. That is, we want to encode:*

> *If* y *and* z *are such that* $\mathsf{motherOf}(y, x)$ *and* $\mathsf{fatherOf}(z, x)$ *are* true*, then the probability of* $\mathsf{gene}(x)$ *depends on* $\mathsf{gene}(y)$ *and* $\mathsf{gene}(z)$*.*

It we try to specify a PRM for this setting, we face a difficulty in that some instances of gene *could depend on other instances of the same parameterized random variable. Indeed, consider drawing a diagram for this PRM, using the conventions suggested by Getoor et al. [19]. We would need a class* **Person**, *containing parameterized random variable* gene*, and two shadow classes, one for the father and one for the mother; a fragment of the diagram is depicted in Figure 5. If we could have a* **Person** *that appears as the father of his own father, we would have a cycle in the generated Bayesian network. Of course, we know that such a cycle can never be generated because neither the transitive closure of* motherOf*, nor of* fatherOf *can contain a cycle. However, just by looking at the diagram, without any background understanding of* motherOf *and* fatherOf*, we cannot determine whether coherence is guaranteed.*

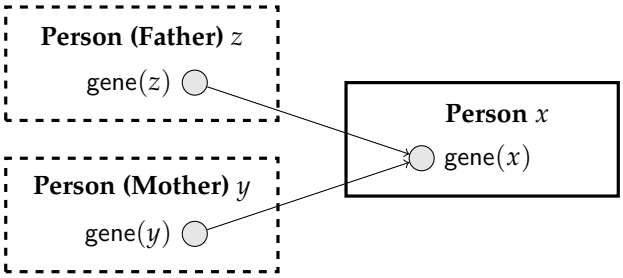


Figure 5. The PRM for the genetic example, as proposed by Getoor et al. [19].

The possibility that RBNs and PRMs may lead to cyclic (thus inconsistent) Bayesian networks has been noticed before. Jaeger [23] suggested that checking whether an RBN always produces consistent Bayesian networks, for a given class of domains, should be solved by logical inference, being reducible to deciding the validity of a formula from first-order logic augmented with a transitive closure operator. This path has been explored by De Bona and Cozman [24], yielding theoretical results of very high computational complexity. On a different path, Getoor et al. [19] put forward an incomplete, but more intuitive way of ensuring coherence for their PRMs; in fact, assuming that some sets of input relations never form cycles, one can easily identify a few cases where coherence is guaranteed.

Thus, we have arrived at the problems of interest in this paper: Suppose we have an RBN or a PRM. Is it *coherent* in the sense that it can be always satisfied by a probability distribution? Is it *coherent* in the sense that it always produces a unique probability distribution? Such are the questions we address, by exploring a cross-pollination of ideas described in the previous paragraph. In doing so, we bring logical rigor to the problem of coherence of PRMs and present a practical alternative to identifying coherent RBNs.

After formally introducing relational Bayesian networks in Section 2, we review, in Section 3, how its coherence problem can be encoded in first-order logic by employing a transitive closure operator. Section 4 presents PRMs and the standard graph-based approach to their coherence checking. The logical methods developed for the coherence problem of RBNs are adapted to PRMs in Section 5. Conversely, in Section 6, we adapt the graph techniques presented for tackling the coherence of PRMs to the formalism of RBNs.

2. Relational Bayesian Networks

In this section, we briefly introduce the formalism of Relational Bayesian Networks (RBNs). We use the version of RBNs presented in [23], as that reference contains a thorough exposition of the topic.

Let S and R be disjoint sets of relation symbols, called the *predefined relations* and *probabilistic relations*, respectively. We assume that S contains the equality symbol $=$, to be interpreted in the usual way. Each predicate symbol is associated with a positive integer k, which is its *arity*. Given a finite domain $D = \{d_1, \ldots, d_n\}$, if V is a set of relation symbols (as R or S), a *V-structure* $\mathcal{D}$ is an interpretation of the symbols in V into sets of tuples in D. Formally, a V-structure $\mathcal{D}$ maps each relation symbol $v \in V$ with arity k into a subset of D^k. We denote by $\mathsf{Mod}_D(V)$ the set of all V-structures over a given finite domain D. Given a domain D, a $v \in V$ with arity k and a tuple $t \in D^k$, $v(t)$ is said to be a *ground V-atom*. A V-structure $\mathcal{D}$ defines truth values for ground atoms: if v is mapped to a relation containing t, we say that $v(t)$ is *satisfied by* $\mathcal{D}$, which is denoted by $\mathcal{D} \models v(t)$.

Employing the syntax of function-free first-order logic, we can construct formulas using a vocabulary of relations V, together with variables, quantifiers and Boolean connectives. We call these *V-formulas*, and their meaning is given by the first-order logic semantics, as usual, through the V-structures. We denote by $\varphi(x_1, \ldots, x_m)$ a V-formula where $x_1, \ldots, x_k$ are free variables, in the usual sense. If φ is a V-formula and $\mathcal{D}$ is a V-structure, $\mathcal{D} \models \varphi$ denotes that φ is satisfied by $\mathcal{D}$.

A *random relational structure model for S and R* is a partial function P that takes an S-structure $\mathcal{D}$, over some finite domain D, and returns a probability distribution $P(\mathcal{D}) : \mathsf{Mod}_D(R) \to [0,1]$ over the R-structures on the same domain. As an R-structure can be seen as total assignment over the ground R-atoms, $P(\mathcal{D})$ can be seen as a joint probability distribution over these ground atoms. An example of random relational structure model would be a function P_{S4} in Scenario 4 of Example 2 that receives an S-structure of neighbors and returns a joint probability distribution over ground atoms for $\mathsf{burglary}(\cdot)$, $\mathsf{alarm}(\cdot)$, $\mathsf{calls}(\cdot)$. In that scenario, a given configuration $\mathcal{D}$ of neighbors, over a given domain D, implies a specific Bayesian network whose variables are the ground atoms for $\mathsf{burglary}(\cdot)$, $\mathsf{alarm}(\cdot)$, $\mathsf{calls}(\cdot)$, which encodes a joint probability distribution, $P_{S4}(\mathcal{D})$, over these variables. If $\mathcal{D}$ is the configuration of neighbors from Scenario 4 of Example 2, $P_{S4}(\mathcal{D})$ would be captured by the Bayesian network in Figure 2.

Relational Bayesian networks provide a way to compactly represent random relational structure models. This is achieved by mapping each S-structure into a *ground* Bayesian network that encodes a probability distribution over R-structures. To begin, this ground Bayesian network has nodes representing $r(t)$ (ground atoms), for each $r \in R$ and $t \in D^k$, where k is the arity of r. Thus, given the domain D of the input S-structure, the nodes in the corresponding Bayesian network are already determined. To define the arcs and parameters of the Bayesian network associated with an arbitrary S-structure, relational Bayesian networks employ their central notion of *probability formula*.

Probability formulas are syntactical constructions intended to link the probability of a ground atom $r(t)$ to the probabilities of other ground atoms $r'(t')$, according to the S-structure. Once an R-structure and an S-structure are fixed, then for elements $t_1, \ldots, t_k$ in the domain D, a probability formula $F(t_1, \ldots, t_k)$ should evaluate to a number in $[0,1]$.

The definition of probability formulas makes use of *combination functions*, which are functions from finite multi-sets over the interval $[0,1]$ to numbers in the same interval. We use $\{|\cdot|\}$ to denote multi-sets. For instance, NOISYOR is a combination function such that, if $c_1, \ldots, c_n \in [0,1]$, $\text{NOISYOR}\{|c_1, \ldots, c_n|\} = 1 - \prod_{i=1}^{n}(1 - c_i)$.

Definition 1. Given disjoint sets S and R of relation symbols and a tuple x of k variables, $F(x)$ is a *(S,R)-probability formula* if:

- (constants) $F(x) = c$ for a $c \in [0,1]$;
- (indicator functions) $F(x) = r(x)$ for an $r \in R$ with arity k;

- (convex combinations) $F(x) = F_1(x)F_2(x) + (1 - F_1(x))F_3(x)$, where $F_1(x), F_2(x), F_3(x)$ are probability formulas, or;
- (combination functions) $F(x) = \mathsf{comb}\{|F_1(x,y), \ldots, F_m(x,y)|y; \varphi(x,y)|\}$, where comb is a combination function, $F_1(x,y), \ldots, F_m(x,y)$ are probability formulas, y is a tuple of variables and $\varphi(x,y)$ is an S-formula.

Relational Bayesian networks associate a probability formula $F_{r\star}(x)$ to each probabilistic relation $r\star \in R$, where x is a tuple of k variables, with k the arity of $r\star$:

Definition 2. Given disjoint sets of relation symbols S and R, the predefined and probabilistic relations, a *relational Bayesian network* is a set $\Phi = \{F_r(x) \mid r \in R\}$, where each $F_r(x)$ is a (S,R)-probability formula.

To have an idea of how probability formulas work, consider a fixed S-structure $\mathcal{D}_S$ over a domain D. Then, an R-structure $\mathcal{D}_\mathcal{R}$ over D entails a numeric value for each *ground probability formula* $F_{r\star}(t)$, denoted by $F_{r\star}(t)[\mathcal{D}_R]$, where t is tuple of elements in D. This is done inductively, by initially defining $r(t)[\mathcal{D}_R] = 1$ if $\mathcal{D}_R \models r(t)$, and $r(t)[\mathcal{D}_R] = 0$ otherwise, for each $r(t)$, for all $r \in R$. If $F_{r\star}(x) = c$, then $F_{r\star}(t)[\mathcal{D}_R] = c$, for any tuple t. The numeric value of $F_{r\star}(t)[\mathcal{D}_R]$ for probability formulas that are convex combinations or combination function will require the evaluation of its subformulas F_i, which recursively end at the evaluation of ground atoms $r(t)$ or constants c. As the set of ground atoms whose evaluation is needed to compute $F_{r\star}(t)[\mathcal{D}_R]$ depends only on the S-structure $\mathcal{D}_S$, and not on $\mathcal{D}_R$, it is denoted by $\alpha(F_{r\star}(x), t, \mathcal{D}_S)$ and can be defined recursively:

- $\alpha(c, t, \mathcal{D}_S) = \varnothing$;
- $\alpha(r(x), t, \mathcal{D}_S) = \{r(t)\}$;
- $\alpha(F_1(x)F_2(x) + (1 - F_1(x))F_3(x), t, \mathcal{D}_S) = \bigcup_{i=1}^{3} \alpha(F_i(x), t, \mathcal{D}_S)$;
- $\alpha(\mathsf{comb}\{|F_1(x,y), \ldots, F_m(x,y)|y; \varphi(x,y)|\}, t, \mathcal{D}_S)$ is given by:

$$\bigcup_{t' \text{ s.t. } \mathcal{D}_S \models \varphi(t,t')} \bigcup_{i=1}^{m} \alpha(F_i(x,y), (t,t'), \mathcal{D}_S).$$

Here, (t,t') denotes the concatenation of the tuples t and t'.

For a given S-structure $\mathcal{D}_S$, we can define a dependency relation between the nodes $r(t)$ and $r'(t')$ in the Bayesian network via the probability formulas F_r and $F_{r'}$ by employing the corresponding $\alpha(\cdot,\cdot,\cdot)$. Intuitively, $\alpha(F_r(x), t, \mathcal{D}_S)$ contains the ground atoms $r'(t')$ whose truth value in a structure $\mathcal{D}_R$ determines the value of $F_r(t)$, which is meant to be the probability of $r(t)$. That is, $\alpha(F_r(x), t, \mathcal{D}_S)$ contains the parents of $r(t)$.

Definition 3. *Relation $\preceq$, over ground R-atoms, is defined as follows:*

$$r(t) \preceq r'(t') \quad \textit{iff} \quad r'(t') \in \alpha(F_r(x), t, \mathcal{D}_S).$$

When this relation is acyclic, a relational Bayesian network $\Phi = \{F_r \mid r \in R\}$ defines, for a given S-structure $\mathcal{D}_S$ over a finite domain D, a probability distribution over the R-structures $\mathcal{D}_R$ over D via:

$$P^{\Phi}_{\mathcal{D}_S}(\mathcal{D}_R) = \prod_{r \in R} \prod_{t, \mathcal{D}_R \models r(t)} F_r(t)[\mathcal{D}_R] \prod_{t, \mathcal{D}_R \not\models r(t)} (1 - F_r(t)[\mathcal{D}_R])$$

Example 4. *Scenario 4 of Example 2: We can define a relational Bayesian network that returns the corresponding Bayesian network for each number and configuration of neighbors. Let* $S = \{\mathsf{neighbor}(\cdot,\cdot)\}$ *and* $R = \{\mathsf{burglary}(\cdot), \mathsf{alarm}(\cdot), \mathsf{calls}(\cdot)\}$. *We assume that the relation* neighbor *is reflexive and symmetrical. For each relation in R, we associate a probability formula, forming the relational Bayesian network* Φ*:*

- $F_{\mathsf{burglary}}(x) = 0.001$; *a constant;*
- $F_{\mathsf{alarm}}(x) = 0.9\mathsf{burglary}(x) + 0.01(1 - \mathsf{burglary}(x))$; *a convex combination;*
- $F_{\mathsf{call}}(x) = \text{NOISYOR}\{|\mathsf{alarm}(y) \mid y; \mathsf{neighbor}(x, y)|\}$; *a combination function.*

Note that, if $F_1(x)$ and $F_2(x)$ are probability formulas, then $1 - F_1(x)$ and $F_1(x)F_2(x)$ are convex combinations and, thus, probability formulas. As the inputs of the NOISYOR *above are in $\{0, 1\}$, the combination function actually works like a disjunction.*

Given an S-structure $\mathcal{D}_S$ over a domain D, Φ determines a joint probability distribution over the ground R-atoms, via a Bayesian network. If we take an S-structure $\mathcal{D}_S$ over a domain $D = \{d_1, d_2, d_3\}$ such that $\mathcal{D}_S \models \mathsf{neighbor}(d_1, d_2) \wedge \mathsf{neighbor}(d_2, d_3)$, but $\mathcal{D}_S \not\models \mathsf{neighbor}(d_1, d_3)$, the resulting $P^{\Phi}_{\mathcal{D}_S}$ is the model for Scenario 3 in Example 2, whose Bayesian network is given in Figure 2.

3. The Coherence Problem for RBNs

It may happen for a relational Bayesian network Φ that some S-structures yield a cyclic dependency relation $\preceq$. When the relation $\preceq$ is cyclic for an S-structure, no probability distribution is defined over the R-structures. In such a case, we say Φ is *incoherent* for that S-structure. This notion can be generalized to a class of S-structures $\mathcal{S}$, so that we say that Φ is *coherent* for $\mathcal{S}$ iff the resulting relation $\preceq$ is acyclic for each S-structure in $\mathcal{S}$. To know whether a relational Bayesian network is coherent for a given class of S-structures is precisely one of the problems we are interested in this work.

In order to reason about the relation between a class of S-structures and the coherence of a relational Bayesian network Φ for it, we need to formally represent these concepts. To define a class of S-structures, note that they can be seen as first-order structures over which S-formulas are interpreted. That is, an S-formula defines the set of S-structures satisfying it. If φ is a closed S-formula (without free variables), we say that $[[\varphi]]$ is the set of S-structures $\mathcal{D}_S$ such that $\mathcal{D}_S \models \varphi$. We denote by $\theta_{\mathcal{S}}$ an S-formula satisfied exactly by the S-structures in a given class $\mathcal{S}$; that is, $[[\theta_{\mathcal{S}}]] = \mathcal{S}$.

To encode the coherence of Φ, we need to encode the acyclicity of the dependency relation $\preceq$ resulting from an S-structure. Ideally, we would like to have a (first-order) S-formula, say ψ_Φ, that would be true only for S-structures yielding acyclic dependency relations $\preceq$. If that formula were available, a decision about the coherence of Φ for the class $\mathcal{S}$ would be reduced to a decision about the validity of the first-order formula $\theta_{\mathcal{S}} \to \psi_\Phi$: When the formula is valid, then every S-structure in the class $\mathcal{S}$ guarantees that the resulting dependency relation $\preceq$ for Φ is acyclic; hence, Φ is coherent for $\mathcal{S}$; otherwise, there is an S-structure in $\mathcal{S}$ yielding a cyclic dependency relation $\preceq$ for Φ. Note that for S-formulas, only S-structures matter, and we could ignore any relation not in S. To be precise, if a first-order structure $\mathcal{D}$ falsifies $\theta_{\mathcal{S}} \to \psi_\Phi$, then there is an S-structure $\mathcal{D}_S$ (formed by ignoring non-S relations) falsifying it.

Alas, to encode cycles in a graph, one needs to encode the notion of path, which is the transitive closure of a relation encoding arcs. It is a well-known fact that first-order logic cannot express transitivity. To circumvent that, we can add a (strict) transitive closure operator to the logic, arriving at the so-called transitive closure logics, as described for instance in [25].

This approach was first proposed by Jaeger [23], who assumed one could write down the S-formula ψ_Φ by employing a transitive closure operator. He conjectured that with some restrictions on the arity of the relations in S and R, one could hope to obtain a formula $\theta_{\mathcal{S}} \to \psi_\Phi$ that is decidable. Nevertheless, no hint was provided as to how to construct such a formula, or as to its general shape. A major difficulty is that, if an S-structure $\mathcal{D}$ satisfying $\theta_{\mathcal{S}}$ has domain $D = \{d_1, \ldots, d_n\}$, the size of the resulting Bayesian network is typically greater than n, with one node per ground atom, so a cycle can also contain more nodes than n. There seems to be no direct way of employing the transitive closure operator to devise a formula $\neg\psi_\Phi$ that encodes cycles with more than n nodes and that is to be satisfied by some structures $\mathcal{D}$ over a domain with only n elements. In the next sections, we review a technique (introduced by the authors in [24]) to encode ψ_Φ for an augmented domain, through an auxiliary formula whose satisfying structures will represent both the S-structure and the resulting ground Bayesian network. Afterwards, we adapt the formula $\theta_{\mathcal{S}}$ accordingly.

3.1. Encoding the Structure of the Ground Bayesian Network

Our idea to construct a formula ψ_Φ, for a given relational Bayesian network Φ, is first to find a first-order V-formula $\mathcal{B}_\Phi$, for some vocabulary V containing S, that is satisfiable only by V-structures that encode both an S-structure $\mathcal{D}_S$ and the structure of the ground Bayesian network resulting from it. These V-structures should contain, besides an S-structure $\mathcal{D}_S$, an element for each node in the ground Bayesian network and a relation capturing its arcs. Then, we can use a transitive closure operator to define the existence of paths (and cycles) via arcs, for enforcing acyclicity by negating the existence of a cycle.

Suppose we have two disjoint vocabularies S and $R = \{r_1, \ldots, r_m\}$ of predefined and probabilistic relations, respectively. We use $a(v)$ to denote the arity of a relation v. Consider a relational Bayesian network $\Phi = \{F_r(x) \mid r \in R\}$, where each $F_r(x)$ is a (S,R)-probability formula. Let $\mathcal{D}$ be a V-structure satisfying $\mathcal{B}_\Phi$. We want $\mathcal{D}$ to be defined over a bipartite domain $D = D_S \cup D_B$, where D_S is used to represent an S-structure $\mathcal{D}_S$ and $D_B = D \setminus D_S$ is the domain where the structure of the resulting ground Bayesian network is encoded. We overload names by including in V a unary predicate $D_S(\cdot)$ that shall be true for all and only the elements in D_S. The structure $\mathcal{D}$ shall represent the structure of the ground Bayesian network $B_\Phi(\mathcal{D}_S)$, over the elements of D_B, that is induced by the S-structure $\mathcal{D}_S$ codified in D_S. In order to accomplish that, $\mathcal{D}$ must have an element in D_B for each ground atom over the domain D_S. Furthermore, the V-structure $\mathcal{D}$ must interpret a relation, say $Parent(\cdot,\cdot)$, over D_B according to the arcs of the Bayesian network $B_\Phi(\mathcal{D}_S)$.

Firstly, we need to define a vocabulary V that includes the predefined relations in S and contains the unary predicate D_S (recall that the equality symbol (=) is included in S). Furthermore, V must contain a binary relation *Parent* to represent the arcs of the ground Bayesian network. As auxiliary relations for defining *Parent*, we will need a relation Dep_i^j, for each pair $r_i, r_j \in R$, whose arity is $a(r_i) + a(r_j)$. For elements in D_B to represent ground atoms $r(t_1, \ldots, t_n)$, we use relations to associate elements in D_B to relations r and to tuples $\langle t_1, \ldots, t_n \rangle$. For each relation $r_i \in R$, we have a unary relation $\bar{r}_i \in V$, where $\bar{r}_i(x)$ is intended to mean that the element $x \in D_B$ represents a ground atom of the form $r_i(\cdot)$. As for the tuples, recall that each t_i represents an element in the set D_S over which the S-structure $\mathcal{D}_S$ is codified. Hence, we insert in V binary relations t_i for every $1 \leq i \leq \max_i a(r_i)$, such that $t_i(x, y)$ should be true iff the element $x \in D_B$ corresponds to a ground atom $r(t_1, \ldots, t_k)$ where $t_i = y$, for a $y \in D_S$ and some $r \in R$.

To save notation, we use $R_i(x, y_1, \ldots, y_k)$ to denote $\bar{r}_i(x) \wedge t_1(x, y_1) \wedge \cdots \wedge t_k(x, y_k)$ henceforth, meaning the element x in the domain represents the ground atom $r_i(y_1, \ldots, y_k)$, where $a(r_i) = k$.

Now, we proceed to list, step-by-step, the set of conjuncts required in ψ_Φ, together with their meaning, for the V-structure $\mathcal{D}$ in $[[\psi_\Phi]]$ to hold the desired properties. To illustrate the construction, each set of conjuncts is followed by an example based on the RBN in Example 4, possibly given in an equivalent form for clarity.

We have to ensure that the elements in D_B correspond exactly to the ground atoms in the ground Bayesian network $B_\Phi(\mathcal{D}_S)$.

- Each element in $D_B = D \setminus D_S$ should correspond to a ground atom for some $r_i \in R$. Hence, we have the formula:

$$\forall x \neg D_S(x) \rightarrow \bigvee_{i=1}^{m} \bar{r}_i(x). \tag{2}$$

$$\forall x \neg D_S(x) \rightarrow \overline{\mathsf{burglary}}(x) \vee \overline{\mathsf{alarm}}(x) \vee \overline{\mathsf{calls}}(x).$$

- No element may correspond to ground atoms for two different $r_i \in R$. Therefore, the formula below is introduced:

$$\forall x \bigwedge_{1 \leq i,j \leq m}^{i \neq j} (\neg \bar{r}_i(x) \vee \neg \bar{r}_j(x)). \tag{3}$$

$$\forall x (\neg \overline{\mathsf{burglary}}(x) \vee \neg \overline{\mathsf{alarm}}(x)) \wedge (\neg \overline{\mathsf{burglary}}(x) \vee \neg \overline{\mathsf{calls}}(x)) \wedge (\neg \overline{\mathsf{alarm}}(x) \vee \neg \overline{\mathsf{calls}}(x)).$$

- Each element corresponding to a ground atom should correspond to exactly one tuple. To achieve that, let $k = \max_j a(r_j)$, and introduce the formula below:

$$\forall x \forall y \forall z \bigwedge_{j=1}^{k} (t_j(x,y) \wedge t_j(x,z) \rightarrow y = z). \tag{4}$$

$$\forall x \forall y \forall z (t_1(x,y) \wedge t_1(x,z) \rightarrow y = z.$$

- Each element corresponding to a ground atom for a $r_i \in R$ should be linked a to tuple with arity $a(r_i)$. Thus, let $k = \max_j a(r_j)$, and introduce the formula below for each $r_i \in R$:

$$\forall x \bar{r}_i(x) \rightarrow (\exists y_1 \ldots \exists y_{a(r_i)} R_i(x, y_1, \ldots, y_{a(r_i)}) \wedge \forall z \neg t_{a(r_i)+1}(x,z) \wedge \cdots \wedge \neg t_k(x,z)). \tag{5}$$

$$\forall x \overline{\mathsf{burglary}}(x) \rightarrow (\exists y t_1(x,y)); \quad \forall x \overline{\mathsf{alarm}}(x) \rightarrow (\exists y t_1(x,y)); \quad \forall x \overline{\mathsf{calls}}(x) \rightarrow (\exists y t_1(x,y)).$$

- Only elements in $D_B = D \setminus D_S$ should correspond to ground atoms. This is enforced by the following formula, where $k = \max_i a(r_i)$:

$$\forall y D_S(y) \rightarrow (\bigwedge_{i=1}^{m} \neg \bar{r}_i(y) \wedge \forall x \bigwedge_{j=1}^{k} \neg t_j(y,x)). \tag{6}$$

$$\forall y D_S(y) \rightarrow (\neg \overline{\mathsf{burglary}}(y) \wedge \neg \overline{\mathsf{alarm}}(y) \wedge \neg \overline{\mathsf{calls}}(y) \wedge \forall x \neg t_1(y,x)).$$

- Each ground atom must be represented by at least one element (in $D_B = D \setminus D_S$). Therefore, for each $r_i \in R$, with $a(r_i) = k$, we need a formula:

$$\forall y_1 \ldots \forall y_k D_S(y_1) \wedge \cdots \wedge D_S(y_k) \rightarrow \exists x R_i(x, y_1, \ldots, y_k). \tag{7}$$

$$\forall y D_S(y) \rightarrow (\exists x_1 \overline{\mathsf{burglary}}(x_1) \wedge t_1(y, x_1)); \text{ same for } \overline{\mathsf{alarm}} \text{ and } \overline{\mathsf{calls}}.$$

  These formulas enforce that each ground atom $r(t)$ is represented by an element x that is in D_B, due to the formula (6).
- No ground atom can be represented by two different elements. Hence, for each $r_i \in R$, with $a(r_i) = k$, we introduce a formula:

$$\forall y_1, \ldots \forall y_k \forall x \forall z R_i(x, y_1, \ldots, y_k) \wedge R_i(z, y_1, \ldots, y_k) \rightarrow x = z. \tag{8}$$

$$\forall y \forall x \forall z \overline{\mathsf{burglary}}(x) \wedge t_1(y,x) \wedge \overline{\mathsf{burglary}}(z) \wedge t_1(z,y) \rightarrow x = z; \text{ same for } \overline{\mathsf{alarm}} \text{ and } \overline{\mathsf{calls}}.$$

The conjunction of all formulas in (2)–(8) is satisfied only by structures $\mathcal{D}$ over the domain $D = D_S \cup D_B$ such that there is a bijection between D_B and the set of all possible ground atoms $\{r(t) \mid$ for some $r \in R$ and $t \in D_S^{a(r)}\}$. Now, we can put the arcs over these nodes to complete the structure of the ground Bayesian network $B_\Phi(\mathcal{D}_S)$.

The binary relation *Parent* must hold only between elements in the domain D representing ground atoms $r(t)$ and $r'(t')$ such that $r(t) \preceq r'(t')$. Recall that the dependency relation $\preceq$ is determined by the S-structure $\mathcal{D}_S$. While the ground atoms represented in D_B, for a fixed R, are determined by the size of D_S by itself, the relation *Parent* between them depends also on the S-formulas that hold for the S-structure $\mathcal{D}_S$. We want these S-structures to be specified by $\mathcal{D}$ over D_S only, not over D_B. To ensure this, we use the following group of formulas:

- For all $s \in S$, consider the formula below, where $a(s) = k$:

$$\forall y_1 \ldots \forall y_k \, s(y_1, \ldots, y_k) \rightarrow D_S(y_1) \wedge \cdots \wedge D_S(y_k). \tag{9}$$

$$\forall y_1 \forall y_2 \, \mathsf{neighbor}(y_1, y_2) \rightarrow D_S(y_1) \wedge D_S(y_2).$$

The formula above forces that $s(t)$, for any $s \in S$, can be true only for tuples $t \in D_S^{a(s)}$.

For a known S-structure $\mathcal{D}_S$, it is straightforward to determine which ground atoms $r'(t')$ are the parents of $r(t)$ in the ground Bayesian network $B_\Phi(\mathcal{D}_S)$. One can use recursively the definition of the set of parents $\alpha(F_r(x), t, \mathcal{D}_S)$ given in Section 2. Nonetheless, with an unknown S-structure $\mathcal{D}_S$ specified in $\mathcal{D}$ over D_S, the situation is a bit trickier. The idea is to construct, for each pair $r_i(t)$ and $r_j(t')$, an S-formula $Dep_i^j(t,t')$ that is true iff $r_i(t) \preceq r_j(t')$ for the $\mathcal{D}_S$ encoded in $\mathcal{D}$. To define $Dep_i^j(t,t')$, we employ auxiliary formulas $C_{F(t)}^{r'(t')}$, for a ground probability formula $F(t)$ and a ground atom $r'(t')$, that will be an S-formula that is satisfied by $\mathcal{D}$ iff $r'(t') \in \alpha(F(x), t, \mathcal{S})$. We define $C_{F(t)}^{r'(t')}$ recursively, starting from the base cases.

- If $F(t) = c$, for a $c \in [0,1]$, then $C_{F(t)}^{r'(t')} = \bot$; e.g., $C_{F_{\mathsf{burglary}}(t)}^{\mathsf{alarm}(t')} = \bot$.
- If $F(t) = r''(t)$, then $C_{F(t)}^{r'(t')} = (t' = t)$ if $r' = r''$; and $C_{F(t)}^{r'(t')} = \bot$ otherwise;
  e.g., $C_{F_{\mathsf{burglary}}(t)}^{\mathsf{burglary}(t')} = (t = t')$ and $C_{F_{\mathsf{burglary}}(t)}^{\mathsf{calls}(t')} = \bot$.

Above, $(t' = t)$ is a short form for $(t'_1 = t_1) \wedge \cdots \wedge (t'_k = t_k)$, where k is the arity of t. These base cases are in line with the recursive definition of $\alpha(F(x), t, \mathcal{S})$ presented in Section 2. The third case is also straightforward:

- If $F(t) = F_1(t)F_2(t) + (1 - F_1(t))F_3(t)$, then $C_{F(t)}^{r'(t')} = \bigvee_{i=1}^{3} C_{F_i(t)}^{r'(t')}$.
  $C_{F_{\mathsf{alarm}}(t)}^{\mathsf{burglary}(t')} = C_{F_{\mathsf{burglary}}(t)}^{\mathsf{burglary}(t')} \vee C_{0.9}^{\mathsf{burglary}(t')} \vee C_{0.01}^{\mathsf{burglary}(t')} = (t = t') \vee \bot \vee \bot$

In other words, the computation of $F(t)[\mathcal{D}_R]$ depends on $r'(t')[\mathcal{D}_R]$, for some $\mathcal{D}_R$, if the computation of some $F_i(t)[\mathcal{D}_R]$, for $1 \leq i \leq 3$, depends on $r'(t')[\mathcal{D}_R]$.

The more elaborated case happens when $F(x)$ is a combination function, for which there is an S-formula involved. Recall that if $F(x) = \mathsf{comb}\{|F_1(x,y), \ldots, F_m(x,y)|y; \varphi(x,y)|\}$, then the parents of $F(t)$ are given by $\bigcup_{t', \mathcal{D}_S \models \varphi(t,t')} \bigcup_{i=1}^{m} \alpha(F_i(x,y), (t,t'), \mathcal{D}_S)$. Thus, to recursively define $C_{F(t)}^{r'(t')}$, we need an S-formula that is satisfied by an S-structure $\mathcal{D}_S$ iff:

$$r'(t') \in \bigcup_{t^\star, \mathcal{D}_S \models \varphi(t,t^\star)} \bigcup_{i=1}^{m} \alpha(F_i(x,y), (t,t^\star), \mathcal{D}_S).$$

The inner union is analogous to the definition of $C_{F(t)}^{r'(t')}$ for convex combinations. However, to cope with any $t^\star$ such that $\mathcal{D}_S \models \varphi(t,t^\star)$, we need an existential quantification:

- If $F(x) = \mathsf{comb}\{|F_1(x,y), \ldots, F_m(x,y)|y; \varphi(x,y)|\}$, then we have that:

$$C_{F(t)}^{r'(t')} = \exists t^\star \varphi(t,t^\star) \wedge \bigvee_{i=1}^{m} C_{F_i(t,t^\star)}^{r'(t')}.$$

$$C_{F_{\mathsf{calls}}(t)}^{\mathsf{alarm}(t')} = \exists t^\star \mathsf{neighbor}(t,t^\star) \wedge C_{F_{\mathsf{alarm}}(t^\star)}^{\mathsf{alarm}(t')} = \exists t^\star \mathsf{neighbor}(t,t^\star) \wedge (t^\star = t')$$

Now, we can employ the formulas $C_{F(t)}^{r'(t')}$ to define the truth value of the ground relation $Dep_i^j(t,t')$, that codifies when $r_i(t) \preceq r_j(t')$.

- For each pair $r_i, r_j \in R$, with $a(r_i) = k$ and $a(r_j) = k'$, we have the formula:

$$\forall x_1 \ldots \forall x_k \forall y_1 \ldots \forall y_{k'} Dep_i^j(x_1, \ldots, x_k, y_1, \ldots, y_{k'}) \leftrightarrow C_{F_{r_i}(x_1,\ldots,x_k)}^{r_j(y_1,\ldots,y_{k'})}. \tag{10}$$

$$\forall x \forall y Dep_{\mathsf{calls}}^{\mathsf{alarm}}(x,y) \leftrightarrow \exists z \mathsf{neighbor}(x,z) \wedge (z = y); \quad \forall x \forall y Dep_{\mathsf{alarm}}^{\mathsf{burglary}}(x,y) \leftrightarrow (x = y).$$

In the formula above, $C^{r_j(y_1,\ldots,y_{k'})}_{F_{r_i}(x_1,\ldots,x_k)}$ has free variables $x_1,\ldots,x_k,y_1,\ldots,y_{k'}$ and is built according to the four recursive rules that define $C^{r'(t')}_{F(t)}$, replacing the tuples t and t' by x and y. We point out that such construction depends only on probability formulas in the relational Bayesian network Φ, and not on any S-structure. To build each $C^{r_j(y)}_{F_{r_i}(x)}$, one just starts from the probability formula $F_{r_i}(x)$ and follows the recursion rules until reaching the base cases, when $C^{r_j(y)}_{F_{r_i}(x)}$ will be formed by subformulas like $\top$, $\bot$, S-formulas $\varphi(\cdot)$ and equalities $(\cdot=\cdot)$, possibly quantified on variables appearing in φ.

The relation $Parent(\cdot,\cdot)$ is defined now over elements that represent ground atoms $r_i(t)$ and $r_j(t')$ such that $Dep^j_i(t,t')$, meaning that $r_i(t) \preceq r_j(t')$. This can be achieved in two parts: ensuring that each $r_i(t) \preceq r_j(t')$ implies $Parent(t,t')$; and guaranteeing that $Parent(t,t')$ is true only if $r_i(t) \preceq r_j(t')$ for a pair of relations r_i, r_j.

- For each pair $r_i, r_j \in R$, with $a(r_i)=k$ and $a(r_j)=k'$, let y and y' denote $y_1,\ldots,y_k$ and $y'_1,\ldots,y'_{k'}$, respectively:

$$\forall x\forall x'\forall y_1\ldots\forall y_k\forall y'_1\ldots\forall y'_{k'} R_i(x,y)\wedge R_j(x',y')\wedge Dep^j_i(y,y')\rightarrow Parent(x,x'). \tag{11}$$

$$\forall x\forall x'\forall y\forall y' \mathsf{calls}(x)\wedge t_1(y,x)\wedge \mathsf{alarm}(x')\wedge t_1(y',x)\wedge Dep^{\mathsf{alarm}}_{\mathsf{calls}}(y,y')\rightarrow Parent(x,x').$$

- Let $k=\max_j a(r_j)$ be the maximum arity in R, and let y and y denote the tuples $y_1,\ldots,y_{a(r_i)}$ and $y'_1,\ldots,y'_{a(r_j)}$, respectively:

$$\forall x\forall x' Parent(x,x')\rightarrow \exists y_1\ldots\exists y_k\exists y'_1\ldots\exists y'_k \bigvee_{1\le i,j\le m} R_i(x,y_{r_i})\wedge R_j(x',y'_{r_j})\wedge Dep^j_i(y_{r_i},y'_{r_j}). \tag{12}$$

Definition 4. *Given disjoint sets of relations S and R and a relational Bayesian network $\Phi=\{F_{r_i}\mid r_i\in R\}$, the formula $\mathcal{B}_\Phi$ is the conjunction of all formulas in (2)–(12).*

For some fixed relational Bayesian networks Φ, the formula $\mathcal{B}_\Phi$ is satisfied only by V-structures $\mathcal{D}$ over a bipartite domain $D_S\cup D_B$ such that:

- the relations in S are interpreted in D_S, forming an S-structure $\mathcal{D}_S$;
- there is a bijection b between the domain $D_B=D\setminus D_S$ and set of all ground R-atoms formed by the tuples in D_S;
- each $x\in D_B$ is linked exactly to one $r_i\in R$, via the predicate $\bar{r}_i(x)$, and exactly $k=a(r_i)$ elements in D_S, via the relations $t_1(x,.),\ldots t_k(x,.)$, and no ground atom is represented through these links twice;
- the relation $Parent(\cdot,\cdot)$ is interpreted as arcs in D_B in such a way that $\langle D_B, Parent\rangle$ form a directed graph that is the structure of the ground Bayesian network $B_\Phi(\mathcal{D}_S)$.

3.2. Encoding Coherence via Acyclicity

The original formula ψ_Φ was intended to capture the coherence of the relational Bayesian network Φ. Our idea is to check the coherence by looking for cycles in the ground Bayesian network $B_\Phi(\mathcal{D}_S)$ encoded in any V-structure satisfying $\mathcal{B}_\Phi$. Hence, we replace ψ_Φ by an implication $\mathcal{B}_\Phi\rightarrow\psi'$, which is to be satisfied only by V-structures $\mathcal{D}$ such that, if $\mathcal{D}$ represents an S-structure $\mathcal{D}_S$ and the resulting ground Bayesian network $B_\Phi(\mathcal{D}_S)$, then $B_\Phi(\mathcal{D}_S)$ is acyclic. Thus, ψ' should avoid cycles of the relation *Parent* in the V-structures satisfying it.

There is a cycle with *Parent*-arcs in a V-structure $\mathcal{D}$ over a domain D iff there exists a $x\in D$ such that there is a path of *Parent*-arcs from x to itself. Consequently, detecting *Parent*-cycles reduces to computing *Parent*-paths or *Parent*-reachability. We say y is *Parent*-reachable from x, in a V-structure $\mathcal{D}$, if there are $z_0,\ldots,z_k\in D$ such that $x=z_0$, $y=z_k$, and $\mathcal{D}\models\bigwedge_{1\le i\le k} Parent(z_{i-1},z_i)$. Thus, for each

k, we can define reachability through k *Parent*-arcs: $ParentPath_k(x,y) = \exists z_0 \ldots \exists z_k (z_0 = x) \wedge (z_k = y) \wedge \bigwedge_{1 \leq i \leq k} Parent(z_{i-1}, z_i)$. Unfortunately, the size of the path (k) is unbounded *a priori*, as the domain D can be arbitrarily large. Therefore, there is no means in the first-order logic language to encode reachability, via arbitrarily large paths, with a finite number of formulas. In order to circumvent this situation, we can resort to a transitive closure logic.

Transitive closure logics enhance first-order logics with a transitive closure operator TC that we assume to be strict [25]. If $\varphi(x,y)$ is a first-order formula, $\mathsf{TC}(\varphi)(x,y)$ means that y is φ-reachable from x, with a non-empty path. Accordingly, a V-structure $\mathcal{D}$, over a domain D, satisfies $\mathsf{TC}(\varphi)(x,y)$ iff there is a $k \in \mathbb{N}$ and there are $z_0, \ldots, z_k \in D$ such that $x = z_0$, $y = z_k$ and $\mathcal{D} \models \bigwedge_{1 \leq i \leq k} \varphi(z_{i-1}, z_i)$.

Employing the transitive closure operator, the existence of a *Parent*-path from a node x to itself (a cycle) is encoded directly by $\mathsf{TC}(Parent)(x,x)$; similarly, the absence of a *Parent*-cycle is enforced by $\psi' = \forall x \neg \mathsf{TC}(Parent)(x,x)$.

At this point, the V-structures $\mathcal{D}$ over a domain D satisfying $\mathcal{B}_\Phi \rightarrow \psi'$ have the following format:

- either it encodes an S-structure in $D_S \subseteq D$ (the part of the domain satisfying $D_S(\cdot)$) and the corresponding *acyclic* ground Bayesian network $B_\Phi(\mathcal{D}_S)$ in $D_B = D \setminus D_S$.
- or it is not the case that $\mathcal{D}$ encodes both an S-structure in $D_S \subseteq D$ and the corresponding ground Bayesian network $B_\Phi(\mathcal{D}_S)$ in $D_B = D \setminus D_S$;

Back to the coherence-checking problem, we need to decide, for a fixed relational Bayesian network Φ, whether or not a given class $\mathcal{S}$ of S-structures ensures the acyclicity of the resulting ground Bayesian network $B_\Phi(\mathcal{D}_S)$. Recall that the class $\mathcal{S}$ must be defined via a (first-order) S-formula $\theta_\mathcal{S}$. As we are already employing the transitive closure operator in ψ', we can also allow its use in $\theta_\mathcal{S}$, which is useful to express S-structures without cycles, for instance.

To check the coherence of Φ for a class $\mathcal{S}$, we cannot just check the validity of:

$$\theta_\mathcal{S} \rightarrow (\mathcal{B}_\Phi \rightarrow \psi'), \tag{13}$$

because $\theta_\mathcal{S}$ specifies S-structures over D, while $\mathcal{B}_\Phi \rightarrow \psi'$ presupposes that the S-structure is given only over $D_S = \{d \in D \mid \mathcal{D} \models D_S(d)\} \subsetneq D$. To see the kind of problem that might occur, think of the class $\mathcal{S}$ of all S-structures $\mathcal{D}$ where each $d \in D$ is such that $s_i(d)$ holds, for some unary predefined relation $s_i \in S$. Consider an S-structure $\mathcal{D} \in \mathcal{S}$ ($\mathcal{D} \models \theta_\mathcal{S}$), over a domain D. The formula $\mathcal{B}_\Phi$ cannot be satisfied by $\mathcal{D}$, for $D_S(x)$ must hold for all $x \in D$, because of the formulas in (9), so no $x \in D$ can represent ground formulas, due to the formulas in (6), contradicting the restrictions in (7) that require all ground atoms to be represented. Hence, this $\mathcal{D}$ satisfies $\theta_\mathcal{S}$ without encoding the ground Bayesian network, thus falsifying $\mathcal{B}_\Phi$ and satisfying $\mathcal{B}_\Phi \rightarrow \psi'$, yielding the satisfaction of Formula (13). Consequently, Formula (13) is valid for this specific class $\mathcal{S}$, no matter what the relational Bayesian network Φ looks like. Nonetheless, it is not hard to think of a Φ that is trivially incoherent for any class of S-structures, like $\Phi = \{F_r(x) = r(x)\}$, with $S = \varnothing$ and $R = \{r\}$, where the probability formula associated with the relation $r \in R$ is the indicator function $r(x)$, yielding a cyclic dependency relation $\preceq$.

In order to address the aforementioned issue, we need to adapt $\theta_\mathcal{S}$, constructing $\theta'_\mathcal{S}$ to represent the class $\mathcal{S}$ in the extended, bipartite domain $D = D_S \cup D_B$. The unary predicate $D_S(\cdot)$ is what delimits the portion of D that is dedicated to define the S-structure. Actually, we can define D_S as the set $\{x \in D \mid \mathcal{D} \models D_S(x)\} \subseteq D$. Therefore, we must construct a V-formula $\theta'_\mathcal{S}$ such that the V-structure $\mathcal{D}$ satisfies $\theta'_\mathcal{S}$ iff the S-structure $\mathcal{D}_S$, formed by $D_S \subseteq D$ and the interpretation of the S relations, satisfies $\theta_\mathcal{S}$. That is, the S-formulas that hold in an S-structure $\mathcal{D}' \in \mathcal{S}$ must hold for the subset of a V-structure $\mathcal{D}$ defined over the part of its domain that satisfies $D_S(\cdot)$. This can be performed by inserting *guards* in the quantifiers inside $\theta_\mathcal{S}$.

Definition 5. Given a (closed) S-formula $\theta_\mathcal{S}$, $\theta'_\mathcal{S}$ is the formula resulting from applying the following substitutions to $\theta_\mathcal{S}$:

- Replace each $\exists x \varphi(x)$ in $\theta_\mathcal{S}$ by $\exists x D_S(x) \wedge \varphi(x)$;

- Replace each $\forall x \varphi(x)$ in $\theta_{\mathcal{S}}$ by $\forall x D_S(x) \rightarrow \varphi(x)$.

Finally, we can define the formula that encodes the coherence of a relational Bayesian network Φ for a class of S-structures $\mathcal{S}$:

Definition 6. For disjoint sets of relations S and R, a given relational Bayesian network Φ and a class of S-structures defined by $\theta_{\mathcal{S}}$, $\mathcal{C}_{\Phi,\mathcal{S}} = \theta'_{\mathcal{S}} \rightarrow (\mathcal{B}_{\Phi} \rightarrow \psi')$.

Putting all those arguments together, we obtain the translation of the coherence-checking problem to the validity of a formula from the transitive closure logic:

Theorem 1 (De Bona and Cozman [24])**.** *For disjoint sets of relations S and R, a given relational Bayesian network Φ and a class of S-structures $\mathcal{S}$ defined by $\theta_{\mathcal{S}}$, Φ is coherent for $\mathcal{S}$ iff $\mathcal{C}_{\Phi,\mathcal{S}}$ is valid.*

As first-order logic in general is already well-known to be undecidable, adding a transitive closure operator clearly does not make things easier. Nevertheless, decidability remains an open problem, even restricting the relations in R to be unary and assuming a decidable $\theta_{\mathcal{S}}$ (even though there are some decidable fragments of first-order logic with transitive closure operators [25,26]). Similarly, a proof of general undecidability remains elusive.

3.3. A Weaker Form of Coherence

Jaeger introduced the coherence problem for RBNs as checking whether every input structure in a given class yields a probability distribution via an acyclic ground Bayesian network. Alternatively, we might define the coherence of an RBN as the existence of at least one input structure, out of a given class, resulting in an acyclic ground Bayesian network. This is closer to the satisfiability-like notion of coherence discussed by de Finetti and closer to work on probabilistic logic [27,28].

In this section, we show that, if one is interested in a logical encoding for this type of coherence for RBNs, the transitive closure operator can be dispensed with.

Suppose we have an RBN Φ and class $\mathcal{S}$ of input structures codified via a first-order formula $\theta_{\mathcal{S}}$ and we want to decide whether Φ is coherent for some structure in $\mathcal{S}$. This problem can be reduced to checking the satisfiability of a first-order formula, using the machinery introduced above, with the bipartite domain. This formula can be easily built as $\theta'_{\mathcal{S}} \wedge \mathcal{B}_{\Phi} \wedge \psi'$. By construction, this formula is satisfiable iff there is a structure $\mathcal{D}$ over a bipartite domain $D = D_S \cup D_B$ where D_S encodes an S-structure in $\mathcal{S}$ ($\mathcal{D} \models \theta'_{\mathcal{S}}$), D_B encodes the corresponding ground Bayesian network ($\mathcal{D} \models \mathcal{B}_{\Phi}$) and the latter is acyclic ($\mathcal{D} \models \psi'$). Nonetheless, since now we are interested in satisfiability instead of validity, we can replace ψ' by a formula ψ'' that does not employ the transitive closure operator.

The idea to construct ψ'' is to use a binary relation $Parent'(\cdot,\cdot)$ and to force it to extend, or to contain, the transitive closure of $Parent(\cdot,\cdot)$. The formula ψ'' then also requires $Parent'(\cdot,\cdot)$ to be irreflexive. If there is such $Parent'(\cdot,\cdot)$, then $Parent(\cdot,\cdot)$ must be acyclic. Conversely, if $Parent(\cdot,\cdot)$ is acyclic, then $Parent'(\cdot,\cdot)$ can be interpreted as its transitive closure, being irreflexive. In other words, we want a structure to satisfy ψ'' iff it interprets a relation $Parent'(\cdot,\cdot)$ that both is irreflexive and extends the transitive closure of $Parent(\cdot,\cdot)$.

In order to build ψ', the vocabulary V is augmented with the binary relation $Parent'$. Now, we can define ψ'' as the conjunction of two parts:

- $\forall x \forall y \forall z \big(Parent(x,y) \rightarrow Parent'(x,y)\big) \wedge \big(Parent'(x,y) \wedge Parent'(y,z) \rightarrow Parent'(x,z)\big)$, forcing $Parent'$ to extend the transitive closure of $Parent$;
- $\forall x \neg Parent'(x,x)$, requiring $Parent'$ to be irreflexive.

By construction, one can verify the following result:

Theorem 2. *For disjoint sets of relations S and R, a given relational Bayesian network* Φ *and a class of S-structures* $\mathcal{S}$ *defined by* $\theta_{\mathcal{S}}$, Φ *is coherent for some structure in* $\mathcal{S}$ *iff* $\theta'_{\mathcal{S}} \wedge \mathcal{B}_{\Phi} \wedge \psi''$ *is satisfiable.*

The fact that $\theta'_{\mathcal{S}} \wedge \mathcal{B}_{\Phi} \wedge \psi''$ does not use the transitive closure operator makes its satisfiability decidable for any decidable fragment of first-order logic.

4. Probabilistic Relational Models

In this section, we introduce the machinery of PRMs by following the terminology by Getoor et al. [19], focusing on the simple case where uncertainty is restricted to descriptive attributes, which are assumed to be binary. We also review the coherence problem for PRMs and the proposed solutions in the literature. In the next section, we show how this coherence problem can also be tackled via logic, as the coherence of RBNs.

4.1. Syntax and Semantics of PRMs

To define a PRM, illustrated in Example 5, we need a relational model, with classes associated with descriptive attributes and reference slots that behave like foreign keys. Intuitively, each object in a class is described by the values of its descriptive attributes, and reference slots link different objects. Formally, a relational schema is described by a set of classes $\mathcal{X} = \{X_1, \ldots, X_n\}$, each of which associated with a set of *descriptive attributes* $\mathcal{A}(X_i)$ and a set of *reference slots* $\mathcal{R}(X_i)$. We assume descriptive attributes take values in $\{0, 1\}$. A reference slot ρ in a class X (denoted $X.\rho$) is a reference to an object of the class $\mathsf{Range}[\rho]$ (its *range type*) specified in the schema. The *domain type* of ρ, $\mathsf{Dom}[\rho]$, is X. We can view this reference slot ρ as a function f_ρ taking objects in $\mathsf{Dom}[\rho]$ and returning singletons of objects in $\mathsf{Range}[\rho]$. That is, $f_\rho(x) = \{y\}$ is equivalent to $x.\rho = y$.

For any reference slot ρ, there is an *inverse slot* ρ^{-1} such that $\mathsf{Range}[\rho^{-1}] = \mathsf{Dom}[\rho]$ and $\mathsf{Dom}[\rho^{-1}] = \mathsf{Range}[\rho]$. The corresponding function, $f_{\rho^{-1}}$ takes an object x from the class $\mathsf{Range}[\rho]$ and returns the set of objects $\{y \mid f_\rho(y) = \{x\}\}$ from the class $\mathsf{Dom}[\rho]$. A sequence of slots (inverted or not) $K = \rho_1, \ldots, \rho_k$ is called a *slot chain* if $\mathsf{Range}[\rho_i] = \mathsf{Dom}[\rho_{i+1}]$ for all i. The function corresponding to a slot chain $K = \rho_1, \rho_2$, f_K, is a type of composition of the functions f_{ρ_1}, f_{ρ_2}, taking an object x from $\mathsf{Range}[\rho_1]$ and returning a set objects $\{z \mid \exists y : y \in f_{\rho_1}(x) \wedge z \in f_{\rho_2}(y)\}$ from $\mathsf{Range}[\rho_2]$. The corresponding function can be obtained by applying this type of composition two-by-two. We write $y \in x.K$ when $y \in f_K(x)$.

An instance $\mathcal{I}$ of a relational schema populates the classes with objects, associating values with the descriptive attributes and reference slots. Formally, $\mathcal{I}$ is an interpretation specifying for each class $X \in \mathcal{X}$: a set of objects $\mathcal{I}(X)$; a value $A.x \in \{0, 1\}$ for each descriptive attribute in $A \in \mathcal{A}(X)$ and each object $x \in \mathcal{I}(X)$; and an object $x.\rho \in \mathcal{I}(\mathsf{Range}[\rho])$ for each reference slot $\rho \in \mathcal{R}(X)$ and object $x \in \mathcal{I}(X)$. Note that, if $x.\rho = y$, $f_\rho(x) = \{y\}$. We use $\mathcal{I}_{x.A}$ and $\mathcal{I}_{x.\rho}$ to denote the value of $x.A$ and $x.\rho$ in $\mathcal{I}$.

Given a relational schema, a PRM defines a probability distribution over its instances. In the simplest form, on which we focus, objects and the relations between them are given as input, and there is uncertainty only over the descriptive attributes values. A *relational skeleton* σ^r is a partial specification of an instance that specifies a set of objects $\sigma^r(X_i)$ for each class X_i in the schema besides the relation holding between these objects: ${\sigma^r}_{x.\rho}$ for each $x \in \sigma^r(X_i)$ and $\rho \in \mathcal{R}(X_i)$. A *completion* of a relational skeleton σ^r is an instance $\mathcal{I}$ such that, for each class $X_i \in \mathcal{X}$: $\mathcal{I}(X_i) = \sigma^r(X_i)$ and, for each $x \in \mathcal{I}(X_i)$ and $\rho \in \mathcal{R}(X_i)$, $\mathcal{I}_{x.\rho} = {\sigma^r}_{x.\rho}$. We can see a PRM as a function taking relational skeletons and returning probability distributions over the completions of these partial instances, which can be seen as joint probability distributions for the random variables formed by the descriptive attributes of each object.

The format of a PRM resembles that of a Bayesian network: for each attribute $X.A$, we have a set of parents $\mathtt{Pa}(X.A)$ and the corresponding parameters $P(X.A \mid \mathtt{Pa}(X.A))$. The parent relation forms a direct graph, as usual, called the *dependency graph*; and the set of parameters define the conditional probability tables. The attributes in $\mathtt{Pa}(X.A)$ are called *formal parents*, as they will be instantiated for each object x in X according to the relational skeleton. There two types of formal parents: $X.A$ can

depend either on another attribute $X.B$ of the same object or on an attribute $X.K.B$ of other objects, where K is a slot chain.

In general, for an object x, $x.K.B$ is a multiset $\{y.B \mid y \in x.K\}$, whose size is defined by the relational skeleton. To compactly represent the conditional probability distribution when $X.K.B \in \mathtt{Pa}(X.A)$, the notion of *aggregation* is used. The attribute $x.A$ will depend on some aggregate function γ of this multiset, like its mean value, mode, maximum or minimum, and so on; that is, $\gamma(X.K.B)$ will be a formal parent of $X.A$.

Definition 7. A *probabilistic Relational model* Π for a relational schema $\mathcal{R}$ is defined as a pair $\langle \Pi_{\mathcal{S}}, \Pi_{\theta} \rangle$ where:

- $\Pi_{\mathcal{S}}$ defines, for each class $X \in \mathcal{X}$ and each descriptive attribute $A \in \mathcal{A}(X)$, a set of formal parents $\mathtt{Pa}(X.A) = \{U_1, \ldots, U_l\}$, where each U_i has the form $X.B$ or $\gamma(X.K.B)$;
- Π_{θ} is the set of parameters defining legal Conditional Probability Distributions (CPDs) $P(X.A \mid \mathtt{Pa}(X.A))$ for each descriptive attribute $A \in \mathcal{A}(X)$ of each class $X \in \mathcal{X}$.

The semantics of a PRM is given by the *ground* Bayesian network induced by a relational skeleton, where the descriptive attributes of each object are the random variables.

Definition 8. A PRM $\Pi = \langle \Pi_{\mathcal{S}}, \Pi_{\theta} \rangle$ and a relational skeleton σ^r define a ground Bayesian network where:

- There is a node representing each attribute $x.A$, for all $x \in \sigma^r(X_i)$, $A \in \mathcal{A}(X_i)$ and $X_i \in \mathcal{X}$;
- For each $X_i \in \mathcal{X}$, each $x \in \sigma^r(X_i)$ and each $A \in \mathcal{A}(X_i)$, there is a node representing $\gamma(x.K.B)$ for each $\gamma(X_i.K.B) \in \mathtt{Pa}(X_i.A)$;
- Each $x.A$ depends on parents $x.B$, for formal parents $X.B \in \mathtt{Pa}(X.A)$, and on parents $\gamma(x.K.B)$, for formal parents $\gamma(X.K.B) \in \mathtt{Pa}(X.A)$, according to $\Pi_{\mathcal{S}}$;
- Each $\gamma(x.K.B)$ depends on parents $y.B$ with $y \in x.K$;
- The CPD for $P(x.A \mid \mathtt{Pa}(x.A))$ is $P(X.A \mid \mathtt{Pa}(X.A))$, according to Π_{θ}.
- The CPD for $P(\gamma(x.K.B) \mid \mathtt{Pa}(\gamma(x.K.B)))$ is computed through the aggregation function γ.

The joint probability distribution over the descriptive attributes can be factored as usual to compute the probability of a specific instance $\mathcal{I}$ that is a completion of the skeleton σ^r. If we delete each $\gamma(x.K.B)$ from the ground Bayesian network, making its children depend directly on the nodes $y.B$ with $y \in x.K$ (defining a new parent relation $\mathtt{Pa}'$) and updating the CPDs accordingly, we can construct a *simplified* ground Bayesian network. The latter can be employed to factor the joint probability distribution over the descriptive attributes:

$$\begin{aligned} P(\mathcal{I} \mid \sigma^r, \Pi) &= \prod_{x \in \sigma^r} \prod_{A \in \mathcal{A}(x)} P(\mathcal{I}_{x.A} \mid \mathcal{I}_{\mathtt{Pa}'(x.A)}) \\ &= \prod_{X_i} \prod_{x \in \sigma^r(X_i)} \prod_{A \in \mathcal{A}(x)} P(\mathcal{I}_{x.A} \mid \mathcal{I}_{\mathtt{Pa}'(x.A)}). \end{aligned}$$

Viewing Π as a function from skeletons to probability distributions over instances, we use $\Pi(\sigma^r)$ to denote the probability distribution $P(\mathcal{I} \mid \sigma^r, \Pi)$ over the completions $\mathcal{I}$ of σ^r.

Example 5. *Recall again Scenario 4 in Example 2. We can define a PRM that returns the corresponding Bayesian network for each number and configuration of neighbors. In our relational schema, we have a class* **Person**, *whose set of descriptive attributes is* $A(\mathbf{Person}) = \{\mathsf{burglary}, \mathsf{alarm}, \mathsf{calls}\}$. *Furthermore, to capture multiple neighbors, we also need a class* **Neighborhood**, *with two reference slots,* $\mathcal{R}(\mathbf{Neighborhood}) = \{\mathsf{neighbor1}, \mathsf{neighbor2}\}$, *whose domain is* **Person**. *For instance, to denote that* Alice *and* Bob *are neighbors, we would have an object, say* nAB, *in the class* **Neighborhood**, *whose reference slots would be* nAB.neighbor1 = Alice *and* nAB.neighbor2 = Bob.

We assume that the relation neighbor *is reflexive (that is, for each* **Person** x*, there is always a* **Neighborhood** n_x *with* n_x.neighbor1 $=$ n_x.neighbor2 $=$ x*) and symmetrical (if* $x \in y$.neighbor1^{-1}.neighbor2*, we also have* $y \in x$.neighbor1^{-1}.neighbor2*).*

For each descriptive attribute in our relational schema, we associate a set of formal parents and a conditional probability table, forming the following PRM Π *to encode Scenario 4:*

- $\mathsf{Pa}(\mathbf{Person}.\mathsf{burglary}) = \varnothing$; $P(\mathbf{Person}.\mathsf{burglary}) = 0.001$;
- $\mathsf{Pa}(\mathbf{Person}.\mathsf{alarm}) = \{\mathsf{burglary}\}$; $P(\mathbf{Person}.\mathsf{alarm} \mid \mathsf{burglary}) = 0.9$ *and* $P(\mathbf{Person}.\mathsf{alarm} \mid \neg\mathsf{burglary}) = 0.1$;
- $\mathsf{Pa}(\mathbf{Person}.\mathsf{calls}) = \{\mathsf{or}(\mathbf{Person}.\mathsf{neighbor1}^{-1}.\mathsf{neighbor2})\}$;
  $P(\mathbf{Person}.\mathsf{calls} \mid \mathsf{or}(\mathbf{Person}.\mathsf{neighbor1}^{-1}.\mathsf{neighbor2}) = c) = c$*, for* $c \in \{0, 1\}$.

Given a relational skeleton σ^r *with persons and neighbors,* Π *determines a joint probability distribution over the the descriptive attributes, via a Bayesian network. Consider a skeleton* σ^r *with* $\sigma^r(\mathbf{Person}) = \{x_1, x_2, x_3\}$ *and* $n12, n23 \in \sigma^r(\mathbf{Neighborhood})$*, with* ${\sigma^r}_{nij}.\mathsf{neighbor1} = x_i$ *and* ${\sigma^r}_{nij}.\mathsf{neighbor2} = x_j$*, for each* $nij \in \sigma^r(\mathbf{Neighborhood})$*, but such that no* $n \in \sigma^r(\mathbf{Neighborhood})$ *has* $n.\mathsf{neighbor1} = x_1$ *and* $n.\mathsf{neighbor2} = x_3$*. Then, the resulting probability distribution is the model of Scenario 3 in Example 2, whose Bayesian network is given in Figure 2.*

4.2. Coherence via Colored Dependency Graphs

As with RBNs, for the model to be coherent, one needs to guarantee that the ground Bayesian network is acyclic. Getoor et al. [19] focused on guaranteeing that a PRM yields acyclic ground Bayesian networks for *all* possible relational skeletons. To achieve that, possible cycles are detected in the class dependency graph.

Definition 9. Given a PRM Π, the *class dependency graph* G_Π is a directed graph with a node for each descriptive attribute $X.A$ and the following arcs:

- Type I arcs: $\langle X.B, X.A \rangle$, where $X.B$ is a formal parent of $X.A$;
- Type II arcs: $\langle Y.B, X.A \rangle$, where $\gamma(X.K.B)$ is a formal parent of $X.A$ and $Y = Range[X.K]$.

When the class dependency graph is acyclic, so is the ground Bayesian network for any relational skeleton. Nevertheless, it may be the case that, even for cyclic class dependency graphs, any relational skeleton occurring in practice leads to a coherent model. In other words, there might be classes of skeletons for which the PRM is coherent. To easily recognize some of these classes, Getoor et al. [19] put forward an approach based on identifying slot chains that are acyclic in practice. A set of slot chains $K_{ga} = \{K_1, \ldots, K_m\}$ is *guaranteed acyclic* if we are guaranteed that, for any possible relational skeleton σ^r, there is a partial ordering $\preceq$ over its objects such that, for each $K_i \in K_{ga}$, $x \prec y$ for any pair x and $y \in x.K_i$ (we use $x \prec y$ to denote $x \preceq y$ and $x \neq y$).

Definition 10. Given a PRM Π and a set of guaranteed acyclic slot chains K_{ga}, the *colored class dependency graph* G_Π is a directed graph with a node for each descriptive attribute $X.A$ and the following arcs:

- Yellow arcs: $\langle X.B, X.A \rangle$, where $X.B$ is a formal parent of $X.A$;
- Green arcs: $\langle Y.B, X.A \rangle$, where $\gamma(X.K.B)$ is a formal parent of $X.A$, $Y = Range[X.K]$ and $K \in K_{ga}$;
- Red arcs: $\langle Y.B, X.A \rangle$, where $\gamma(X.K.B)$ is a formal parent of $X.A$, $Y = Range[X.K]$ and $K \notin K_{ga}$.

Intuitively, yellow cycles in the colored class dependency graph correspond to attributes of the same object, yielding a cycle in the ground Bayesian network. If we add some green arcs to such a cycle, then it is guaranteed that, departing from a node $x.A$ in the ground Bayesian network, these arcs form a path to $y.A$, where $x \prec y$, since $\preceq$ is transitive. Hence, x is different from y, and there is no cycle. If there is a red arc in a cycle, however, one may have a skeleton that produces a cycle.

A colored class dependency graph is *stratified* if every cycle contains at least one green arc and no red arc. Then:

Theorem 3 (Getoor et al. [19]). *Given a PRM* Π *and a set of guaranteed acyclic slot chains* K_{ga}*, if the colored class dependency graph* G_Π *is stratified, then the ground Bayesian network is acyclic for any possible relational skeleton.*

In the result above and in the definition of guaranteed acyclic slot chains, "possible relational skeleton" refers to the class of skeletons that can occur in practice. The user must detect the guaranteed acyclic slot chains, taking advantage of his *a priori* knowledge on the possible skeletons in practice. For instance, consider a slot chain motherOf linking objects of the same class **Person** (Example 3). A genetic attribute, like **Person**.blueEyes, might depend on **Person**.motherOf.blueEyes. Mathematically, we can conceive of a skeleton with a cyclic relation motherOf, resulting in a red cycle in the colored class dependency graph. Nonetheless, being aware of the intended meaning of motherOf, we know that such skeletons are not possible in practice, so the cycle is green, and coherence is guaranteed.

Identifying guaranteed acyclic slot chains is by no means trivial. In fact, Getoor et al. [19] also define guaranteed acyclic (g.a.) reference slots and g.a. slot chains are defined as those formed only by g.a. reference slots. Still, these maneuvers miss the cases where two possible reference slots cannot be g.a. according to the same $\preceq$, but combine to form a g.a. slot chain. Getoor et al. [19] mention the possibility of assuming different partial orders to define different sets of g.a. slot chains: in that case, each ordering would correspond to a shade of green in the colored class dependency graph, and coherence would not be ensured if there were two shades of green in a cycle.

5. Logic-Based Approach to the Coherence of PRMs

The simplest approach to the coherence of PRMs, via the non-colored class dependency graph, is intrinsically incomplete, in the sense that some skeletons might yield a coherent ground Bayesian network even for cyclic graphs. The approach via colored class dependency graph allows some cyclic graphs (the stratified ones) to guarantee consistency for the class of all possible skeletons. However, this method depends on a pre-specified set of guaranteed acyclic slot chains, and the colored class dependency graph being stratified for this set is only a sufficient, not a necessary condition for coherence. Therefore, the colored class dependency graph method is incomplete, as well. Even using different sets of g.a. slot chains (corresponding to shades of green) to eventually capture all of them, it is still possible that a cycle with red arcs cannot entail incoherence in practice. Besides being incomplete, the graph-based method is not easily applicable to an arbitrary class of skeletons. Given a class of skeletons as input, the user would have to detect somehow which slot chains are guaranteed acyclic for that specific class; this can be considerably more difficult than ensuring acyclicity in the general case.

To address these issues, thus obtaining a general, complete method for checking the coherence of PRMs for a given class of skeletons, we can resort to the logic-based approach we introduced for the RBNs in previous sections. The goal of this section is to adapt the logic-basic techniques to PRMs.

PRMs can be viewed as RBNs, as conditional probability tables of the former can be embedded into combination functions of the latter. This translation is out of our scope though, and it suffices for our purposes to represent PRMs as random relational structures, taking S-structures to probability distributions on R-structures. While the S-vocabulary is used to specify classes of objects and relations between them (that is, the relational skeleton), the R-vocabulary expresses the descriptive attributes of the objects. Employing this logical encoding of PRMs, we can apply the approach from Section 3.1 to the coherence problem for PRMs.

To follow this strategy, we first show how a PRM can be seen as a random relational structure described by a logical language.

5.1. PRMs as Random Relational Structures

Consider a PRM $\Pi = \langle \Pi_S, \Pi_\theta \rangle$ over a relational schema described by a set of classes $\mathcal{X} = \{X_1, \ldots X_n\}$, each associated with a set of descriptive attributes $\mathcal{A}(X_i)$ and a set of reference slots $\mathcal{R}(X_i)$. Given a skeleton σ^r, which is formed by objects and relations holding between them, the PRM Π yields a ground Bayesian network over the descriptive attributes of these objects, defining a probability distribution $\Pi(\sigma^r)$ over the completions of σ^r. Hence, if the relational skeleton is given as a first-order S-structure over a set of objects and a set of unary relations R denotes their attributes, the PRM becomes a random relational structure.

We need to represent a skeleton σ^r as a first-order S-structure Σ. Objects in σ^r can be seen as the elements of the domain D of Σ. Note that PRMs are typed, with each object belonging to specific class $X_i \in X$. Thus, we use unary relations $X_1, \ldots, X_n$ in the vocabulary S to denote the class of each object. Accordingly, for each $x \in D$, $X_i(x)$ holds in Σ iff $x \in \sigma^r(X_i)$. As each object belongs to exactly one class in the relational skeleton, the class of possible first-order structures is restricted to those where the relations $X_1, \ldots, X_n$ form a partition of the domain.

The first-order S-structure Σ must also encode the relations holding between the objects in the skeleton that are specified via the values of the reference slots. To capture these, we assume they have unique names and consider, for each reference slot $X_i.\rho \in \mathcal{R}(X_i)$ with $Range[\rho] = X_j$, a binary relation S^ρ. In Σ, $S^\rho(x, y)$ holds iff ${\sigma^r}_{x.\rho} = y$. Naturally, $S^\rho(x, y)$ should imply $X_i(x)$ and $X_j(y)$. Now, Σ encodes, through the vocabulary S, all objects of a given class, as well as the relations between them specified in the reference slots. In other words, there is a computable function b_S from relational skeletons σ^r to S-structures $\Sigma = b_S(\sigma^r)$. For b_S to be a bijection, we make its codomain equal to its range.

The probabilistic vocabulary of the random relational structure corresponding to a PRM is formed by the descriptive attributes of every class in the relational schema. We assume that attributes in different classes have different names, as well, in order to define the vocabulary of unary relations $R = \{A \in \mathcal{A}(X_i) \mid X_i \in \mathcal{X}\}$. If A_j is an attribute of X_i, $x.A_j = 1$ (resp. $x.A_j = 0$) in the PRM is mirrored by the ground R-atom $A_j(x)$ being true (resp. false) in the random relational structure. Thus, as a completion $\mathcal{I}$ corresponds to a value assignment to descriptive attributes of objects $x_1, \ldots, x_m$ from a relational skeleton σ^r, it also corresponds to an R-structure $\mathcal{D}_\mathcal{I}$ over a domain $D = \{x_1, \ldots, x_m\}$ in the following way: $\mathcal{D}_\mathcal{I} \models A_i(x_j)$ iff $x_j.A_i = 1$. Note that we assume that for $\mathcal{D}_\mathcal{I}$ to correspond to a completion $\mathcal{I}$ of σ^r, $\mathcal{D}_\mathcal{I} \not\models A_i(x_j)$ whenever A_i is not an attribute of the class $X \in \mathcal{X}$ such that $x_j \in \sigma^r(X)$. Let b_R denote the function taking instances $\mathcal{I}$ and returning the corresponding R-structures $\mathcal{D}_\mathcal{I} = b_R(\mathcal{I})$. As we cannot recover the skeleton σ^r from the R-structure $\mathcal{D}_\mathcal{I} = b_R(\mathcal{I})$, b_R is not a bijection. Nevertheless, fixing a skeleton σ^r, there is a unique $\mathcal{I}$ such that $b_R(\mathcal{I}) = \mathcal{D}_\mathcal{I}$.

Now, we can define a random relational structure P_Π that corresponds to the PRM Π. For every relational skeleton σ^r over a domain D, let $P_\Pi(b_S(\sigma^r)) : \mathsf{Mod}_R(D) \to [0, 1]$ be a probability distribution over R-structures such that $P_\Pi(b_S(\sigma^r))(\mathcal{D}_R) = \Pi(\sigma^r)(\mathcal{I}_\mathcal{R})$, if $\mathcal{D}_R = b_R(\mathcal{I}_\mathcal{R})$, for a completion $\mathcal{I}_R$ of σ^r, and $P_\Pi(b_S)(\mathcal{D}_R) = 0$ otherwise.

5.2. Encoding the Ground Bayesian Network and its Acyclicity

The probability distribution $P_\Pi(b_S(\sigma^r))$ can be represented by a ground Bayesian network $B_{P_\Pi}(b_S(\sigma^r))$, where nodes represent the ground R-atoms. The structure of this network is isomorphic to the simplified ground Bayesian network yielded by Π for the skeleton σ^r, if we ignore the isolated nodes representing the spurious $A_i(x_j) = 0$, when A_i is not an attribute of the class to which x_j belongs. The coherence of $\Pi(\sigma^r)$ depends on the acyclicity of the corresponding ground Bayesian network $B_\Pi(\sigma^r)$, which is acyclic iff $B_{P_\Pi}(\sigma^r)$ is so. Therefore, we can encode the coherence of a PRM Π for a skeleton σ^r via the acyclicity of $B_{P_\Pi}(b_s(\sigma^r))$ by applying the techniques from Section 3.

We want to construct a formula that is satisfied only by those S-structures $b_S(\sigma^r)$ such that $\Pi(\sigma^r)$ is coherent. Again, we consider an extended, bipartite domain $D = D_S \cup D_B$, with $b_S(\sigma^r)$ encoded over D_S and the structure of $B_{P_\Pi}(\sigma^r)$ encoded in D_B. We want to build a formula $\mathcal{B}_\Pi$ that is satisfied by

structures $\mathcal{D}$ over $D = D_S \cup D_B$ such that, if $\mathcal{D}$ encodes $b_S(\sigma^r)$ over D_S, then $\mathcal{D}$ encodes the structure of $B_{P_\Pi}(b_S(\sigma^r))$ over D_B. The nodes are encoded exactly as shown in Section 3.1.

To encode the arcs, we employ once more a relation $Parent(\cdot,\cdot)$. $Parent(y,y')$ must hold only if $x, y \in D_B$ denote ground R-atoms $A_i(x)$ and $A_j(x')$ such that $x'.A_j \in \mathtt{Pa}'(x.A_i)$ in the simplified ground Bayesian network, which is captured by the formula $Dep_i^j(x,x')$, as in Section 3.1. The only difference here is that now $Dep_i^j(x,x')$ can be defined directly. We use $Dep_i^j(x,x')$ here to denote a formula recursively defined, not an atom over the binary relation $Dep_i^j(\cdot,\cdot)$. For each pair $A_i, A_j \in R$, we can simply look at Π_S to see the conditions on which $x'.A_j$ is a parent of $x.A_i$ in the simplified ground Bayesian network ($x'.A_j \in \mathtt{Pa}'(x.A_i)$), in which case $A_j(x')$ will be a parent of $A_i(x)$ in $B_{P_\pi}(b_S(\sigma^r))$. If $X.A_j \in \mathtt{Pa}(X.A_i)$, then $Dep_i^j(x,x')$ should be true whenever $x = x'$. If $\gamma(X.K.A_j) \in \mathtt{Pa}(X.A_i)$, for a slot chain K, then $x.K.A_j \subseteq \mathtt{Pa}'(x.A_i)$ and $Dep_i^j(x,x')$ should be true whenever x' is related to x via $K = \rho_1, \ldots, \rho_k$. This is the case if:

$$\exists y_1, \exists y_2 \ldots \exists y_{k-1} S^{\rho_1}(x,y_1) \wedge S^{\rho_2}(y_1,y_2) \wedge \cdots \wedge S^{\rho_k}(y_{k-1},x')$$

is true. If $K = \rho$, this formula is simply $S^\rho(x,x')$.

Note that it is possible that both $X.A_j$ and $\gamma(X.K.A_j)$ are formal parents of $X.A_i$, and there can even be different parents $\gamma(X.K.A_j)$, for different K. Thus, we define $Dep_i^j(x,x')$ algorithmically. Initially, make $Dep_i^j(x,x') = \bot$. If $X.A_j \in \mathtt{Pa}(X.A_i)$ for some A_j, make $Dep_i^j(x,x') = Dep_i^j(x,x') \vee (x = x')$. Finally, for each $\gamma(X.K.A_j)$ in $\mathtt{Pa}(X.A_i)$, for a slot chain $K = \rho_1, \ldots, \rho_k$, make:

$$Dep_i^j(x,x') = Dep_i^j(x,x') \vee \exists y_1, \exists y_2 \ldots \exists y_{k-1} S^{\rho_1}(x,y_1) \wedge S^{\rho_2}(y_1,y_2) \wedge \cdots \wedge S^{\rho_k}(y_{k-1},x'),$$

using fresh $y_1, \ldots, y_{k-1}$.

Analogously to Section 3.1, we have a formula $\mathcal{B}_\Pi$, for a fixed PRM Π, that is satisfied only by structures $\mathcal{D}$ over a bipartite domain $D_S \cup D_B$ where the $Parent(\cdot,\cdot)$ relation over D_B brings the structure of the ground Bayesian network $B_{P_\Pi}(\Sigma)$ corresponding to the S-structure Σ encoded in D_S. Again, acyclicity can be captured via a transitive closure operator: $\psi_\Pi = \forall x \neg \mathsf{TC}(Parent)(x,x)$. The PRM Π is coherent for a skeleton σ^r if, for every structure $\mathcal{D}$ over a bipartite domain $D_S \cup D_B$ encoding $b_S(\sigma^r)$ in D_S, we have $\mathcal{D} \models \mathcal{B}_\Pi \rightarrow \psi_\Phi$.

Consider now a class $\mathcal{S}$ of skeletons σ^r such that $\{b_S(\sigma^r) \mid \sigma^r \in \mathcal{S}\}$ is the set of S-structures satisfying a first-order formula $\theta_\mathcal{S}$. To check whether the PRM Π is coherent for the class $\mathcal{S}$, we construct $\theta'_\mathcal{S}$ by inserting guards to the quantifiers, as explained in Definition 5. Finally, the PRM Π is coherent for a class $\mathcal{S}$ of relational skeletons iff $\theta'_\mathcal{S} \rightarrow (\mathcal{B}_\Pi \rightarrow \psi_\Pi)$ is valid.

We have thus succeeded in turning coherence checking for PRMs into a logical inference, by adapting techniques we developed for RBNs. In the next section, we travel, in a sense, the reverse route: we show how to adapt the existing graph-based techniques for coherence checking of PRMs to coherence checking of RBNs.

6. Graph-Based Approach to the Coherence of RBNs

The logic-based approach to the coherence problem for RBNs can be applied to an arbitrary class of input structures, as long as the class can be described by a first-order formula, possibly with a transitive closure operator. Given any class of input structures $\mathcal{S}$, via the formula $\theta_\mathcal{S}$, we can verify the coherence of a RBN Φ via the validity of $\theta'_\mathcal{S} \rightarrow (\mathcal{B}_\Phi \rightarrow \psi')$, as explained in Section 3. Furthermore, this method is complete, as Φ is coherent for $\mathcal{S}$ if and *only if* such a formula is valid. Nonetheless, completeness and flexibility regarding the input class come at a very high price, as deciding the validity of this first-order formula involving a transitive closure operator may be computationally hard, if at all decidable. Therefore, RBNs users can benefit from the ideas introduced by Getoor et al. [19] for the coherence of PRMs, using the (colored) dependency graphs. While Jaeger [23] proposes to

investigate the coherence for a given class of models described by a logical formula, Getoor et al. [19] are interested in a single class of inputs: the skeletons that are possible in practice. With *a priori* knowledge, the RBN user perhaps can attest to the acyclicity of the resulting ground Bayesian network for all possible inputs.

Any arc $\langle r'(t'), r(t)\rangle$ in the output ground Bayesian network $B_\Phi(\mathcal{D}_S)$, for an RBN Φ and input $\mathcal{D}_S$ reflects that the probability formula $F_r(x)$, when $x = t$, depends on $r(t')$. Hence, possible arcs in this network can be anticipated by looking into the probability formulas $F_r(x)$, for the probabilistic relations $r \in R$, in the definition of Φ. In other words, by inspecting the probability formula $F_r(x)$, we can detect those $r' \in R$ for which an arc $\langle r'(t'), r(t)\rangle$ can possibly occur in the ground Bayesian network. Similarly to the class dependency graph for PRMs, we can construct a high-level dependency graph for RBNs that brings the possible arcs, and thus cycles, in the ground Bayesian network.

Definition 11. Given an RBN Φ, the *R-dependency graph* G_Φ is a directed graph with a node for each probabilistic relation $r \in R$ and the following arcs:

- Type I arcs: $\langle r', r\rangle$, where $r'(x)$ occurs in $F_r(x)$ outside the scope of a combination function;
- Type II arcs: $\langle r', r\rangle$, where $r'(y)$ occurs in $F_r(x)$ inside the scope of a combination function.

Intuitively, a Type I arc $\langle r', r\rangle$ in the R-dependency graph of an RBN Φ means that, for any input structure $\mathcal{D}_S$ over D and any tuple $t \in D^{a(r)}$, $F_r(t)$ depends on $r'(t)$ in the ground Bayesian network $B_\Phi(\mathcal{D}_S)$; formally, $r'(t) \in \alpha(F_r(x), t, \mathcal{D}_S)$. For instance, if $F_{r_1}(x) = \mathsf{mean}(\{|r_2(y) \mid y; S(x,y)|\})(1 - r_3(x))$, then, given any S-structure, $F_{r_1}(t)$ depends on $r_3(t)$ for any t. Type II arcs capture dependencies that are contingent on the S-relations holding in the input structure. In other words, a Type II arc $\langle r', r\rangle$ means that $F_r(t)$ will depend on $r'(t')$ if some S-formula φ holds in the input structure $\mathcal{D}_S$; $\mathcal{D}_S \models \varphi$. For instance, if $F_{r_1}(x) = (\mathsf{mean}(\{|r_2(y) \mid y; S(x,y)|\})(1 - r_3(x))$, $r_1(t)$ depends on $r_2(t')$ (for t, t' in the domain D) in the output ground Bayesian network iff the input $\mathcal{D}_S$ is such that $\mathcal{D}_S \models S(t, t')$. If combination functions are nested, the corresponding S-formula might be fairly complicated. Nevertheless, the point here is simply noting that, given a Type II arc $\langle r', r\rangle$, the conditions on which $r(t)$ is actually a child of $r'(t)$ in the ground Bayesian network can be expressed with an S-formula parametrized by t, t', which will be denoted by $\varphi_S^{r,r'}(t, t')$. Consequently, for $t, t' \in D$, $\mathcal{D}_S \models \varphi_S^{r,r'}(t, t')$ iff $r'(t') \in \alpha(F_r(x), t, \mathcal{D}_S)$, i.e., $r(t)$ depends on $r'(t')$ in $B_\Phi(\mathcal{D}_S)$.

As each arc in the ground Bayesian network corresponds to an arc on the R-dependency graph, when the latter is acyclic, so will be the former, for any input structure. As it happens with class dependency graphs and PRMs, though, a cycle in the R-dependency graph does not entail a cycle in the ground Bayesian network if a Type II arc is involved. It might well be the case that the input structures $\mathcal{D}_S$ found in practice do not cause cycles to occur. This can be captured via a colored version of the R-dependency graph.

In the same way that Type I arcs in the class dependency graph of a PRM relate to attributes of different objects, in the R-dependency graph of an RBN, these arcs encode the dependency between relations $r, r' \in R$ to be grounded with (possibly) different tuples. For a PRM, the ground Bayesian network can never reflect a cycle with green arcs, but no red one, in the class dependency graph, for a sequence of green arcs guarantees different objects, according to a partial ordering. Analogously, with domain knowledge, the user can identify Type II arcs in the R-dependency graph whose sequence will prevent cycles in the ground Bayesian network, via a partial ordering over the tuples.

For a vocabulary S of predefined relations, let $T^D = \bigcup\{D^a \mid a \in \mathbb{N}\}$ denote the set of all tuples with elements of D. We say a set $A_{ga} = \{\langle r'_i, r_i\rangle \mid 1 \leq i \leq n\}$ of Type II arcs is guaranteed acyclic if, for any possible input structure $\mathcal{D}_S$ over D, there is a partial ordering $\preceq$ over T^D such that, if $\mathcal{D}_S \models \varphi_S^{r,r'}(t, t')$ for some $t, t' \in T^D$, then $t \prec t'$. Here, again, "possible" means "possible in practice".

Definition 12. Given the R-dependency graph of an RBN Φ and a set A_{ga} of guaranteed acyclic type II arcs, the *colored R-dependency graph* G_{Π} is a directed graph with a node for each $r \in R$ and the followin arcs:

- Yellow arcs: Type I arcs in the R-dependency graph;
- Green arcs: Type II arcs $\langle r', r \rangle$ in the R-dependency graph such that $\langle r', r \rangle \in A_{ga}$;
- Red arcs: The remaining (Type II) arcs in the R-dependency graph.

Again, yellow cycles in the colored R-dependency graph correspond to relations $r \in R$ grounded with the same tuple t, yielding a cycle in the ground Bayesian network. If green arcs are added to a cycle, then it is guaranteed that, departing from a node $r(t)$ in the ground Bayesian network, these arcs form a path to $r(t')$, where $t \prec t'$ for a partial ordering $\preceq$, and there is no cycle. Once more, red arcs in cycles may cause $t = t'$, and coherence is not ensured. Calling *stratified* a R-dependency graph whose every cycle contains at least one green arc and no red arc, we have:

Theorem 4. *Given the R-dependency graph of an RBN Φ and a set A_{ga} of guaranteed acyclic Type II arcs, if the colored class dependency graph G_{Φ} is stratified, then the ground Bayesian network is acyclic for any possible input structure.*

Of course, detecting guaranteed acyclic Type II arcs in R-dependency graphs of RBNs is even harder than, as a generalization of, detecting guaranteed acyclic slot chains in PRMs. In any case, if the involved relations $r, r' \in R$ are unary, one is in a position similar to finding acyclic slot chains, as the arguments of r, r' can be seen as objects, and only a partial ordering over the elements of the domain (not tuples) is needed.

7. Conclusions

In this paper, we examined a new version of *coherence checking*, a central problem in the foundations of probability as conceived by de Finetti. The simplest formulation of coherence checking takes a set of events and their probabilities and asks whether there can be a probability measure over an appropriate sample space [1]. This sort of problem is akin to inference in propositional probabilistic logic [28]. Unsurprisingly, similar inference problems have been studied in connection with first-order probabilistic logic [27]. Our focus here is on coherence checking when one has events specified by first-order expressions, on top of which one has probability values and independence relations. Due to the hopeless complexity of handling coherence checking for any possible set of assessments and independence judgments, we focus on those specifications that enhance the popular language of Bayesian networks. In doing so, we address a coherence checking problem that was discussed in the pioneering work by Jaeger [23].

We have first examined the problem of checking the coherence of relational Bayesian networks for a given class of input structures. We used first-order logic to encode the output ground Bayesian network into a first-order structure, and we employed a transitive closure operator to express the acyclicity demanded by coherence, finally reducing the coherence checking problem to that of deciding the validity of a logical formula. We conjecture that Jaeger's original proposal concerning the format of the formula encoding the consistency of a relational Bayesian network Φ for a class S cannot be followed as originally stated; as we have argued, the possible number of tuples built from a domain typically outnumbers its size, so that there is no straightforward way to encode the ground Bayesian network, whose nodes are ground atoms, into the input S-structure. Therefore, it is hard to think of a method that translates the acyclicity of the ground Bayesian network into a formula φ_{Φ} to be evaluated over an input structure in the class S (satisfying θ_S). Our contribution here is to present a logical scheme that bypasses such difficulties by employing a bipartite domain, encoding both the S-structure and the corresponding Bayesian network. We have also extended those results to

PRMs, in fact mixing the existing graph-based techniques for coherence checking with our logic-based approach. Our results seem to be the most complete ones in the literature.

Future work includes searching for decidable instances of the formula encoding the consistency of a relational Bayesian network for a class of input structures and exploring new applications for the logic techniques herein developed.

Acknowledgments: GDB was supported by Fapesp, Grant 2016/25928-4. FGC was partially supported by CNPq, Grant 308433/2014-9. The work was supported by Fapesp, Grant 2016/18841-0.

Author Contributions: Both authors have contributed to the text and read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; nor in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CPD	Conditional Probability Table
g.a.	guaranteed acyclic
iff	if and only if
PRM	Probabilistic Relational Model
PSAT	Probabilistic Satisfiability
RBN	Relational Bayesian Network

References

1. De Finetti, B. *Theory of Probability*; Wiley: New York, NY, USA, 1974; Volumes 1 and 2.
2. Coletti, G.; Scozzafava, R. *Probabilistic Logic in a Coherent Setting*; Trends in Logic, 15; Kluwer: Dordrecht, The Netherlands, 2002.
3. Lad, F. *Operational Subjective Statistical Methods: A Mathematical, Philosophical, and Historical, and Introduction*; John Wiley: New York, NY, USA, 1996.
4. Berger, J.O. In Defense of the Likelihood Principle: Axiomatics and Coherency. In *Bayesian Statistics 2*; Bernardo, J.M., DeGroot, M.H., Lindley, D.V., Smith, A.F.M., Eds.; Elsevier Science: Amsterdam, The Netherlands, 1985; pp. 34–65.
5. Regazzini, E. De Finetti's Coherence and Statistical Inference. *Ann. Stat.* **1987**, 15, 845–864.
6. Shimony, A. Coherence and the Axioms of Confirmation. *J. Symb. Logic* **1955**, 20, 1–28.
7. Skyrms, B. Strict Coherence, Sigma Coherence, and the Metaphysics of Quantity. *Philos. Stud.* **1995**, 77, 39–55.
8. Savage, L.J. The Foundations of Statistics; Dover Publications, Inc.: New York, NY, USA, 1972.
9. Darwiche, A. *Modeling and Reasoning with Bayesian Networks*; Cambridge University Press: Cambridge, UK, 2009.
10. Koller, D.; Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*; MIT Press: Cambridge, MA, USA, 2009.
11. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Morgan Kaufmann: San Mateo, CA, USA, 1988.
12. Getoor, L.; Taskar, B. *Introduction to Statistical Relational Learning*; MIT Press: Cambridge, MA, USA, 2007.
13. De Raedt, L. *Logical and Relational Learning*; Springer: Berlin, Heidelberg, 2008.
14. Raedt, L.D.; Kersting, K.; Natarajan, S.; Poole, D. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*; Morgan & Claypool: San Rafael, CA, USA, 2016.
15. Cozman, F.G. Languages for Probabilistic Modeling over Structured Domains. *Tech. Rep.* **2018**, submitted.
16. Poole, D. First-order probabilistic inference. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico, 9–15 August 2003; pp. 985–991.
17. Gilks, W.; Thomas, A.; Spiegelhalter, D. A language and program for complex Bayesian modeling. *Statistician* **1993**, 43, 169–178.
18. Lunn, D.; Spiegelhalter, D.; Thomas, A.; Best, N. The BUGS project: Evolution, critique and future directions. *Stat. Med.* **2009**, 28, 3049–3067.

19. Getoor, L.; Friedman, N.; Koller, D.; Pfeffer, A.; Taskar, B. Probabilistic relational models. In *Introduction to Statistical Relational Learning*; Getoor, L., Taskar, B., Eds.; MIT Press: Cambridge, MA, USA, 2007.
20. Koller, D. Probabilistic relational models. In Proceedings of the International Conference on Inductive Logic Programming, Bled, Solvenia, 24–27 June 1999; pp. 3–13.
21. Heckerman, D.; Meek, C.; Koller, D. Probabilistic Entity-Relationship Models, PRMs, and Plate Models. In *Introduction to Statistical Relational Learning*; Getoor, L., Taskar, B., Eds.; MIT Press: Cambridge, MA, USA, 2007; pp. 201–238.
22. Jaeger, M. Relational Bayesian networks. In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Providence, RI, USA, 1–3 August 1997; pp. 266–273.
23. Jaeger, M. Relational Bayesian networks: A survey. *Electron. Trans. Art. Intell.* **2002**, 6, 60.
24. De Bona, G.; Cozman, F.G. Encoding the Consistency of Relational Bayesian Networks. Available online: http://sites.poli.usp.br/p/fabio.cozman/Publications/Article/bona-cozman-eniac2017F.pdf (accessed on 23 March 2018).
25. Alechina, N.; Immerman, N. Reachability logic: An efficient fragment of transitive closure logic. *Logic J. IGPL* **2000**, 8, 325–337.
26. Ganzinger, H.; Meyer, C.; Veanes, M. The two-variable guarded fragment with transitive relations. In Proceedings of the 14th IEEE Symposium on Logic in Computer Science, Trento, Italy, 2–5 July 1999; pp. 24–34.
27. Fagin, R.; Halpern, J.Y.; Megiddo, N. A Logic for Reasoning about Probabilities. *Inf. Comput.* **1990**, 87, 78–128.
28. Hansen, P.; Jaumard, B. *Probabilistic Satisfiability*; Technical Report G-96-31; Les Cahiers du GERAD; École Polytechique de Montréal: Montreal, Canada, 1996.