Specifying Probabilistic Relational Models with Description Logics

Fabio G. Cozman
Escola Politécnica
Universidade de São Paulo, Brazil

Denis D. Mauá
Instituto de Matemática e Estatística
Universidade de São Paulo, Brazil

***Abstract*—In this paper we propose a modeling process for probabilistic relational models (PRMs), based on description logics and recursive relational Bayesian networks. A unified set of logical constructs is used both to specify the template skeleton of a PRM (that is, to carry out conceptual modeling) and to specify dependences and template probabilities. We also discuss inference problems: logical inference, elementary probabilistic inference, and global consistency checking.**

I. Introduction

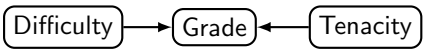
A Bayesian network represents probabilities over a set of variables, as well as independence relations among them. In practice many applications display repetitive structures; accordingly, there are several extensions of the Bayesian network language that represent relational patterns. In this paper we propose a modeling process for probabilistic relational models (PRMs), where we employ constructs from description logics to carry out both conceptual modeling and probabilistic specification. Two benefits accrue from this strategy. First, we have at our disposal well studied inference services and solid results concerning decidability, complexity, and expressivity. Second, we can use existing powerful knowledge representation packages to specify PRMs with little effort.

We start by summarizing relevant formalisms, so as to emphasize the challenges one faces when specifying probabilistic relational models (Sections II and III). We dedicate significant space to literature review and discussion, as we wish to develop a different way of looking at challenges raised by previous work. We then show, in Section IV, that a unified scheme can be produced by combining description logic sentences with simple probabilistic assessments. We examine the effect of various constructs in conceptual modeling; we then look at specification of template probabilities, and at inference problems. Section V summarizes our contributions and looks at possible future work.

II. Plates and Probabilistic Relational Models

In this section we fix some important notation and terminology; we also summarize the main ideas, and some of the challenges, behind probabilistic relational models.

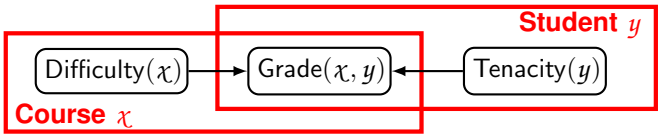
A Bayesian network uses a directed acyclic graph to represent dependences among variables [38]. For instance, an example is the "University World" [18], where a student's grade in a course depends on the student's tenacity and the course's level of difficulty:



Formally: we have a set of random variables, $X_1, \ldots, X_n$, and a directed acyclic graph where each node is a variable; the variables satisfy a Markov condition (X_i is independent of its nondescendants in the graph, given its parents $\text{PA}[X_i]$ in the graph). Hence, $\mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} \mathbb{P}(X_i = x_i | \text{PA}[X_i] = \pi_i)$, where π_i is the projection of $[x_1, \ldots, x_n]$ onto $\text{PA}[X_i]$. Each variable X_i is associated with probabilities $\mathbb{P}(X_i = x | \text{PA}[X_i] = \pi)$, so unique joint probabilities can be calculated.

Often it is important to manipulate a set of random variables together. To do so, we define a *parvariable* to be a function that yields a random variable for each element of a set (called the *domain* of the parvariable) [41]. For instance, a parvariable $X(\mathcal{X})$ with domain $\{a, b\}$ yields variables $X(a)$ or $X(b)$, depending on the value of $\mathcal{X}$. We refer to each index of a parvariable as a logical variable, abbreviated *logvar*, and to each variable produced by substituting logvars by elements of given domains as a *grounding* of the parvariable.

One successful extension of Bayesian networks is the *plate model* [21]. Each plate consists of a set of parvariables that share the same logvars (hence the same domains). A plate is typically depicted as a rectangle containing its parvariables, and information about its domain. The intersection of plates contains parvariables associated with logvars in all intersecting plates. For instance, consider again the University World, now with several students and several courses:



With each parvariable X we must associate a *template probability distribution* $\mathbb{P}(X|\text{PA}[X])$; in our example, we would have for instance $\mathbb{P}\big(\mathsf{Grade}(\mathcal{X}, y) | \mathsf{Difficulty}(\mathcal{X}), \mathsf{Tenacity}(y)\big)$. By grounding all parvariables, we obtain a possibly large Bayesian network (details on how to ground template probability distributions can be found in the literature [27]).

Originally plates were constrained so that a parvariable could only have children in the same plate or in the intersection of its current plate and other plates (with the understanding that a parvariable outside of all plates could have children everywhere). One can also find *enhanced plate models* in the literature, where a parvariable can have children everywhere. This requires one to specify template probability distributions

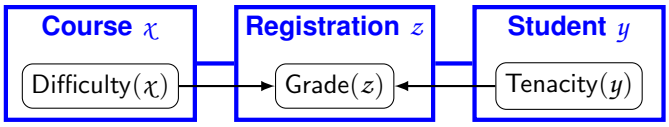


Fig. 1. PRM for the University World. We show logvars explicitly, even though they are not always used in PRMs.

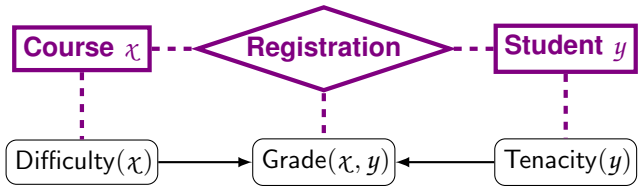


Fig. 2. DAPER diagram for the University World. Again we show logvars, even though they are not used in DAPER diagrams.

where a parvariable depends on a set of parvariables, all indexed by a logvar. For instance the parvariable $\mathsf{Ranking}(y)$ of student y may depend on all of the student's registrations. It is then necessary to specify a template probability distribution $\mathbb{P}\big(\mathsf{Ranking}(y)|\{\mathsf{Grade}(x,y)\}_y\big)$. The solution is to consider an *aggregation function* that returns a single value given a set of variables. For instance, an aggregation function may return true when at least a variable takes a particular value, and false otherwise, or perhaps it may return the number of variables taking a particular value.

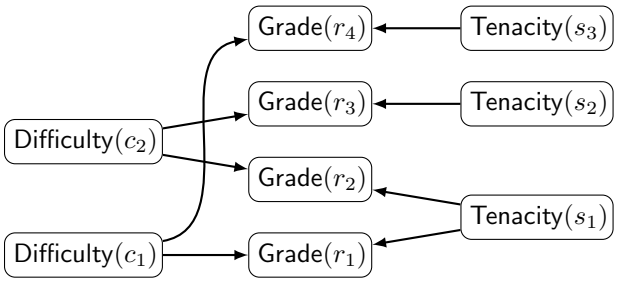
Other early proposals resorted to object orientation [29], [33] and frames [30] so as to encode repetitive structures in large Bayesian networks. Some of these proposals led to a family of models referred to as Probabilistic Relational Models [17]. We employ here the definitions proposed by Getoor et al. [18]. Most of these definitions are conveyed by a simple example; for instance, consider the simple diagram about the University World in Figure 1.

The parvariables and their domains, and the associations between objects in different domains, are the *relational schema* of the PRM. A *skeleton* is an explicit specification of objects for each domain, plus the explicit specification of pairs of objects that are associated across domains. For instance: suppose in a practical scenario that course a is the one that appears in registration b; then the pair (a, b) must appear in the skeleton.

Each parvariable is again associated with a template probability distribution. In the University World of Figure 1, we must have $\mathbb{P}\big(\mathsf{Grade}(z)|\mathsf{Difficulty}(x), \mathsf{Tenacity}(y)\big)$, with the understanding that x and y are both associated uniquely through registration z. Again, one may have to specify aggregation functions when a parvariable depends on several groundings of another parvariable. One example would be, as before, a parvariable $\mathsf{Ranking}(y)$ depending on all groundings of $\mathsf{Grade}(z)$. Hence the specification of a PRM is not entirely given by the diagram of domains and parvariables; to guarantee that a PRM is properly defined, one must check the properties of possible skeletons, and specify template probabilities accordingly.

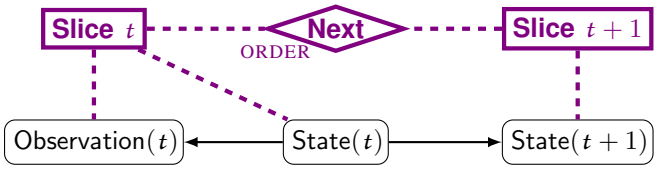
In an effort to make modeling assumptions explicitly, one might consider using the machinery of Entity-Relationship Diagrams. Indeed the DAPER language does exactly that [22]. As an example, in Figure 2 we have a DAPER diagram for the University World.

Given a PRM and a skeleton, suppose one wishes to compute probabilities or best explanations with respect to the grounded Bayesian network. For instance, suppose one is given the PRM in Figure 1, and domains $\{c_1, c_2\}$ for **Course**, $\{s_1, s_2, s_3\}$ for **Student**, and $\{r_1, r_2, r_3, r_4\}$ for **Registration**. Suppose student s_1 registers in both courses, but s_2 registers only in c_2 while s_3 registers only in c_1. The following grounded Bayesian network is produced:



To compute probabilities or best explanations, one may process the grounded Bayesian network, or perhaps exploit repetitive structures [41]. In any case, inferences that focus on properties of grounded networks are referred to as *elementary inferences* [25].

Returning to modeling, one may be interested in constraining some of the relationships between objects; for instance, the dependence of Grade on Tenacity may be present only if some other relations exist. DAPER diagrams allow constraints based on first-order logic to determine whether edges are present in groundings of a diagram [22]. Moreover, it may be necessary to allow recursive relationships. Indeed, this is the usual situation in temporal modeling, where $X(t)$ depends on $X(t')$ for $t' < t$. In fact, DAPER diagrams that encode Hidden Markov Models have been proposed [22], where the relationship **Next** orders time slices (hence the keyword ORDER):



Another example where recursive behavior is important is gene transmission. Consider the gene that determines blood type. Here a person (the child) receives information from her father and her mother. Both a PRM and a DAPER diagram have been suggested for this setting [18], [22], as depicted in Figure 3. These PRM and the DAPER models allow domains to appear several times, if properly labeled. The DAPER diagram can even resort to the keyword 2DAG to mean that "each child has at most two parents and cannot be his or her own ancestor." As for the PRM, one may be interested in marking the "is mother of" and "is father of" relations as acyclic; in fact, if there were cycles in their groundings, there would be no grounding into a valid Bayesian network. One proposal in the literature [18] is to mark associations between classes as *guaranteed-acyclic* (that means that there is no way to reach any object from itself, by following this association), and to associate colors to edges. Consider a graph where each parvariable is a node, so that: If there is an edge between two

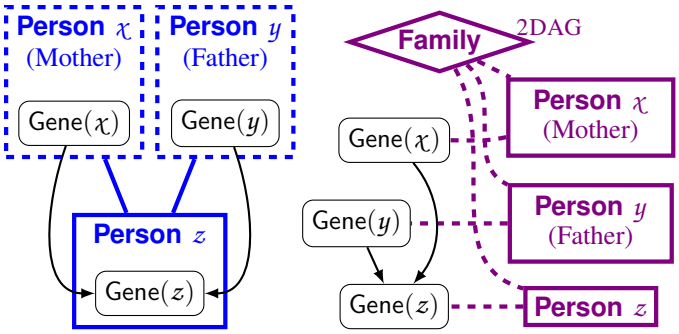


Fig. 3. The blood-type example.

parvariables whose enclosing classes are associated, then the edge is *green* if the association is guaranteed-acyclic; if the association is not guaranteed-acyclic, the edge is *red*. If the resulting *colored dependency graph* is such that every cycle contains at least a green edge and no red edge, then any grounding of this PRM is a valid (that is, acyclic) Bayesian network.

We have just defined the meaning of *global consistency checking* in probabilistic relational models: given the description of a model, determine whether any given skeleton induce a valid Bayesian network [25]. As noted later, the problem is in general quite difficult.

III. Non-diagrammatic specifications

The use of diagrams to specify repetitive structures in Bayesian networks is visually attractive. But a great deal of syntax is needed for practical modeling, and the semantics is not always transparent. Diagrams are sometimes too liberal; for instance, by letting edge constraints be expressed through sentences in first-order language [22], one cannot guarantee decidability of global consistency checking.

To avoid such difficulties, it seems sensible to resort to textual descriptions, by using formal languages to specify probabilistic relational models. On the one hand, we wish to adopt expressive languages; on the other hand, we would like to handle a limited set of constructs, to simplify analysis.

As an example, one existing proposal is the Probabilistic Relational Language (PRL) [19]. PRL offers a translation of PRMs into logic programming constructs, so as to make the grounding process unambiguous. PRL does not regulate the specification of probability values (indeed, it allows "arbitrary functional representations" for the conditional probability distributions [19, Section 3.3]). PRL divides the constructs into logical background, probabilistic background, and probabilistic dependency. The language uses predicates of varied arity to encode both the conceptual scheme and the dependencies between random variables (expressed through logical functions). Assumptions on the order of dependences are needed to obtain globally consistent models; that is, to obtain models that can be grounded into acyclic graphs. Results on complexity for PRL do not seem to be available, and it seems that producing them would be quite difficult given the intrincacies of the language.

Two other examples of formalisms that could be used to encode PRMs are Logical Bayesian Networks (LBNs) [15] and Bayesian Logic Programs (BLPs) [44]. Both distinguish between *logical* predicates that specify the grounding into graphs, and *probabilistic* or *Bayesian* predicates that encode probabilistic entities. To illustrate, consider the following Bayesian Logic Program example [44]:

$$\mathsf{carrier}(x) \mid \textbf{founder}(x)$$
$$\mathsf{carrier}(x) \mid \textbf{mother}(m,x), \textbf{father}(f,x), \mathsf{carrier}(m), \mathsf{carrier}(f)$$
$$\mathsf{suffers}(x) \mid \mathsf{carrier}(x)$$

where **founder**, **mother** and **father** are logical predicates while carrier and suffers are Bayesian ones. There are several other formalisms that move even closer to the syntax and semantics of logic programming to specify Bayesian networks [40], [45]; we do not emphasize logic programming in this paper. Other formalisms combine a functional style with probabilities [36], [48], to produce languages that are very general and powerful [16], [34]; one challenge then is that generality goes against guarantees on decidability and inferential complexity.

There are also several formalisms that employ description logics to specify Bayesian networks, often coupled with relational structures [9], [10], [13], [28]. Their goals are similar to ours; in fact the work by Carvalho et al. offers a similar framework to build (multi-entity) Bayesian networks through OWL constructs [9]. The main difference is that we encode both the skeletons and probabilities using the same constructs, and so we can employ logical inference more broadly (we discuss this matter later when we comment on inferences). Additionally, there is also a large variety of probabilistic description logics [32]; some of these logics do employ Bayesian networks [12] to represent uncertainties (but do not necessarily use the logic to specify the networks).

Another notable formalism in which PRMs can be expressed consists of Recursive Relational Bayesian Networks (RRBNs), where probability values are specified using a functional style [23], [24]. RRBNs are defined with a relatively small set of constructs, so that a solid semantics can be given without difficulties. Because the semantics of RRBNs is the semantics we adopt for our own relational expressions, we briefly summarize here the main concepts behing RRBNs.

Basically, a RRBN consists of two relational vocabularies, plus a set of *probability formulas*. One vocabulary, called the *guard vocabulary*, contains the *guard relations*; that is, the relations whose interpretation is assumed to be given in any particular application [27]. Guard relations specify the skeleton of the PRM.[1] The second vocabulary, the *parvariable vocabulary*, consists of relations whose interpretations are to receive probabilities (these are the probabilistic relations in LBNs). To connect to our previous examples: as guard vocabulary, one may use unary relations that denote classes (such as **Course**, **Student**, etc), and binary relations that denote associations (such as the association between **Course** and **Registration**). And Difficulty, Grade, and Tenacity, to be associated with probabilities, are the parvariable relations.

To specify a RRBN, one must give, for each parvariable $r(\vec{x})$, a *probability formula* with free logvars in $\vec{x}$ (we use

[1] We note that in their most recent textbook presentation [27], PRMs are just depicted as Bayesian networks where nodes are relations, in drawings that can be directly translated into RRBNs. The term "guard relation" is in fact borrowed from that presentation, not from the work on RRBNs.

$\overrightarrow{x}$ to denote a vector of logvars). Each probability formula specifies a template probability distribution $r(\overrightarrow{x})$ given some other relations: if a probability formula associated with relation r contains relations $s_1, \ldots, s_m$, then $s_1, \ldots, s_m$ are the *parents* of r, and are denoted by $\text{PA}[r]$. Given a particular interpretation for r and its parents, the corresponding probability formula yields a real number in $[0, 1]$.

Jaeger's original proposal was to have a small set of ways in which one can define probability formulas, to simplify their elicitation and analysis. A probability formula $F(\overrightarrow{x})$ can be specified (recursively) in four different ways. First, a single number in $[0, 1]$ is a probability formula. Second, an atom is a probability formula, where an atom is simply a relation $s(\overrightarrow{y})$, with its associated logvars $\overrightarrow{y}$. Third, an expression $F_1F_2 + (1 - F_1)F_3$, where F_1, F_2 and F_3 are probability formulas, is a probability formula. Fourth, to allow parents to be associated with additional logvars, Jaeger introduced *combination functions*: such a function takes a multiset of numbers in $[0, 1]$ and returns a single number in $[0, 1]$. The idea is that a probability formula may have the form $f(F_1, \ldots, F_k|\Gamma)$, where f is a combination function, the F_i are probability formulas with logvars in $\overrightarrow{x} \cup \overrightarrow{y}$, and Γ is a formula that consists of Boolean operations on relations in the guard vocabulary plus equality of logvars (that is, expressions such as $x = y$). The idea is that one will take every $\overrightarrow{y}$ that satisfies Γ; for each one of them, each F_i will produce a number, and finally all of those numbers will produce a single probability value through f [23], [24].

The intended semantics is that each probability formula $F(\overrightarrow{x})$ yields

$$\mathbb{P}\big(r(\overrightarrow{x}) = \mathsf{true}|\text{PA}[r](\overrightarrow{x}, \overrightarrow{y})\big) = F(\overrightarrow{x}).$$

Informally: each grounding of $\overrightarrow{x}$, as each logvar runs over a domain, produces a probabilistic assessment for r given the parents of r (with the implicit understanding that F contains logvars in $\overrightarrow{x} \cup \overrightarrow{y}$). As an example, consider the University World, where we have the parvariable $\mathsf{Tenacity}(y)$. As we are now restricted to relations, we should interpret $\mathsf{Tenacity}(y)$ for instance as indicating whether tenacity is high (true) or low (false). Likewise, we might interpret $\mathsf{Grade}(x, y)$ as indicating whether grades are high (true) or low (false). In any case, for the University World we could write

$$\begin{aligned}
&\mathbb{P}(\mathsf{D}(x) = \mathsf{true}) = \alpha_1,\\
&\mathbb{P}\big(\mathsf{T}(y) = \mathsf{true}\big) = \alpha_2,\\
&\mathbb{P}\big(\mathsf{G}(x, y) = \mathsf{true}|\mathsf{D}(x), \mathsf{T}(y)\big) =\\
&\qquad (\alpha_3\mathsf{D}(x) + \alpha_4(1 - \mathsf{D}(x)))\mathsf{T}(y) +\\
&\qquad (\alpha_5\mathsf{D}(x) + \alpha_6(1 - \mathsf{D}(x)))(1 - \mathsf{T}(y)),
\end{aligned}$$

using $\mathsf{D}(x)$ for $\mathsf{Difficulty}(x)$, $\mathsf{T}(y)$ for $\mathsf{Tenacity}(y)$, and $\mathsf{G}(x, y)$ for $\mathsf{Grade}(x, y)$. The first probability formula, equal to α_1, means that any grounding of x, say to element a, yields an assessment $\mathbb{P}(\mathsf{D}(a) = \mathsf{true}) = \alpha_1$. Note that $\mathsf{D}(a)$ is interpreted as a random variable over the set of interpretations for all relations; hence the assessment can be read as: probability α_1 is assigned to the set of interpretations such that $\mathsf{D}(a)$ is true. The third formula above encodes a conditional probability table for any grounding $\mathbb{P}(\mathsf{G}(a, b) = \mathsf{true}|\mathsf{D}(a), \mathsf{T}(b))$. Again, $\mathsf{T}(b)$ a random variable over all interpretations; similarly, $\mathsf{D}(a)$ and $\mathsf{G}(a, b)$ are random variables over all interpretations, for any selected elements a and b. We can thus read the probability formula as specifying, for instance, that probability α_3 is assigned to the set of interpretations such that $\mathsf{G}(a, b)$ given the set of interpretations where $\mathsf{D}(a)$ and $\mathsf{T}(b)$ are true. Even though this example does not contain combination functions, the latter are used similarly, to produce probability values for each grounding of corresponding logvars.

By grounding each probability formula, we obtain a directed graph that, if acyclic, defines a unique Bayesian network. This Bayesian network is the semantics of the RRBN. Now it may be the case that by grounding the RRBN we obtain cycles; in this case the RRBN is not globally consistent.

IV. PRM Specification based on Description Logics

One wonders whether a convenient scheme for PRM can be found, where syntax and semantics are relatively easy to grasp, and yet based on solid formalization. With such a formalism, one can comfortably specify models *and* have them analyzed automatically by suitable inference tools. Even though diagrams are usually elegant, they are not always transparent semantically; even though textual languages are usually solid, they are often difficult to digest.

Our purpose in this paper is to propose a specification framework for probabilistic relational models that is both solid and simple. We pursue a unified strategy where conceptual modeling and uncertainty management are expressed using the same set of constructs from description logics. In a sense, we wish to bring probabilistic relational modeling up to date with description logic technologies developed during the last decade or so.

A. Conceptual modeling

As far as conceptual modeling is concerned, even relatively basic description logics can represent fairly sophisticated diagrams with entities and relationships. Logics such as DL-Lite [6] already capture some diagrams, and variants of the popular logic $\mathcal{ALC}$ [47] can even cover most features of UML class diagrams [3] and conceptual graphs [46]. Before we proceed, we spend two paragraphs on basic syntax and semantics for description logics.

Typically a description logic deals with *individuals* $(a, b, \ldots)$, *concepts* $(C, D, \ldots)$, and *roles* $(r, s, \ldots)$. There are vocabularies containing the names of individuals, concepts and roles (the names in these vocabularies refer to *basic* individuals, concepts and roles). Concepts and roles can be combined to form new concepts using a set of *constructors*: *intersection* ($C \sqcap D$), *union* ($C \sqcup D$), *complement* ($\neg C$), *existential restriction* ($\exists r.C$), and *value restriction* ($\forall r.C$). The logic $\mathcal{ALC}$ adopts all of these constructs [47]. Many description logics denote by r^- the *inverse role* of r. Moreover, *concept inclusions/definitions* are denoted respectively by $C \sqsubseteq D$ and $C \equiv D$, where C and D are concepts. Concept $C \sqcup \neg C$ is denoted by $\top$, and concept $C \sqcap \neg C$ is denoted by $\bot$. Restrictions $\exists r.\top$ and $\forall r.\top$ are abbreviated by $\exists r$ and $\forall r$ respectively. A set of concept inclusions and definitions is a *terminology*. If an inclusion/definition contains a concept C in its left hand side and a concept D in its right hand side, C *directly uses* D. Indicate the transitive closure of *directly uses* by *uses*. A terminology is *acyclic* if it is a set of concept

inclusions/definitions such that no concept in the terminology uses itself [1].

The semantics of the constructs in the previous paragraph is as follows. Take a nonempty set $\mathcal{D}$, the *domain*, and a mapping $\mathcal{I}$, the *interpretation*. An interpretation $\mathcal{I}$ maps each individual to an element of the domain, each concept name to a subset of the domain, each role name to a binary relation on $\mathcal{D} \times \mathcal{D}$. An interpretation is extended to other concepts as follows: $(\neg C)^{\mathcal{I}} = \mathcal{D} \backslash (C)^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = (C)^{\mathcal{I}} \cap (D)^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = (C)^{\mathcal{I}} \cup (D)^{\mathcal{I}}$, $(\exists r.C)^{\mathcal{I}} = \{x \in \mathcal{D} | \exists y \in \mathcal{D} : (x,y) \in (r)^{\mathcal{I}} \wedge y \in (C)^{\mathcal{I}}\}$, $(\forall r.C)^{\mathcal{I}} = \{x \in \mathcal{D} | \forall y \in \mathcal{D} : (x,y) \in (r)^{\mathcal{I}} \rightarrow y \in (C)^{\mathcal{I}}\}$. Also, r^- is a role such that $(x,y) \in (r^-)^{\mathcal{I}}$ if and only if $(y,x) \in (r)^{\mathcal{I}}$. We have $C \sqsubseteq D$ if and only if $(C)^{\mathcal{I}} \subseteq (D)^{\mathcal{I}}$; and $C \equiv D$ if and only if $(C)^{\mathcal{I}} = (D)^{\mathcal{I}}$.

Consider for instance the University World. We have concepts **Course** and **Student**. We should state that these concepts are disjoint:

$$\textbf{Course} \sqsubseteq \neg \textbf{Student}.$$

In fact the disjointness of concepts is such a common circumstance, that many description logics have keywords to indicate disjointness [2]. Next we have the relationship **Registration**, as depicted in Figure 2. We might introduce a role **Registration**, to connect the other two concepts. However, if we continue with the modeling process, we see that such a strategy is not adequate. To understand the problem, consider now the grade associated with a registration. To associate a particular grade with a particular registration, we would need a ternary relation; this would move us away from most description logics, and possibly into undecidability problems. To keep matters manageable, we must treat **Registration** as a concept, and treat its associations with other guards as roles. This is indeed an example of reification, where a n-ary relation is attached to a concept, and broken down into a number of roles [4, Section 10.6.1].[2] In any case, we must encode the following diagram:

Start by declaring concepts **Course**, **Student** and **Registration** to be disjoint. Now introduce roles **registration-student** and **registration-course**, and adopt the following inclusions, where we use obvious abbreviations to save space:

$$\begin{aligned}
\textbf{Registration} &\sqsubseteq (\forall \textbf{r-c}.\textbf{C}) \sqcap (\exists \textbf{r-c}) \sqcap (\forall \textbf{r-s}.\textbf{S}) \sqcap (\exists \textbf{r-s}), \\
\textbf{Student} &\sqsubseteq (\forall \textbf{r-s}^-.\textbf{R}) \sqcap (\exists \textbf{r-s}^-), \\
\textbf{Course} &\sqsubseteq (\forall \textbf{r-c}^-.\textbf{R}) \sqcap (\exists \textbf{r-c}^-).
\end{aligned}$$

Any interpretation for this terminology is then a valid skeleton for the underlying probabilistic relational model.[3]

To refine the conceptual model even further, we can use *number restrictions*. Consider the construct $(\geq nr.C)$, where n is a non-negative integer, r is a role, and C is a concept. The semantics of $(\geq n : r.C)^{\mathcal{I}}$is given by the set

$$\{x \in \mathcal{D} : |\{y \in \mathcal{D} : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}| \geq n\},$$

where $|\cdot|$ returns cardinality of a set. Informally $(\geq nr.C)$ is the subset of the domain containing elements with no less than n links (through role r) to elements in the concept C. We also use, when needed, obviously derived constructs such as $(< nr.C)$, $(= nr.C)$, and so on. Note that number restrictions are syntactic variants of *counting quantifiers* [31], and we can read them as $\exists^{\geq n} r.C$, $\exists^{< n} r.C$, and so on.

To illustrate number restrictions for conceptual modeling, suppose we wish to enforce that a registration must be associated with a single student and a single course, while a course can have at most 50 students, and a student can attend at most 8 courses. Using UML notation to express cardinalities:

To encode this diagram, write:

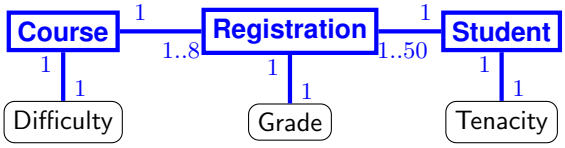
$$\begin{aligned}
\textbf{Registration} &\sqsubseteq (\forall \textbf{r-c}.\textbf{C}) \sqcap (=1\textbf{r-c}) \sqcap (\forall \textbf{r-s}.\textbf{S}) \sqcap (=1\textbf{r-s}), \\
\textbf{Student} &\sqsubseteq (\forall \textbf{r-s}^-.\textbf{R}) \sqcap (\exists \textbf{r-s}^-) \sqcap (\leq 8\ \textbf{r-s}^-), \\
\textbf{Course} &\sqsubseteq (\forall \textbf{r-c}^-.\textbf{R}) \sqcap (\exists \textbf{r-c}^-) \sqcap (\leq 50\ \textbf{r-c}^-).
\end{aligned}$$

B. *Probabilistic modeling*

Consider now the parvariables and their template probabilities. Our first problem is the ontological status of parvariables. Take parvariable Tenacity; is it a unary or a binary relation? If we take tenacity as a concept, then by writing $\mathsf{Tenacity} \sqsubseteq \textbf{Student}$ we say that some of the $a \in \mathcal{D}$ that are students, are also "tenacities". This seems incorrect, as tenacity is a property of a student.

Our solution is to encode each parvariable through a pair role-concept. For instance, we take **student-tenacity** to connect a particular student with its tenacity. And then we introduce a concept that represents the parvariable itself; in this example, we might call it Tenacity. The idea is that each element of the domain that belongs to **Student** is associated with an element of the domain that is the corresponding tenacity.

In the University World, we must encode the following diagram, where all concepts are disjoint:



To do so, we introduce not only the role **student-tenacity**, but also **course-difficulty** and **registration-grade**. The following inclusions convey the whole conceptual model (using some obvious abbreviations):

$$\begin{aligned}
\textbf{R} &\sqsubseteq (\forall \textbf{r-c}.\textbf{C}) \sqcap (=1\textbf{r-c}) \sqcap (\forall \textbf{r-s}.\textbf{S}) \sqcap (=1\textbf{r-s}) \sqcap \\
&\quad (\forall \textbf{r-g}.\mathsf{G}) \sqcap (=1\textbf{r-g}), \\
\textbf{S} &\sqsubseteq (\forall \textbf{r-s}^-.\textbf{R}) \sqcap (\exists \textbf{r-s}^-) \sqcap (\leq 8\ \textbf{r-s}^-) \sqcap \\
&\quad (\forall \textbf{s-t}.\mathsf{T}) \sqcap (=1\textbf{s-t}), \\
\textbf{C} &\sqsubseteq (\forall \textbf{r-c}^-.\textbf{R}) \sqcap (\exists \textbf{r-c}^-) \sqcap (\leq 50\ \textbf{r-c}^-) \sqcap \\
&\quad (\forall \textbf{c-d}.\mathsf{D}) \sqcap (=1\textbf{c-d}), \\
\mathsf{G} &\sqsubseteq (\forall \textbf{r-g}^-.\textbf{R}) \sqcap (=1\textbf{r-g}^-),
\end{aligned}$$

[2] This strategy is actually recommend by the W3C in their report "Defining N-ary Relations on the Semantic Web", at http://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/.

[3] Note that *domain* and *range* restrictions often used in description logics would be useful in this example, as suggested to us by a reviewer.

$$\begin{aligned}\mathsf{T} &\sqsubseteq \left(\forall \textbf{s-t}^{-}.\mathbf{S}\right) \sqcap \left(=1\textbf{s-t}^{-}\right),\\ \mathsf{D} &\sqsubseteq \left(\forall \textbf{c-d}^{-}.\mathbf{C}\right) \sqcap \left(=1\textbf{c-d}^{-}\right),\end{aligned}$$

We refer to the set of guard relations as the *guard terminology*; we do not require this terminology to be acyclic as it simply describes constraints that any skeleton is supposed to satisfy in our model. We note that all constructs discussed so far belong to the description logic $\mathcal{ALCQI}$; that is, $\mathcal{ALC}$ with inverse roles and number restrictions [8]. Typical inference services, like satisfiability and subsumption, have been studied for this logic, and have their decidability and complexity well mapped. In particular, general satisfiability and subsumption in $\mathcal{ALCQI}$ are EXP-complete problems [49].

We now examine the second issue concerning parvariables; namely, how to specify template probabilities for them. One possibility is to develop another language in which to express such templates. Instead, our proposal is to express dependences and probabilities again through description logics.

To do so, we resort to a *functional* specification of probabilities, a specification strategy that has been advocated for Bayesian networks for quite some time [42]. A functional specification can capture any probability distribution, and at the same time it can expose conditional independences betwen events (that is, "local structure" in conditional probabilities [27]). Functional specifications restricted to logical languages have been used to investigate the relationship between expressivity and complexity in Bayesian networks [11], and general functional networks are the basis of significant work on causality [39, Section 1.4].

The idea is easily explained in the propositional setting. Suppose we have three propositions A, B, C. Each proposition is associated with a random variable that yields 1 if the proposition is true, and 0 otherwise; to simplify the notation, we refer to a proposition and to its associated random variable by the same letter. Suppose additionally we have the Bayesian network $A \rightarrow C \leftarrow B$. We must specify $\mathbb{P}(A = a)$, $\mathbb{P}(B = b)$, and $\mathbb{P}(C = c|A = a, B = b)$ for all possible (a, b, c). It is easy enough to specify $\mathbb{P}(A = a)$: a single real number in $[0, 1]$ will do. Similarly, it is easy to specify $\mathbb{P}(B = b)$. Now to specify $\mathbb{P}(C = c|A = a, B = b)$, we might write down a table. Instead, we might write

$$C \Leftrightarrow (A \wedge B \wedge C'_1) \vee (A \wedge \neg B \wedge C'_2) \vee (\neg A \wedge B \wedge C'_3) \vee (\neg A \wedge \neg B \wedge C'_4),$$

where the C'_i are auxiliary binary random variables, each associated with a simple assessment $\mathbb{P}(C'_i = 1) = \alpha_i$ (these auxiliary variables are often called "independent choices" [42]). Any propositional Bayesian network can be encoded this way. Special cases are easily handled: if C is a conjunction of A and B, we just write $C \Leftrightarrow A \wedge B$.

We can directly transfer the discussion in the previous paragraph to the relational setting, where we are interested in template probabilities. So, with relation $r(\overrightarrow{x})$ we may associate the template assessment $\mathbb{P}\big(r(\overrightarrow{x})\big) = \alpha$, meaning that $\mathbb{P}(r(\overrightarrow{a})) = \alpha$ for all instantiations $\overrightarrow{a}$ of $\overrightarrow{x}$. Note that here we are implicitly equating a grounding $r(\overrightarrow{a})$ with a random variable that yields 1 if $r(\overrightarrow{a})$ is true and 0 otherwise. Of course, we can also apply this to sentences formed out of description logic constructs.

To illustrate, a possible specification in the University World is

$$\mathsf{Grade}(x, y) \Leftrightarrow \mathsf{Choice} \wedge \neg \mathsf{Difficulty}(x) \wedge \mathsf{Tenacity}(y), \quad (1)$$

to mean that a high grade is attained when the course is easy and the student's tenacity is high, provided that the independent choice Choice is true (and we might add for instance that $\mathbb{P}(\mathsf{Choice}) = 0.9$).

Our idea is to use functional specifications, where all formulas belong to description logics. Note that we must use the skeleton to build the right links between parvariables. Basically, we must "travel" through the associations in the conceptual model; again, this is best understood by taking our University World example. Here is the functional specification in first-order logic, where g, r, s, t, c and d run respectively over the (disjoint) domains for grades, registrations, students, tenacities, courses, and dedications:

$$\begin{aligned}\mathsf{G}(g) \;\Leftrightarrow\; & \mathsf{Choice} \wedge \\ & \neg\left(\forall r : \textbf{r-g}(r, g) \rightarrow (\forall c : \textbf{r-c}(r, c) \rightarrow (\forall d : \textbf{c-d}(c, d) \rightarrow \mathsf{D}(d)))\right) \\ & \wedge \left(\forall r : \textbf{r-g}(r, g) \rightarrow (\forall s : \textbf{r-s}(r, s) \rightarrow (\forall t : \textbf{s-t}(s, t) \rightarrow \mathsf{T}(t)))\right).\end{aligned}$$

This is clearly too long for easy reading; the version in $\mathcal{ALCQI}$ is much cleaner:

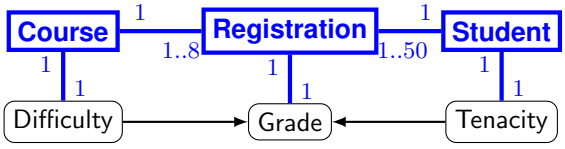
$$\begin{aligned}\mathsf{G} \;\equiv\; & \mathsf{Choice} \wedge \neg\left(\forall \textbf{r-g}^{-}.\left(\forall \textbf{r-c}.\left(\forall \textbf{c-d}.\mathsf{D}\right)\right)\right) \sqcap \\ & \left(\forall \textbf{r-g}^{-}.\left(\forall \textbf{r-s}.\left(\forall \textbf{s-t}.\mathsf{T}\right)\right)\right).\end{aligned}$$

In fact, we can write a shorter specification if we allow role composition; that is, if we allow the symbol $r \circ s$ to be used, meaning composition of relations [8]. Then we have:

$$\mathsf{G} \equiv \mathsf{Choice} \sqcap \neg\left(\forall \textbf{r-g}^{-} \circ \textbf{r-c} \circ \textbf{c-d}.\mathsf{D}\right) \sqcap \left(\forall \textbf{r-g}^{-} \circ \textbf{r-s} \circ \textbf{s-t}.\mathsf{T}\right).$$

To write these expressions, one must look at the conceptual model, and write down the "path" between the concept in the left hand side and any other concept in the right hand side. For instance, the path from Grade to Difficulty goes through the inverse of **r-g**, then through **r-c**, and finally through **c-d**.

Even though a textual description always carries a precise specification of conceptual and probabilistic modeling, in this example we may even draw a pleasant diagram that visually conveys the description:



At this point we have all facilities of RRBNs at our disposal, except one: in RRBNs one can have equality of logvars, with expressions such as $x = y$ or even $\neg(x = y)$. These constraints can be expressed using *identity roles* and Boolean operations on roles [5]. An identity role $\mathsf{id}(C)$, where C is a concept, is a role with semantics $\{(x, x) \in \mathcal{D} \times \mathcal{D} : x \in C^{\mathcal{I}}\}$ [35]; the negation of such a role creates a "difference" operator. Even though the interaction amongst all these constructs leads to a decidable fragment of first-order logic [43], refined algorithms for logical inference with them do not seem to be

available, so we do not emphasize equality handling in this paper.

To summarize: we use a functional specification, where each parvariable is either associated with a basic unconditional probabilistic assessment, or a formula in a suitable description logic. Presumably such a logic will have Boolean operators, some quantification and numeric restrictions, plus inverses (and the identity role). The description logic $\mathcal{ALCQI}_{reg}$ offers all of these constructs (and others), with EXP-complete complexity for typical inference services [8, Theorem 5.16].

C. Inference

Once a PRM is specified through the process here proposed, one has a guard vocabulary, and a guard terminology, that specify the possible skeletons; and a parvariable vocabulary, together with a funcional specification of template probabilities.

One advantage of adopting description logics to specify both skeletons and probabilities is that one can run logical inferences to verify consistency of conceptual modeling, subsumption of concepts, satisfiability of datasets. The immense body of knowledge on these matters can be used without difficulty. Logical inference is just available, if needed.

However, presumably someone interested in building a PRM is actually interested in probabilistic forms of inference. We must distinguish between elementary inference and global consistency.

Suppose a PRM is specified, and guards are fixed so that a skeleton is given. If the resulting grounded network is cyclic, the model and guards are inconsistent. If instead the grounded network is acyclic, then one can run any Bayesian network inference algorithm there [27]. Or perhaps one can operate directly over repetitive structures, running *lifted* inference [41], [50]. The complexity of such inferences have received some attention [11], [26]. Consider the problem of computing the probability of a grounding given a set of other groundings. Existing results for relational Bayesian networks show that, except for very simple languages, this problem is #P-complete as long as domains are given explicitly, if all relations have bounded arity [11]. Given that in description logics all relations are either unary or binary, these results apply to the specification languages discussed in this paper.

Much less explored is the problem of global consistency checking. One should note that global consistency checking is important in several circumstances. For instance, to deal with structural uncertainty (that is, uncertainty about guard relations [18]), one must be able to discard models that are not globally consistent. As another example, to learn PRMs from data, one must be able to search within the space of globally consistent models.

As noted previously, some simple cases of global consistency checking are dealt with by coloring arcs in different ways [18]. However, the problem of deciding whether a specified PRM is consistent is in general quite difficult, and mostly open. Jaeger has discussed this problem in connection with RRBNs [25], but his analysis focuses on rather expressive models for which consistency checking is undecidable. Jaeger conjectures that simpler languages might allow decidable consistency checking. One potential advantage of our approach is that description logics do have constructs in which to express transitive closure and, consequently, acyclicity [20]. For instance, consider expressing that associations in the Blood-Type example (Figure 3) do not produce cycles. We can write, in the guard terminology:

$$\begin{aligned}
\textbf{Founder} &\sqsubseteq \textbf{Person} \sqcap \forall \textbf{motherOf}^{-}.\bot \sqcap \forall \textbf{fatherOf}^{-}.\bot, \\
\textbf{Person} &\sqsubseteq (\forall \textbf{motherOf}^{-}.\textbf{Person}) \sqcap (\leq 1 \textbf{motherOf}^{-}) \sqcap \\
&\quad (\forall \textbf{fatherOf}^{-}.\textbf{Person}) \sqcap (\leq 1 \textbf{fatherOf}^{-}) \sqcap \\
&\quad (\exists (\textbf{motherOf}^{-})^{+}.\textbf{Founder}) \sqcap \\
&\quad (\exists (\textbf{fatherOf}^{-})^{+}.\textbf{Founder}),
\end{aligned}$$

where the superscript $+$ denotes transitive closure. In fact, relatively simple sentences can express cycles in a logic with the identity role and role operators, as long as transitive closure is available [35].

As transitive closure cannot be expresed in the two-variable fragment of first-order logic [31], decidability might be a concern. However, a description logic with all constructs of $\mathcal{ALCQI}_{reg}$ plus transitive-reflexive closure is still decidable [7], [37]; indeed, decidability has been established for even more expressive logics with transitive closure [14]. We leave for the future an exploration of this topic, but we conjecture that several expressive description logics (with transitive closure) will lead to decidable global consistency checking.

V. Conclusion

We have presented a modeling process for probabilistic relational models. Because we resort to description logics, we benefit from their solid semantics, and from results concerning expressivity and complexity. Moreover, and very importantly in practice, our modeling process can be directly used in ontology building packages. Compared to existing graphical languages (such as DAPER diagrams), our formalism is more controlled and easier to formalize; compared to existing textual languages (such as RBBNs), our formalism is based on well-known constructs from description logics that are more convenient to anyone interested in knowledge representation.

We plan, in our future work, to develop algorithms both for elementary inference and global consistency checking. Hopefully we will benefit from the relational structure to run lifted forms of elementary inference, and we will be able to use transitive closure to obtain automatic checking of acyclicity for grounded networks.

In fact, we would like to stress that global consistency checking is a logical inference problem: given a model, are cycles impossible? For such a problem to be dealt with, one must use a modeling language that allows such a question to be asked. This is a point where our framework is distinguished from other efforts: by staying within a logical formalism, we can concretely discuss, and hope to solve, global consistency checking.

Acknowledgements

The first author was partially supported by CNPq and the second author was partially supported by FAPESP.

REFERENCES

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, *Description Logic Handbook*. Cambridge Univ. Press, 2002.

[2] F. Baader and W. Nutt, "Basic description logics," in *Description Logic Handbook*. Cambridge Univ. Press, 2002, pp. 47–100.

[3] D. Berardi, D. Calvanese, and G. D. Giacomo, "Reasoning on UML class diagrams," *Artificial Intelligence*, vol. 68, pp. 70–118, 2005.

[4] A. Borgida and R. J. Brachman, "Conceptual modeling with description logics," in *Description Logic Handbook*. Cambridge Univ. Press, 2003, pp. 359–381.

[5] A. Borgida, "On the relative expressiveness of description logics and predicate logics," *Artificial Intelligence*, vol. 82, no. 1-2, pp. 353–367, 1996.

[6] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "DL-Lite: Tractable description logics for ontologies," in *AAAI*, 2005, pp. 602–607.

[7] D. Calvanese, T. Eiter, and M. Ortiz, "Regular path queries in expressive description logics with nominals," in *Int. Joint Conf. on Artificial Intelligence*, 2009, pp. 714–720.

[8] D. Calvanese and G. D. Giacomo, "Expressive description logics," in *Description Logic Handbook*. Cambridge Univ. Press, 2003, pp. 184–225.

[9] R. N. Carvalho, K. B. Laskey, and P. C. Costa, "PR-OWL 2.0 — bridging the gap to OWL semantics," in *URSW 2008-2010/UniDL 2010, LNAI 7123*, 2013, pp. 1–18.

[10] P. C. G. Costa and K. B. Laskey, "PR-OWL: A framework for probabilistic ontologies," in *Conf. on Formal Ontology in Information Systems*, 2006.

[11] F. G. Cozman and D. D. Mauá, "Bayesian networks specified using propositional and relational constructs: Combined, data, and domain complexity," in *AAAI Conf. on Artificial Intelligence*, 2015.

[12] C. d'Amato, N. Fanizzi, and T. Lukasiewicz, "Tractable reasoning with Bayesian description logics," in *Int. Conf. on Scalable Uncertainty Management*, S. Greco and T. Lukasiewicz, Eds., vol. 5291. Springer, 2008, pp. 146–159.

[13] Z. Ding, Y. Peng, and R. Pan, "BayesOWL: Uncertainty modeling in semantic web ontologies," in *Soft Computing in Ontologies and Semantic Web*, ser. Studies in Fuzziness and Soft Computing. Berlin/Heidelberg: Springer, 2006, vol. 204, pp. 3–29.

[14] C. L. Duc, M. Lamolle, and O. Cure, "$\mathcal{SHOIQ}$ with transitive closure of roles is decidable," in *Description Logics*, 2013, pp. 735–746.

[15] D. Fierens, H. Blockeel, M. Bruynooghe, and J. Ramon, "Logical Bayesian networks and their relation to other probabilistic logical models," in *Int. Conf. on Inductive Logic Programming*, 2005, pp. 121–135.

[16] D. Fierens, G. van den Broeck, J. Renkens, D. Shrerionov, B. Gutmann, G. Janssens, and L. de Raedt, "Inference and learning in probabilistic logic programs using weighted Boolean formulas," *Theory and Practice of Logic Programming*, vol. 15, pp. 358–401, 2014.

[17] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, "Learning probabilistic relational models," in *Int. Joint Conf. on Artificial Intelligence*, 1999, pp. 1300–1309.

[18] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar, "Probabilistic relational models," in *Introduction to Statistical Relational Learning*, 2007.

[19] L. Getoor and J. Grant, "PRL: A probabilistic relational language," *Machine Learning*, vol. 62, pp. 7–31, 2006.

[20] G. D. Giacomo and M. Lenzerini, "TBox and ABox reasoning in expressive description logics," in *Knowledge Representation*, 1996, pp. 316–327.

[21] W. Gilks, A. Thomas, and D. Spiegelhalter, "A language and program for complex Bayesian modelling," *The Statistician*, vol. 43, pp. 169–178, 1993.

[22] D. Heckerman, C. Meek, and D. Koller, "Probabilistic entity-relationship models, PRMs, and plate models," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. MIT Press, 2007, pp. 201–238.

[23] M. Jaeger, "Relational Bayesian networks," in *Conf. on Uncertainty in Artificial Intelligence*, D. Geiger and P. P. Shenoy, Eds. San Francisco, California: Morgan Kaufmann, 1997, pp. 266–273.

[24] ——, "Complex probabilistic modeling with recursive relational Bayesian networks," *Annals of Mathematics and Artificial Intelligence*, vol. 32, pp. 179–220, 2001.

[25] ——, "Relational Bayesian networks: a survey," *Linkoping Electronic Articles in Computer and Information Science*, vol. 6, 2002.

[26] M. Jaeger and G. V. D. Broeck, "Liftability of probabilistic inference: Upper and lower bounds," in *2nd Statistical Relational AI (StaRAI-12) Workshop*, 2012.

[27] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[28] D. Koller, A. Y. Levy, and A. Pfeffer, "P-CLASSIC: A tractable probablistic description logic," in *AAAI*, 1997, pp. 390–397.

[29] D. Koller and A. Pfeffer, "Object-oriented Bayesian networks," in *Conf. on Uncertainty in Artificial Intelligence*, 1997, pp. 302–313.

[30] ——, "Probabilistic frame-based systems," in *National Conf. on Artificial Intelligence (AAAI)*, 1998, pp. 580–587.

[31] L. Libkin, *Elements of Finite Model Theory*. Springer, 2004.

[32] T. Lukasiewicz, "Expressive probabilistic description logics," *Artificial Intelligence*, vol. 172, no. 6-7, pp. 852–883, April 2008.

[33] S. Mahoney and K. B. Laskey, "Network engineering for complex belief networks," in *Conf. on Uncertainty in Artificial Intelligence*, 1996.

[34] V. Mansinghka and A. Radul, "CoreVenture: a high-level, reflective machine language for probabilistic programming," in *NIPS Workshop on Probabilistic Programming*, 2014.

[35] F. Massacci, "Decision procedures for expressive description logics with intersection, composition, converse of roles and role identity," in *IJCAI*, 2001, pp. 193–198.

[36] B. Milch, B. Marthi, D. Sontag, S. Russell, D. L. Ong, and A. Kolobov, "BLOG: Probabilistic models with unknown objects," in *IJCAI*, 2005.

[37] M. Ortiz, "An automata-based algorithm for description logics around $\mathcal{SRIQ}$," in *LANMR*, 2008.

[38] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, California: Morgan Kaufmann, 1988.

[39] ——, *Causality: Models, Reasoning, and Inference*. Cambridge, United Kingdom: Cambridge Univ. Press, 2000.

[40] D. Poole, "The independent choice logic for modelling multiple agents under uncertainty," *Artificial Intelligence*, vol. 94(1/2), pp. 7–56, 1997.

[41] ——, "First-order probabilistic inference," in *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2003, pp. 985–991.

[42] ——, "Probabilistic programming languages: Independent choices and deterministic systems," in *Heuristics, Probability and Causality — A Tribute to Judea Pearl*, R. Dechter, H. Geffner, and J. Y. Halpern, Eds., 2010, pp. 253–269.

[43] I. Pratt-Hartmann, "Complexity of the two-variable fragment with counting quantifiers," *Journal of Logic, Language and Information*, vol. 14, no. 3, pp. 369–395, 2005.

[44] L. D. Raedt and K. Kersting, "Probabilistic inductive logic programming," in *Int. Conf. on Algorithmic Learning Theory*, 2004, p. 19-36.

[45] T. Sato and Y. Kameya, "Parameter learning of logic programs for symbolic-statistical modeling," *Journal of Artificial Intelligence Research*, vol. 15, pp. 391–454, 2001.

[46] U. Sattler, D. Calvanese, and R. Molitor, "Relationship with other formalisms," in *Description Logic Handbook*. Cambridge Univ. Press, 2003, pp. 142–183.

[47] M. Schmidt-Schauss and G. Smolka, "Attributive concept descriptions with complements," *Artificial Intelligence*, vol. 48, pp. 1–26, 1991.

[48] M. Tenorth and M. Beetz, "KnowRob: A knowledge processing infrastructure for cognition-enabled robots," *The Int. Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[49] S. Tobies, "The complexity of reasoning with cardinality restrictions and nominals in expressive description logics," *Journal of Artificial Intelligence Research*, vol. 12, pp. 199–217, 2000.

[50] G. van den Broeck, "On the completeness of first-order knowledge compilation for lifted probabilistic inference," in *Neural Processing Information Systems*, 2011.