Encoding the Consistency of Relational Bayesian Networks

Glauber De Bona[1]*, **Fabio G. Cozman**[1]

[1]Escola Politécnica – Universidade de São Paulo (EP-USP)

Abstract. *Given an input structure, a relational Bayesian network returns a Bayesian network over ground atoms. In this paper, we analyze the problem of checking the consistency of a relational Bayesian network for a given class of input structures. This consistency is defined as the acyclicity of the output Bayesian network. We employ first-order logic to encode both the input structure and the structure of the corresponding output Bayesian network. Then we use a transitive closure operator to express the acyclicity of the resulting Bayesian network, translating the consistency of a relational Bayesian network for a given input class into the validity of a logical formula.*

1. Introduction

Computational systems targeting real-world problems usually need to reason both under uncertainty and about complex relational structures. On the one hand, probability theory is the standard mathematical tool to model uncertainty, and probabilistic graphical models, as Bayesian networks, efficiently encode probability distributions, allowing practicable inference and learning tasks. On the other hand, first-order logic is the formal framework to reason about objects and their relation, enabling one to represent general knowledge (or beliefs) over a domain. Consequently, several formalisms endowing probabilistic graphical models with first-order logic elements have been proposed in the Artificial Intelligence community in the past decades, under the field named as *statistical relational learning* [Getoor and Taskar 2007]. Instances of such approaches include Markov logic networks [Richardson and Domingos 2006], probabilistic-logic programming [Poole 1993, Sato 1995], probabilistic relational models [Koller 1999] and relational Bayesian networks [Jaeger 1997, Jaeger 2002]. The latter is the focus of the present work, as it can be regarded as at least as expressive as Markov logics [Jaeger 2008] and have probabilistic relational models as a particular case.

Relational Bayesian networks were proposed by Jaeger [Jaeger 1997, Jaeger 2002] in order to model probability distributions over predicates (or relations) whose domain is *a priori* unknown and not bounded. To grasp the expressive power of this framework, consider the following example.

Example 1 *(Scenario 1) Consider* 3 *neighbours,* x_1, x_2, x_3*, who live close enough to hear each other's house alarm. As they live in a quite dangerous zone, each day there is a probability of* 0.1% *for each house to be robbed. When a house is robbed, there is probability of* 90% *that its alarm will go off. As sometimes happens, alarms can also ring without an apparent cause, so even without any burglary, an alarm rings at a given day with probability* 1%*. If an alarm goes off, all three neighbours immediately call the police to report the incident, and they do not call otherwise. Figure 1 depicts the Bayesian network modelling this situation, when we consider the dashed arcs.*

*The first author is supported by Fapesp (grant 2016/25928-4). The second author was partially supported by CNPq (grant 308433/2014-9) and the project has been funded by FAPESP (grant 2016/18841-0).

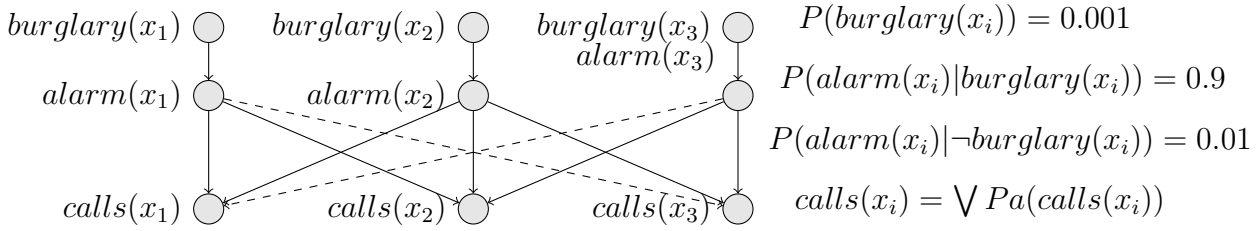

Figure 1. Bayesian network modelling the burglary-alarm-call situation with 3 people. $Pa(X)$ denotes the parents in the graph of the node X.

(Scenario 2) Now consider there are n neighbours $x_1, x_2, \ldots, x_n$ in the same situation. It is straightforward to expand the Bayesian network in Figure 1 to accommodate these n people. But if the number of neighbours is unknown, there is no fixed Bayesian networks that can allow for modelling the situation. Nevertheless, some kind of template for Bayesian networks can still capture this situation, as there is a fixed underlying pattern for the dependency arcs.

(Scenario 3) Suppose a slightly different situation with 3 people now: x_1 and x_2 are neighbours, so are x_2 and x_3, but x_1 and x_3 are not neighbours. We assume that only neighbours can hear the alarm of each other. This means that x_1 calls the police if his own alarm or x_2's goes off; the same happens with x_3. But x_2, the common neighbour, calls the police if any of the three alarms rings. This new situation is codified into the Bayesian network given in Figure 1 when we ignore the dashed arcs.

(Scenario 4) Imagine we want to generalize this new scenario for n people. Without knowing which pairs can be considered neighbours, there is no way we can predict in advance how the Bayesian network structure will be.

Relational Bayesian networks can capture exactly the type of situations like (Scenario 4) above, where the resulting Bayesian network depends not only on the domain size, but also on some relations within this domain. These relations are given in the form of a first-order structure as an input to a relational Bayesian network, whose output can be seen as a typical Bayesian network. Nevertheless, as the output Bayesian network depends on the relations that hold in the input structure, it may not be obvious in principle to check if it is a consistent – i.e., an acyclic – Bayesian network. More generally, one is interested in ensuring that input structures in a given class will always yield acyclic Bayesian networks, when the Bayesian network is defined as consistent for the given class. In this work, we encode the problem of checking consistency of a relational Bayesian network for a given class into a formula of a first-order logic with transitive closure operator.

After formally introducing relational Bayesian networks in Section 2, the consistency problem is formally defined in Section 3. In Section 4, we show how the output Bayesian network can be encoded into a first-order structure. We proceed to show how to encode the acyclicity of direct graphs by employing a transitive closure operator in Section 5. In Section 6, the consistency problem is translated into a logical formula.

2. Relational Bayesian Networks

In this section, we briefly introduce the formalism of relational Bayesian networks, in the version presented in [Jaeger 2002], which brings a thorough exposition on the topic.

Let S and R be disjoint sets of relation symbols, called the *predefined relations* and *probabilistic relations*, respectively. We assume that S contain 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 mathematical relation on 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 $\mathcal{D} \models v(t)$. According to the syntax of 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 that takes an S-structure $\mathcal{D}$, over some finite domain D, and returns a probability distribution $P(\mathcal{D}) : \mathsf{Mod}_D(R) \rightarrow [0, 1]$ over the R-structures. An example of random relational structure model would be the function in (Scenario 4) of Example 1 that receives an S-structure of neighbours and returns a joint probability distribution over ground atoms for $burglary(\cdot), alarm(\cdot), calls(\cdot)$. Each single probability distribution $P(\mathcal{D})$ is called an *instance* of the random relational structure model.

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. 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. Fixed an R-structure and an S-structure, 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, $\mathsf{noisy-or}$ is a combination function such that, if $c_1, \ldots, c_n \in [0, 1]$, $\mathsf{noisy-or}\{\!|c_1, \ldots, c_n|\!\} = \prod_{i=1}^{n} 1 - (1 - c_i)$.

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, the arity of r[1]:

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

  Above (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 corresponding 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 $r(t)$. Formally, we define the relation $\preceq$, over ground R-atoms, such that $r(t) \preceq r'(t')$ iff $r'(t') \in \alpha(F_r(x), t, \mathcal{D}_S)$. 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 2 *Recall the situation in the (Scenario 4) of Example 1. We can define a relational Bayesian network that returns a Bayesian network for each number and configuration of neighbours. Let* $S = \{neighbour(\cdot,\cdot)\}$ *and* $R =$

[1]Note that $F_{r\star}(x)$ cannot mention another $F_{r'}(x')$, but has to be inductively built from constants and indicator functions with convex combinations and combination functions; otherwise the depedency relation $\preceq$ (to be defined) would be cyclic, yielding inconsistency.

$\{burglary(\cdot), alarm(\cdot), calls(\cdot)\}$. *We assume that the relation* $neighbour$ *is reflexive symmetrical. For each relation in* R*, we associate a probability formula, forming the relational Bayesian network* Φ*:*

- $F_{burglary}(x) = 0.001$*; a constant;*
- $F_{alarm}(x) = burglary(x)0.9 + 0.1(1 - burglary(x))0.1$*; (nested) convex combinations;*
- $F_{call}(x) = \mathsf{noisy-or}\{\![alarm(y) \mid y; neighbour(x, y)]\!\}$*; a combination function.*

Note that $1 - F_1$ *and* F_1F_2 *are convex combinations and, thus, probability formulas. As the inputs of the* $\mathsf{noisy-or}$ *above are in* $\{0, 1\}$*, it actually works like a Boolean* or.

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* $D = \{x_1, x_2, x_3\}$ *such that* $\mathcal{D}_S \models neighbour(x_1, x_2) \wedge neighbour(x_2, x_3)$ *but* $\mathcal{D}_S \not\models neighbour(x_1, x_3)$*, the resulting* $P^{\Phi}_{\mathcal{D}_S}$ *is the model of (Scenario 3) in Example 1, whose Bayesian network is given in Figure 1 when we ignore the dashed arcs.*

3. Defining the Consistency Problem

For a relational Bayesian network Φ, not all S-structures yield an acyclic dependency relation $\preceq$. When the relation $\preceq$ is cyclic, the resulting Bayesian network is cyclic, and no probability distribution is defined over the R-structures. In such cases, we say Φ is *inconsistent* for that S-structure. This notion can be generalized to a class of S-structures $\mathcal{S}$, so that we say that Φ is consistent for $\mathcal{S}$ iff the resulting relation $\preceq$ is acyclic for each S-structure in $\mathcal{S}$. To know if a relational Bayesian network is consistent for given class of S-structures is precisely the problem we are interested in this work.

In order to reason about the relation between a class of S-structures and the consistency of a relational Bayesian network Φ for it, we need to formally represent these concepts. To define a class of S-structures, we can remind 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 $[[\varphi]]_S$ 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 such that $[[\theta_S]] = \mathcal{S}$.

To encode the consistency 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 is true only for S-structures yielding acyclic dependency relations $\preceq$. If that were the case, to decide the consistency of Φ for the class $\mathcal{S}$ would be reduced to decide the validity of the first-order formula $\theta_{\mathcal{S}} \rightarrow \psi_\Phi$. If this 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 consistent for $\mathcal{S}$. Otherwise, there is an S-structure in $\mathcal{S}$ yielding a cyclic dependency relation $\preceq$ for Φ[2]. Unfortunately, to encode cycles in a graph, one needs to encode the notion of path, which is the transitive closure of a relation encoding arcs, but it is a well-known fact that first-order logic cannot express transitivity. To circumvent that, we can add a transitive closure operator to the logic, arriving at the so called transitive closure logics, as described for instance in [Alechina and Immerman 2000].

[2]Note that for S-formulas only S-structures matters, and we could ignore any relation not in S. To be precise, if a first-order structure $\mathcal{D}$ falsifies $\theta_{\mathcal{S}} \rightarrow \psi_\Phi$, then there is an S-structure $\mathcal{D}_S$ (formed by ignoring non-S relations) falsifying it.

This approach was put forward by Jaeger [Jaeger 2002], 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 $\theta_S \rightarrow \psi_\Phi$ that is decidable. Nevertheless, no hint was provided as to how to construct such a formula, or as to its general aspect. A main difficulty is that, if an S-structure $\mathcal{D}$ satisfying θ_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. Thus, there is no direct way of employing the transitive closure operator to devise a formula $\neg\psi_\Phi$, encoding cycles with more than n nodes, that is to be satisfied by some structures $\mathcal{D}$ over a domain with only n elements. In the next section, we explore a path 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 θ_S accordingly.

4. 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 include 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, which includes the predefined relations S[3] and contains the unary predicate D_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 $t_i(x, y)$

[3]Recall that the equality symbol (=) is included in S.

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 $\bar{r}_i(x) \wedge t_1(x, y_1) \wedge \ldots \wedge t_k(x, y_k)$ 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.

4.1. Encoding the Nodes of the Bayesian network

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:

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

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

$$\forall x \bigwedge_{1 \leq i,j \leq m}^{i \neq j} (\neg \bar{r}_i(x) \vee \neg \bar{r}_j(x)) \tag{2}$$

- Each element corresponding to a ground atom should correspond to exactly one tuple. To achieve that, let $k = \mathsf{max}_j a(r_j)$ and consider the formula below:

$$\forall x \forall y \forall z \bigwedge_{j=1}^{k} (t_j(x, y) \wedge t_j(x, z) \rightarrow y = z) \tag{3}$$

- 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 $k = \mathsf{max}_j a(r_j)$ and consider the formula below for each $r_i \in R$:

$$\forall x \bar{r}_i(x) \rightarrow (\exists y_1 \ldots \exists y_{a(r_i)} R_i(x, y_1, \ldots, y_{a(r_i)}) \wedge \forall z \neg t_{a(r_i)+1}(x, z) \wedge \ldots \wedge \neg t_k(x, z)) \tag{4}$$

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

$$\forall y D_S(y) \rightarrow (\bigwedge_{i=1}^{m} \neg \bar{r}_i(y) \wedge \forall x \bigwedge_{j=1}^{k} \neg t_j(y, x)) \tag{5}$$

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

$$\forall y_1 \ldots \forall y_k D_S(y_1) \wedge \ldots \wedge D_S(y_k) \rightarrow \exists x R_i(x, y_1, \ldots, y_k) \tag{6}$$

  These formulas enforce that each ground atom $r(t)$ is represented by an element x that is in D_B, due to the formula (5).

- No ground atom can be represented by two different elements. Hence, for each $r_i \in R$, with $a(r_i) = k$, we employ a formula:

$$\forall y_1, \ldots \forall y_k \forall x \forall z R_i(x, y_1, \ldots, y_k) \wedge R_i(z, y_1, \ldots, y_k) \rightarrow x = z \tag{7}$$

The conjunction of all formulas in (1)–(7) 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 $\{r(t) \mid \text{for some } r \in R \text{ and } t \in D_S^{a(r)}\}$. Now we can put the arcs over these nodes to complete the structure of the ground Bayesian network $B_\Phi(\mathcal{D}_S)$.

4.2. Encoding the Arcs of the Bayesian network

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 $a(s) = k$:

$$\forall y_1 \ldots \forall y_k s(y_1, \ldots, y_k) \rightarrow D_S(y_1) \wedge \ldots \wedge D_S(y_k) \tag{8}$$

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 simply 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 $C_{F(t)}^{r'(t')} = \bot$.
- If $F(t) = r''(t)$, then $C_{F(t)}^{r'(t')} = (t' = t)$ if $r' = r''$; and $C_{F(t)}^{r'(t')} = \bot$ otherwise.

Above, $(t' = t)$ is a short form for $(t'_1 = t_1) \wedge \ldots \wedge (t'_k = t_k)$, 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 $C_{F(t)}^{r'(t')} = \bigvee_{i=1}^{3} C_{F_i(t)}^{r'(t')}$.

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 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 $C^{r'(t')}_{F(t)}$, 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 $C^{r'(t')}_{F(t)}$ for convex combinations. But 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 $C^{r'(t')}_{F(t)} = \exists t^\star\varphi(t,t^\star) \wedge \bigvee_{i=1}^{m} C^{r'(t')}_{F_i(t,t^\star)}$.

Now we can employ the formulas $C^{r'(t')}_{F(t)}$ to define the truth value of the ground relation $Dep^j_i(t,t')$, 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:

$$\forall x_1\ldots\forall x_k\forall y_1\ldots\forall y_{k'} Dep^j_i(x_1,\ldots,x_k,y_1,\ldots,y_{k'}) \leftrightarrow C^{r_j(y_1,\ldots,y_{k'})}_{F_{r_i}(x_1,\ldots,x_k)} \quad (9)$$

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 constructed 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 $r_i(t) \preceq r_j(t')$.

- 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:

$$\forall x\forall x'\forall y_1\ldots\forall y_k\forall y'_1\ldots\forall y'_{k'} R_i(x,y) \wedge R_j(x',y') \wedge Dep^j_i(y,y') \leftrightarrow Parent(x,x') \quad (10)$$

Definition 3 *Given disjoint sets of relations S and R and a relational Bayesian network $\Phi = \{F_{r_i} \mid r_i \in R\}$, the formula $\mathcal{B}_\Phi$ can be defined as the conjunction of all formulas in (1)–(10).*

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 the domain $D_B = D \setminus D_S$ 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 $B_\Phi(\mathcal{D}_S)$.

5. Encoding Acyclicity

The original formula ψ_Φ was intended to capture the consistency of the relational Bayesian network Φ. Our idea is to check the consistency 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 $\mathcal{B}_\Phi \rightarrow \psi'_\Phi$, 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'_Φ 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 compute $Parent$-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 $\mathcal{D} \models \bigwedge_{1 \leq i \leq k} Parent(z_{i-1}, z_i)$. Thus, for each k, we can define reachability through k $Parent$-arcs: $Parent{-}Path_k(x, y) = \exists z_0 \ldots \exists z_k (z_0 = x) \wedge (z_k = y) \wedge \bigwedge_{1 \leq i \leq k} Parent(z_{i-1}, z_i)$. 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 – see e.g. [Alechina and Immerman 2000]. If $\varphi(x, y)$ is a first-order formula, $\mathsf{TC}(\varphi)(x, y)$ means that y is φ-reachable from x. 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) can be encoded directly by $\mathsf{TC}(Parent)(x, x)$; and the absence of a $Parent$-cycle can be enforced by $\psi'_\Phi = \forall x \neg \mathsf{TC}(Parent)(x, x)$.

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

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

6. Encoding the Consistency-Checking Problem

Back to the consistency-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'_Φ, we can also allow its use in $\theta_\mathcal{S}$, which is useful to express S-structures without cycles, for instance.

To check the consistency of Φ for a class $\mathcal{S}$, we cannot just check the validity of $\mathcal{C}'_{\Phi,\mathcal{S}} = \theta_\mathcal{S} \rightarrow (\mathcal{B}_\Phi \rightarrow \psi'_\Phi)$, since $\theta_\mathcal{S}$ specifies S-structures over D, while $\mathcal{B}_\Phi \rightarrow \psi'_\Phi$

presupposes 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 (8), so no $x \in D$ can represent ground formulas, due to the formulas in (5), contradicting the restrictions in (6) 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 $\mathcal{B}_\Phi \rightarrow \psi'_\Phi$, yielding the satisfaction of $\mathcal{C}'_{\Phi,\mathcal{S}}$. Consequently, $\mathcal{C}'_{\Phi,\mathcal{S}}$ is valid for this specific class $\mathcal{S}$, no matter how the relational Bayesian network Φ looks like. Nonetheless, it is not hard to think of a Φ that is trivially inconsistent for any class of S-structures, like $\Phi = \{F_r(x) = r(x)\}$, with $S = \varnothing$ and $R = \{r\}$, where the probability formula associated to the relation $r \in R$ is the indicator function $r(x)$, yielding a cyclic dependecy 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 a 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 4 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 consistency of a relational Bayesian network Φ for a class of S-structures $\mathcal{S}$:

Definition 5 For disjoint sets of relations S and R, a given relational Bayesian network Φ and a class of S-structures defined by $\theta_{\mathcal{S}}$, $\mathcal{C}_{\Phi,\mathcal{S}} = \theta'_{\mathcal{S}} \rightarrow (\mathcal{B}_\Phi \rightarrow \psi'_\Phi)$.

The main result of this work is the translation of the consistency-checking problem to the validity of a formula from the transitive closure logic:

Theorem 1 *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 consistent 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. The first-order formula $\theta_{\mathcal{S}}$ (and thus $\theta'_{\mathcal{S}}$) might already be undecidable, so we assume it is decidable. Ideally, we would like to show cases for which the validity of $\mathcal{C}_{\Phi,\mathcal{S}}$ would also be decidable. Nevertheless, even restricting the relations in R to be unary, besides assuming a decidable $\theta_{\mathcal{S}}$, we could not show the decidability of the resulting problem, although there are some decidable fragments of first-order logic with transitive closure operators – e.g. [Alechina and Immerman 2000, Ganzinger et al. 1999]. Conversely, we did not manage to prove general undecidability. Hence, the question is still open for investigations.

7. Conclusion and Future Work

In this work, we studied the problem of checking the consistency 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. Then we employed a transitive closure operator to express the acyclicity needed for consistency, finally reducing the consistency-check problem to that of deciding the validity of a logical formula.

We conjecture that Jaeger's original proposal to the format of the formula encoding the consistency of a relational Bayesian network Φ for a class $\mathcal{S}$, $\theta_{\mathcal{S}} \rightarrow \psi_{\Phi}$, cannot be followed as stated. As we 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 conceive of a method that translates the acyclicity of the ground Bayesian network into a formula ψ_{Φ} to be evaluated over an input structure in the class $\mathcal{S}$ (satisfying $\theta_{\mathcal{S}}$).

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 here developed.

References

Alechina, N. and Immerman, N. (2000). Reachability logic: An efficient fragment of transitive closure logic. *Logic Journal of IGPL*, 8(3):325–337.

Ganzinger, H., Meyer, C., and Veanes, M. (1999). The two-variable guarded fragment with transitive relations. In *Logic in Computer Science, 1999. Proceedings. 14th Symposium on*, pages 24–34. IEEE.

Getoor, L. and Taskar, B. (2007). *Introduction to statistical relational learning*. MIT press.

Jaeger, M. (1997). Relational bayesian networks. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 266–273. Morgan Kaufmann Publishers Inc.

Jaeger, M. (2002). Relational bayesian networks: a survey. *Electronic Transactions in Artificial Intelligence*, 6:60.

Jaeger, M. (2008). Model-theoretic expressivity analysis. *Lecture Notes in Computer Science*, 4911:325–340.

Koller, D. (1999). Probabilistic relational models. In *International Conference on Inductive Logic Programming*, pages 3–13. Springer.

Poole, D. (1993). Probabilistic horn abduction and bayesian networks. *Artificial intelligence*, 64(1):81–129.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine learning*, 62(1):107–136.

Sato, T. (1995). A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP'95)*, pages 715–729. Citeseer.