Computing Inferences for Relational Bayesian Networks Based on $\mathcal{ALC}$ Constructs

Fabio G. Cozman[1](✉), Rodrigo B. Polastro[1], Felipe I. Takiyama[1], and Kate C. Revoredo[2]

[1] Universidade de São Paulo, Av. Prof. Mello Moraes, 2231, São Paulo–SP, Brazil
fgcozman@usp.br, rodrigopolastro@gmail.com, felipe.takiyama@usp.br
[2] Departamento de Informática Aplicada, UNIRIO, Av. Pasteur, 458, Rio de Janeiro–RJ, Brazil
katerevoredo@gmail.com

Abstract. Credal $\mathcal{ALC}$ combines the constructs of the well-known $\mathcal{ALC}$ logic with probabilistic assessments, so as to let terminologies convey uncertainty about concepts and roles. We present a restricted version of Credal $\mathcal{ALC}$ that can be viewed as a description language for a class of relational Bayesian networks. The resulting "CR$\mathcal{ALC}$ networks" offer a simplified and illuminating route both to Credal $\mathcal{ALC}$ and to relational Bayesian networks. We then describe the implementation, in freely available packages, of approximate variational and lifted exact inference algorithms.

1 Introduction

This paper focuses on a probabilistic description logic, called Credal $\mathcal{ALC}$ [7], that adds probabilistic operators to the well-known description logic $\mathcal{ALC}$ [2]. Credal $\mathcal{ALC}$ lets terminologies convey uncertainty about concepts and roles. The idea is to adopt all constructs of acyclic $\mathcal{ALC}$ terminologies, plus probabilistic assessments such as $\mathbb{P}(C|D) \in [\underline{\alpha}, \overline{\alpha}]$, where C and D are concepts, and $\mathbb{P}(r) \in [\underline{\beta}, \overline{\beta}]$, where r is a role. These probabilities are supposed elicited from experts, or learned from data.

The semantics of Credal $\mathcal{ALC}$ is based on probabilities over interpretations, with implicit independence assumptions that are encoded through a Markov condition. Given a domain, a Credal $\mathcal{ALC}$ terminology can be grounded into a set of Bayesian networks. Instead of the usual satisfiability or subsumption problems that are studied in description logics [2], here the focus is on probabilistic *inference*: given a domain and a set of assertions, compute the conditional probability of some assertion.

While the syntax of Credal $\mathcal{ALC}$ is relatively simple to grasp, the semantics is quite complex. The adopted Markov condition is far from obvious, and one needs several assumptions to guarantee that any well-formed terminology specifies a single probability measure over all interpretations.

In this paper we present a reformulation of Credal $\mathcal{ALC}$, such that any well-formed set of formulas can be directly translated into a *relational* Bayesian network. The semantics is then inherited from the theory of relational Bayesian

© Springer International Publishing Switzerland 2014
F. Bobillo et al. (Eds.): URSW 2011-2013, LNAI 8816, pp. 21–40, 2014.
DOI: 10.1007/978-3-319-13413-0_2

networks [28,29]. This reformulation simplifies the development of probabilistic terminologies, and leads to insights concerning inference algorithms. Additionally, the syntax offers a new way of specifying relational Bayesian networks. Indeed, a profitable way to understand Credal $\mathcal{ALC}$ is to take it as a description language for a restricted but useful class of relational Bayesian networks, a class that can be valuable in specifying terminologies containing uncertainty.

We then move to inference algorithms; that is, algorithms that compute probabilities given a set of sentences and assertions in the language. One advantage of connecting Credal $\mathcal{ALC}$ with Bayesian networks is that algorithms for the latter formalism can be applied to the former. We present an implementation of variational message-passing algorithms for inference, in particular algorithms that exploit symmetries amongst individuals in a domain. Such symmetries allow us to cluster variables together, and to approximate inferences solely by exchanging messages between such clusters. We then present an implementation of exact inference using *lifted* algorithms; that is, algorithms that again treat sets of variables together. We examine the use of *aggregation parfactors* in exact lifted inference.

The paper is organized as follows. Basic definitions and notation, as well as related literature, are reviewed in Sect. 2. Credal $\mathcal{ALC}$ is presented as a syntax for relational Bayesian networks in Sect. 3. Sections 4 and 5 respectively describe our implementation of variational and lifted algorithms.

2 $\mathcal{ALC}$ and (Relational) Bayesian Networks

In this section we review some necessary notions, mostly related to knowledge representation formalisms. We are interested in description logics and in relational Bayesian networks, respectively as representation for deterministic and probabilistic relationships between objects.

2.1 The Description Logic $\mathcal{ALC}$

In the popular description logic $\mathcal{ALC}$ [66] we have *individuals*, *concepts*, and *roles*, to be understood as constants, unary relations, and binary relations. Throughout, $a, b, a_1, a_2, \ldots$ are individuals; $C, D, C_1, C_2, \ldots$ are concepts; and $r, r_1, r_2, \ldots$ are roles. Concepts and roles can 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* $(\forall r.C)$. *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 [2].

A terminology may be associated with *assertions* about individuals or pairs of individuals; for instance, $\mathsf{Fruit}(\mathsf{appleFromJohn})$ and $\mathsf{buyFrom}(\mathsf{houseBob}, \mathsf{John})$. Intuitively, an assertion is the grounding of a unary/binary relation. A set of assertions is called an *Abox*.

The semantics of $\mathcal{ALC}$ is given by 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: $(\neg C)^{\mathcal{I}} = \mathcal{D} \backslash (C)^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = (C)^{\mathcal{I}} \cap (D)^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = (C)^{\mathcal{I}} \cup (D)^{\mathcal{I}}$, $(\exists r.C)^{\mathcal{I}} = \{x \in \mathcal{D} | \exists y \in \mathcal{D} : (x, y) \in (r)^{\mathcal{I}} \wedge y \in (C)^{\mathcal{I}}\}$, $(\forall r.C)^{\mathcal{I}} = \{x \in \mathcal{D} | \forall y \in \mathcal{D} : (x, y) \in (r)^{\mathcal{I}} \rightarrow y \in (C)^{\mathcal{I}}\}$. We have $C \sqsubseteq D$ if and only if $(C)^{\mathcal{I}} \subseteq (D)^{\mathcal{I}}$; and $C \equiv D$ if and only if $(C)^{\mathcal{I}} = (D)^{\mathcal{I}}$.

Most description logics have direct translations into multi-modal logics [65] and fragments of first-order logic [4]. We often treat a concept C as a unary predicate $C(x)$, and a role r as a binary predicate $r(x, y)$.

2.2 Bayesian Networks

Now consider Bayesian networks, a popular representation for probability distributions. A Bayesian network consists of a directed acyclic graph $\widehat{\mathbf{G}}$ where each node is a random variable V_i and where the following Markov condition is assumed [53]: every random variable V_i is independent of its nondescendants nonparents given its parents. For categorial variables $V_1, \ldots, V_n$, this Markov condition implies the following factorization for joint probabilities:

$$\mathbb{P}(V_1 = v_1, \ldots, V_n = v_n) = \prod_{i=1}^{n} \mathbb{P}(V_i = v_i | \mathsf{pa}(V_i) = \pi_i) \,, \tag{1}$$

where $\mathsf{pa}(V_i)$ denotes the parents of V_i in the graph, and π_i denotes the configuration of parents of random variable V_i. Note that if a random variable V_i has no parents, then the unconditional probability $\mathbb{P}(V_i = x_i)$ is used in Expression (1). We say that $\mathbb{P}$ factorizes according to $\widehat{\mathbf{G}}$ if $\mathbb{P}$ satisfies Expression (1).

2.3 Probabilistic Description Logics

There has been considerable interest in languages that mix probability assessments and constructs employed in description logics [44,61]. Early proposals by Heinsohn [24], Jaeger [27] and Sebastiani [67] adopt probabilistic inclusion axioms with a *domain-based* semantics; that is, probabilities are assigned to subsets of the domain. Proposals in the literature variously adopt a domain-based semantics [13,14,20,35,37,42,76], or an interpretation-based semantics where probabilities are assigned to sets of interpretations [6,21,43,45,67].

Several probabilistic description logics rely on graphs to encode stochastic independence relations. The first language to resort to Bayesian networks, P-CLASSIC, enlarges the logic CLASSIC with a set of Bayesian networks so as to specify a single probability measure over the domain [37]. A limitation is that P-CLASSIC does not handle assertions. Other logics that combine terminologies with Bayesian networks are Yelland's Tiny Description Logic [76], Ding and Peng's BayesOWL language [13], and Staker's logic [69] (none can handle assertions). Costa and Laskey's PR-OWL language [6] adopts an interpretation-based semantics inherited from multi-entity Bayesian networks (MEBNs) [5]. Another path is to consider undirected models, for instance based on Markov logic [49].

For the purposes of this paper, a particularly interesting class of languages has been produced by combining Poole's choice variables [57] with description logics [8,43,63].

Besides the literature just reviewed, there is a large body of work on knowledge databases [26,59] and on fuzzy description logics [44]; also notable is Nottelmann and Fuhr's probabilistic version of the OWL language [51].

2.4 Relational Bayesian Networks

Combinations of logic, probabilities and independence assumptions are not limited to description logics. They range from simple template languages [46,71,74], to rule-based languages akin to Prolog [48,56,64], and to more sophisticated languages such as multi-entity Bayesian networks [39] and Markov logic [62]. Research on probabilistic logics sometimes emphasizes automated learning [19,60]. The term *Probabilistic Relational Model* (PRM) is frequently associated with languages that combine Bayesian networks with relational logic [16,18,38]. Overall, these languages move beyond older probabilistic logics [3,9,50] by explicitly considering Markov conditions. For our purposes, relational Bayesian networks [28–30] offer the most relevant language, which we now discuss.

A *relational Bayesian network* is a compact, graph-based representation for a joint distribution over a set of random variables specified via relations and their groundings over a domain [28,29]. We start with a vocabulary $\mathcal{S}$ containing finitely many relations. We wish to specify a probability measure over the set of interpretations for these relations. To do so, we specify a directed acyclic graph $\mathbf{G}$ where each node is a relation in $\mathcal{S}$. Each relation s is then associated with a *probability formula* F_s. To understand these formulas, we must understand the intended semantics.

To define the semantics, consider a domain $\mathcal{D}$ (a set with individuals). An *interpretation* $\mathcal{I}$ is a function that takes each k-ary relation to a set of k-tuples of elements of $\mathcal{D}$. Now given a k-ary relation $s \in \mathcal{S}$ and a k-tuple $\mathbf{a} \in \mathcal{D}^k$, associate with the grounding $s(\mathbf{a})$ the indicator function

$$\mathbb{1}_{s(\mathbf{a})}(\mathcal{I}) = \begin{cases} 1 \text{ if } \mathbf{a} \in (s)^{\mathcal{I}}, \\ 0 \text{ otherwise.} \end{cases}$$

Note: to emphasize the connection between interpretations in $\mathcal{ALC}$ and in relational Bayesian networks, we used the notation $(s)^{\mathcal{I}}$ in this expression.

We extend this notation to any formula ϕ, indicating by $\mathbb{1}_{\phi}(\mathcal{I})$ the function that yields 1 if ϕ holds in $\mathcal{I}$, and 0 otherwise. We wish to specify a probability

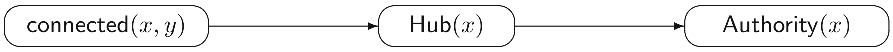


Fig. 1. Relational Bayesian network in Example 1.

distribution over the set of all indicator variables $\mathbb{1}_\phi$. Probability formulas allow us to do so: a probability formula F_s specifies how to compute the distribution of $\mathbb{1}_{s(\mathbf{a})}$ for any appropriate tuple $\mathbf{a}$ of elements of $\mathcal{D}$.

Now return to the syntax. Jaeger restricted probability formulas to four constructs, defined recursively as follows [28,29]. First, a real number in $[0,1]$ is a probability formula. Second, a parameterized indicator function $\mathbb{1}_{s(\mathbf{x})}$, where $\mathbf{x}$ is a tuple of logical variables, is a probability formula. Third, $F_1 \times F_2 + (1 - F_1) \times F_3$, where F_1, F_2, and F_3 are probability formulas, is a probability formula.

Example 1. Consider the graph in Fig. 1. The assessment

$$\forall (x,y) \in \mathcal{D} \times \mathcal{D} : \mathbb{P}\big(\mathbb{1}_{\mathsf{connected}(x,y)} = 1\big) = 0.2$$

corresponds to the concise probability formula $F_{\mathsf{connected}}(x, y) = 0.2$. Consider also the assessments: for $\eta \in \{0, 1\}$,

$$\forall x \in \mathcal{D} : \mathbb{P}\big(\mathbb{1}_{\mathsf{Authority}(x)} = 1 | \mathbb{1}_{\mathsf{Hub}(x)} = \eta\big) = 0.9\eta + 0.2(1 - \eta).$$

These assessments correspond to the probability formula:

$$F_{\mathsf{Authority}}(x) = 0.9 \times \mathbb{1}_{\mathsf{Hub}(x)} + 0.2 \times (1 - \mathbb{1}_{\mathsf{Hub}(x)}).$$

Finally, consider the assessments: for $\eta \in \{0, 1\}$,

$$\forall x \in \mathcal{D} : \mathbb{P}\big(\mathbb{1}_{\mathsf{Hub}(x)} = 1 | \mathbb{1}_\phi = \eta\big) = 0.3\eta + 0.01(1 - \eta),$$

where ϕ is the first-order formula $\exists y \in \mathcal{D} : \mathsf{connected}(x, y)$. To encode this into a probability formula, we need to somehow quantify over y. To do so, Jaeger introduced a fourth construct, as we now describe. $\square$

A *combination function* is any function that takes a tuple of numbers in $[0,1]$ and returns a number in $[0,1]$. Examples are:

$$\text{Noisy-OR}(A) = 1 - \prod_{p \in A} (1 - p), \qquad \text{Mean}(A) = \sum_{p \in A} p/|A|.$$

Tuples are specified as follows. Denote by $c(\mathbf{x}, \mathbf{y})$ a set of equality constraints containing logical variables in tuples $\mathbf{x}$ and $\mathbf{y}$. Denote by $\langle \mathbf{y} : c(\mathbf{x}, \mathbf{y}) \rangle$ the set of all groundings of $\mathbf{y}$ that satisfy $c(\mathbf{x}, \mathbf{y})$ for fixed $\mathbf{x}$. For each tuple $\mathbf{x}$, generate the set $\langle \mathbf{y} : c(\mathbf{x}, \mathbf{y}) \rangle$; now for each tuple in this set, evaluate $F_1, \dots, F_k$. So if there are m tuples in $\langle \mathbf{y} : c(\mathbf{x}, \mathbf{y}) \rangle$ for fixed $\mathbf{x}$, then there are $k \times m$ elements in the resulting tuple. Denote by $\{F_1(\mathbf{x}, \mathbf{y}), \dots, F_k(\mathbf{x}, \mathbf{y}); \langle \mathbf{y} : c(\mathbf{x}, \mathbf{y}) \rangle\}$ the tuple; note that not necessarily all variables in $(\mathbf{x}, \mathbf{y})$ appear in all probability formulas F_i.

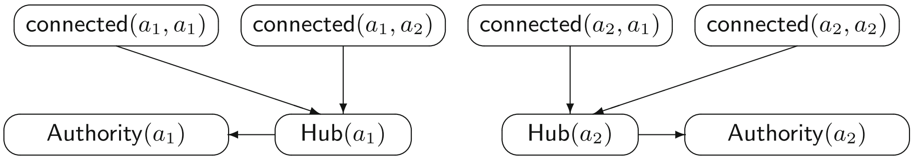


Fig. 2. Grounded graph $\widehat{\mathbf{G}}$ for Example 1.

Returning to Example 1, we can write the last assessment as:

$$F_{\mathsf{Hub}}(x) = \text{Noisy-OR}(\{\mathsf{connected}(x, y); \langle y : y = y\rangle\}).$$

Combination functions are quite powerful, but as a knowledge representation tool they have somewhat difficult syntax, and may be hard to understand.

We can take the graph $\mathbf{G}$ and the associated probability formulas, and generate a grounded graph $\widehat{\mathbf{G}}$. The nodes of $\widehat{\mathbf{G}}$ are indicator functions $\mathbb{1}_{s(\mathbf{a})}$ for all s and all appropriate $\mathbf{a}$. An edge is added to $\widehat{\mathbf{G}}$ from node $s_1(\mathbf{a}_1)$ to node $s_2(\mathbf{a}_2)$ if the probability formula for s_2 mentions the relation s_1. The Markov condition for Bayesian networks is then assumed for $\widehat{\mathbf{G}}$, and hence we obtain a factorization for the joint distribution of all indicator functions $\mathbb{1}_{s(\mathbf{a})}$.

Returning again to Example 1, Fig. 2 shows the grounded graph $\widehat{\mathbf{G}}$ for domain $\{a_1, a_2\}$. To simplify the figure, every indicator function $\mathbb{1}_{s(\mathbf{a})}$ is denoted simply by $s(\mathbf{a})$.

In this paper we do not consider *recursive relational Bayesian networks*, defined by Jaeger to allow typed relations and temporal evolution [28,29].

3 Credal $\mathcal{ALC}$

Credal $\mathcal{ALC}$ was proposed [7] as a syntactically simple probabilistic extension of $\mathcal{ALC}$. The syntax and associated semantics allows one to specify sets of probability measures over the set of interpretations for a given vocabulary.

In Sect. 3.1 we summarize the properties of the most flexible and general version of the language. Then in Sect. 3.2 we reformulate Credal $\mathcal{ALC}$ as a specification language for (a class of) relational Bayesian networks.

3.1 Credal $\mathcal{ALC}$ as a Flexible Mix of $\mathcal{ALC}$ and Probabilities

The language consists of well-formed $\mathcal{ALC}$ sentences, plus assessments

$$\mathbb{P}(C|D) \in [\underline{\alpha}, \overline{\alpha}], \qquad \mathbb{P}(r) \in [\underline{\beta}, \overline{\beta}],$$

where C is a concept name, D is a concept, and r is a role name. If the concept D is equal to $\top$ in the first assessment, we just write $\mathbb{P}(C) \in [\underline{\alpha}, \overline{\alpha}]$. Hence we have a subset of many existing probabilistic logics such as Lukasiewicz's conditional constraints [41]. The idea behind Credal $\mathcal{ALC}$ is to impose additional

$\mathbb{P}(\mathsf{Animal}) = 0.9$,
$\mathbb{P}(\mathsf{Rational}) = 0.6$,
$\mathbb{P}(\mathsf{hasChild}) = 0.3$,
$\mathsf{Human} \equiv \mathsf{Animal} \sqcap \mathsf{Rational}$,
$\mathsf{Beast} \equiv \mathsf{Animal} \sqcap \neg\mathsf{Rational}$,
$\mathsf{Parent} \equiv \mathsf{Human} \sqcap \exists\mathsf{hasChild}.\mathsf{Human}$,
$\mathbb{P}(\mathsf{Kangaroo}|\mathsf{Beast}) = 0.4$,
$\mathsf{Kangaroo} \sqsubseteq \mathsf{Beast}$,
$\mathsf{MaternityKangaroo} \equiv \mathsf{Kangaroo} \sqcap \exists\mathsf{hasChild}.\mathsf{Kangaroo}$.

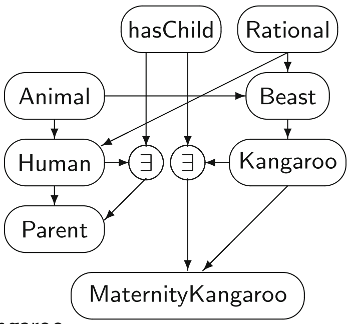


Fig. 3. The Kangaroo network. Note that, as a visual aid, existential restrictions are explicitly shown as nodes in the graph, even though these nodes do not correspond directly to concepts in the terminology.

assumptions so that each consistent set of sentences can be grounded into a set of Bayesian networks, all sharing the same graph. Note that a set of probability distributions is often called a *credal set* [1], hence the name Credal $\mathcal{ALC}$.

We first need to have an appropriate concept of acyclicity. We adopt the relations *directly uses* and *uses* from $\mathcal{ALC}$, and extend them as follows. Given the assessment $\mathbb{P}(C|D) \in [\underline{\alpha}, \overline{\alpha}]$, we say that C *directly uses* D. And *uses* is the transitive closure of *directly uses*. Now for any concept C in the terminology, denote by $\mathsf{pa}(C)$ the set of concepts directly used by C, and by $\mathsf{nd}(C)$ the set of concepts that do not use C. Say that an assertion directly uses another one if the corresponding concepts satisfy the *directly uses* relation. Again, a terminology is acyclic if no concept in the terminology uses itself.

Given an acyclic terminology, we can draw a directed acyclic graph where nodes are concept and role names, and arrows encode the *directly uses* relation. For instance, Fig. 3 shows a probabilistic version of the Kangaroo ontology and the associated graph.[1]

We assume that every terminology is acyclic. By adopting acyclicity, we can define an appropriate Markov condition; to so, we need to examine the semantics.

First, all $\mathcal{ALC}$ constructs have the usual semantics, based on a domain and interpretations. As in many probabilistic logics [22,23], we adopt an assumption of *rigidity*: the interpretation of an individual does not change across interpretations. We also adopt throughout the *unique name assumption*: distinct individual names refer to distinct elements of the domain. Thus we can equate individuals with elements of the domain.

To define the semantics of probabilistic assessments, take a domain $\mathcal{D}$. Then define, for any individual a and concept C, a set of interpretations

$$\langle\langle C(a) \rangle\rangle = \{\mathcal{I} : a \in (C)^{\mathcal{I}}\}.$$

[1] The Kangaroo ontology is distributed with the CEL System at the site http://lat.inf.tu-dresden.de/systems/cel/.

Here $\langle\langle C(a)\rangle\rangle$ depends on $\mathcal{D}$, but we leave this dependency implicit to avoid burdening the notation (the domain of interest can be inferred from the context). Similarly, define the set of interpretations $\langle\langle r(a,b)\rangle\rangle = \{\mathcal{I} : (a,b) \in (r)^{\mathcal{I}}\}$, for any pair of individuals (a,b) and any role r. The semantics of a set of assessments is given by a probability measure $\mathbb{P}$ over the set of interpretations: each assessment $\mathbb{P}(C|D) \in [\underline{\alpha}, \overline{\alpha}]$ means

$$\forall x \in \mathcal{D} : \mathbb{P}(\langle\langle C(x)\rangle\rangle | \langle\langle D(x)\rangle\rangle) \in [\underline{\alpha}, \overline{\alpha}],$$

and each assessment $\mathbb{P}(r) \in [\underline{\beta}, \overline{\beta}]$ means

$$\forall (x,y) \in \mathcal{D} \times \mathcal{D} : \mathbb{P}(\langle\langle r(x,y)\rangle\rangle) \in [\underline{\beta}, \overline{\beta}].$$

Note that we abuse notation by using the same symbol $\mathbb{P}$ in the syntax and the semantics.

Given a terminology and a domain, we can always construct a directed acyclic graph $\widehat{\mathbf{G}}$ where nodes are all possible assertions, and where arrows encode the *directly uses* relation. As in Fig. 3, we add a node to $\widehat{\mathbf{G}}$ for each existential (or universal) quantifier in the terminology. Thus $\widehat{\mathbf{G}}$ is the grounding of the terminology for the given domain.

For example, consider a terminology with the following axioms and assessments (based on Ref. [7]): $\mathbb{P}(A) = 0.9$, $B \sqsubseteq A$, $D \equiv \forall r.A$, $C \equiv B \sqcup \exists r.D$, $\mathbb{P}(r) = 0.3$. The terminology can be drawn as a directed acyclic graph as in Fig. 4. Now suppose we have a domain with just two individuals, a and b. The grounded graph $\widehat{\mathbf{G}}$ is also shown in Fig. 4.

With this, we can state a *Markov condition*:[2] $\langle\langle C(a)\rangle\rangle$ is independent of all $\langle\langle ND\rangle\rangle$ where ND is a nondescendant nonparent of $C(a)$ in $\widehat{\mathbf{G}}$, given all $\langle\langle PA\rangle\rangle$ where PA is a parent of $C(a)$ in $\widehat{\mathbf{G}}$. That is, we just have the usual Markov condition for the grounded directed acyclic graph.

In practice it may be useful to adopt a number of assumptions that imply that a terminology can always be grounded into a single probability measure over interpretations (for instance, each assessment collapses to a single number). Indeed, Cozman and Polastro [7] have identified a number of assumptions that together guarantee uniqueness; some of these assumptions are easy to grasp, while others are quite convoluted.

We present in the next section a syntax, and a set of associated assumptions, that guarantees that any set of well-formed sentences can be grounded into a single Bayesian network given a finite domain. We do so by framing Credal $\mathcal{ALC}$ as a language for specification of relational Bayesian networks.

[2] We use the following concept of independence: an event E is independent of a set of events $\{F_i\}_i$ given a set of events $\{G_j\}_j$ if $\mathbb{P}(E \cap H'|H'') = \mathbb{P}(E|H'')\,\mathbb{P}(H'|H'')$ for any $H' = \cap_{i \in I} F_i$ and any nonempty $H'' = (\cap_{j \in J} G_j) \cap (\cap_{k \in K} G_k^c)$, for any subsets of indexes I, J, K.

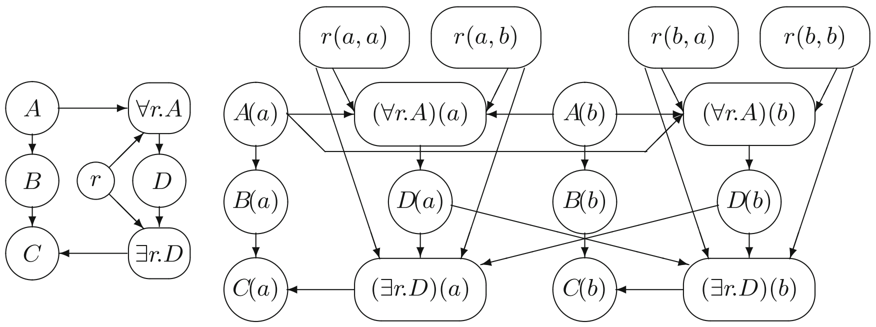


Fig. 4. Left: directed acyclic graph representing terminology. Right: grounding of the terminology for $\mathcal{D} = \{a, b\}$.

3.2 Credal $\mathcal{ALC}$ as Specification Language for Relational Bayesian Networks

As before, consider a vocabulary $\mathcal{S}$ containing individuals, concepts, and roles. A CR$\mathcal{ALC}$ *network* consists of a directed acyclic graph **G** where each node is either a concept name or a relation name, and where each node is associated either with

– a direct assessment: $\mathbb{P}(C) = \alpha$ if the node is a concept C, or $\mathbb{P}(r) = \alpha$ if the node is a role r, for $\alpha \in [0, 1]$; or
– a definition $C \equiv \phi$, if the node is a concept C, that must be a well-formed definition in $\mathcal{ALC}$ whose right-hand side only refers to parents of C.

To establish semantics for CR$\mathcal{ALC}$ networks, we translate this syntax directly into relational Bayesian networks. Consider a domain $\mathcal{D}$. Concepts and roles are viewed as unary and binary relations, and the semantics is given by a probability measure over $\mathcal{I}$, the set of interpretations of these relations. Additionally:

– A direct assessment $\mathbb{P}(C) = \alpha$, where C is a concept, is interpreted just as the probability formula $F_C(x) = \alpha$; that is, as

$$\forall x \in \mathcal{D} : \mathbb{P}\big(\mathbb{1}_{C(x)} = 1\big) = \alpha.$$

– A direct assessment $\mathbb{P}(r) = \alpha$, where r is a role, is interpreted just as the probability formula $F_r(x, y) = \alpha$; that is, as

$$\forall (x, y) \in \mathcal{D} \times \mathcal{D} : \mathbb{P}\big(\mathbb{1}_{r(x,y)} = 1\big) = \alpha.$$

The semantics of a definition $C \equiv \phi$ is immediate: for all $x \in \mathcal{D}$, every interpretation satisfies $C(x) \leftrightarrow \phi(x)$, where $\phi(x)$ is the translation of ϕ to first-order logic, mapping intersection to conjunction, complement to negation, and so on. As a digression, note that such definitions can be expressed through probability functions, as any first-order logic formula can be encoded through Jaeger's probability formulas [28, Lemma 2.4].

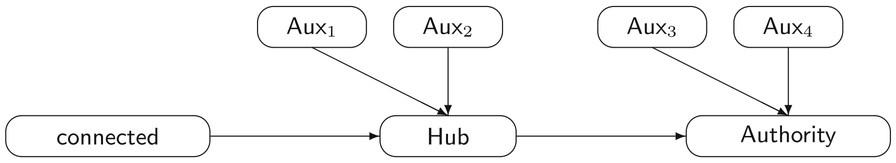


Fig. 5. Simple CR$\mathcal{ALC}$ network.

Now consider the graph $\widehat{\mathbf{G}}$ where each node is a grounded relation, exactly as done previously for relational Bayesian networks. Attach to $\widehat{\mathbf{G}}$ the usual Markov condition for Bayesian networks. The semantics of a CR$\mathcal{ALC}$ network is given by a probability measure that factorizes in accordance with this Markov condition.

Example 2. Fig. 5 shows a CR$\mathcal{ALC}$ network with seven variables, four of which are auxiliary variables Aux_i, where the same probabilities in Example (1) can be obtained as follows:

$$\mathbb{P}(\mathsf{connected}) = 0.2,$$
$$\mathsf{Hub} \equiv (\mathsf{Aux}_1 \sqcap \exists\mathsf{connected}) \sqcup (\mathsf{Aux}_2 \sqcap \neg\exists\mathsf{connected}),$$
$$\mathsf{Authority} \equiv (\mathsf{Aux}_3 \sqcap \mathsf{Hub}) \sqcup (\mathsf{Aux}_4 \sqcap \neg\mathsf{Hub}),$$
$$\mathbb{P}(\mathsf{Aux}_1) = 0.3, \quad \mathbb{P}(\mathsf{Aux}_2) = 0.01, \quad \mathbb{P}(\mathsf{Aux}_3) = 0.9, \quad \mathbb{P}(\mathsf{Aux}_4) = 0.2.$$

The grounded graph in Fig. 2 is again obtained for domain $\{a_1, a_2\}$. □

This example shows that, if properly combined, direct assessments and definitions can be used to build conditional probabilities. In fact, suppose we have a CR$\mathcal{ALC}$ network where C and D are concepts, D is the only parent of C, and we wish to express that, for all $x \in \mathcal{D}$,

$$\mathbb{P}\big(\mathbb{1}_{C(x)} = 1 | \mathbb{1}_{D(x)} = 1\big) = \alpha \quad \text{and} \quad \mathbb{P}\big(\mathbb{1}_{C(x)} = 1 | \mathbb{1}_{D(x)} = 0\big) = \beta.$$

Introduce two fresh concept names C_1 and C_2, add them as parents of C, but leave them without parents; then specify:

$$C \equiv (C_1 \sqcap D) \sqcup (C_2 \sqcap \neg D), \qquad \mathbb{P}(C_1) = \alpha, \qquad \mathbb{P}(C_2) = \beta. \tag{2}$$

Note that, as desired, $\mathbb{P}\big(\mathbb{1}_{C(x)} = 1 | \mathbb{1}_{D(x)} = 1\big)$ is equal to

$$\sum_{\mathbb{1}_{C_1(x)}, \mathbb{1}_{C_2(x)}} \mathbb{P}\big(\mathbb{1}_{C(x)} = 1 | \mathbb{1}_{C_1(x)}, \mathbb{1}_{C_2(x)}, \mathbb{1}_{D(x)} = 1\big)\, \mathbb{P}\big(\mathbb{1}_{C_1(x)}\big)\, \mathbb{P}\big(\mathbb{1}_{C_2(x)}\big)$$
$$= \sum_{\mathbb{1}_{C_2(x)}} \mathbb{P}\big(\mathbb{1}_{C_1(x)} = 1\big)\, \mathbb{P}\big(\mathbb{1}_{C_2(x)}\big) = \alpha.$$

By similar reasoning, we see that $\mathbb{P}\big(\mathbb{1}_{C(x)} = 1 | \mathbb{1}_{D(x)} = 0\big) = \beta$. This sort of device let us specify conditional probability tables for a concept conditional on arbitrarily many concepts. In fact we can even introduce some syntactic sugar by writing the set of constructs in Expression (2) directly as $C \equiv \alpha D \sqcup \beta(\neg D)$. Moreover, there is no need to draw auxiliary variables in our graphs, as the next example shows.

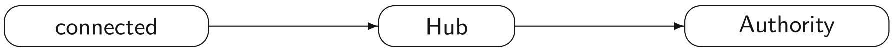


Fig. 6. Simple CR$\mathcal{ALC}$ network without auxiliary variables.

Example 3. Consider again the CR$\mathcal{ALC}$ network in Example 2. The graph can be drawn as in Fig. 6, with additional concise assessments:

$$\mathbb{P}(\mathsf{connected}) = 0.2, \qquad \begin{array}{rl} \mathsf{Hub} \equiv & 0.3\ (\exists \mathsf{connected}) \sqcup \\ & 0.01\ (\neg \exists \mathsf{connected}), \end{array} \qquad \begin{array}{rl} \mathsf{Authority} \equiv & 0.9\ (\mathsf{Hub}) \sqcup \\ & 0.2\ (\neg \mathsf{Hub}). \end{array}$$

Again, for domain $\{a_1, a_2\}$, the grounded graph is depicted in Fig. 2. □

Another example is given by the Kangaroo network in Fig. 3. We can replace both the assessment $\mathbb{P}(\mathsf{Kangaroo}|\mathsf{Beast}) = 0.4$ and the inclusion $\mathsf{Kangaroo} \sqsubseteq \mathsf{Beast}$ by

$$\mathsf{Kangaroo} \equiv 0.4 \mathsf{Beast}.$$

Note that auxiliary variables resemble *choice* construct in ICL [57], but the semantics for the latter employs different assumptions regarding negation. Similarly, auxiliary variables resemble *switches* in PRISM [64]; the latter are more general and take parameters. Finally, auxiliary variables resemble the *exogenous variables* used in structural equations [54]; of course the latter are couched in terms of algebraic modeling.

We can even extend the previous constructs if we have three concepts C_1, C_2 and C_3. We can then use the definition

$$C \equiv (C_1 \sqcap C_2) \sqcup ((\neg C_1) \sqcap C_3),$$

similarly to Jaeger's probability formula $F_{C_1} F_{C_2} + (1 - F_{C_1}) F_{C_3}$.

Now consider the final construct in Jaeger's relational Bayesian networks; that is, combination functions. We do not have them in full generality here. For a fixed and finite domain, the node $\exists r.C$ is in fact a Noisy-OR: the indicator function $\mathbb{1}_{(\exists r.C)(x)}$ is a disjunction of all conjunctions of nodes $C(y)$ with inhibitor nodes $r(x, y)$, for all $y \in \mathcal{D}$. Likewise a node $\forall r.C$ can be written as a conjunction of implications.

Hence, as far as finite fixed domains are concerned, the syntax for CR$\mathcal{ALC}$ networks presented here can be viewed as syntactic sugar for relational Bayesian networks that only contain unary and binary relations, where binary relations have no parents, and where combination functions are restricted to Noisy-OR. What reasons can we offer to study such a subset of relational Bayesian networks? At the risk of repeating arguments already stated, we offer the following answers. First, the syntax and semantics of CR$\mathcal{ALC}$ networks are much easier to grasp and to use than general Credal $\mathcal{ALC}$, and also easier to grasp than general relational Bayesian networks — in our experience, the syntax and semantics of Jaeger's probability functions are somewhat difficult for the novice. Second, techniques honed by current research for both description logics and relational Bayesian networks can be used. Third, theoretical results about description logics can lead to novel interesting results about relational Bayesian networks and related languages. We do not focus on the latter topic, as it has been examined before [7].

3.3 A Few Applications

We now briefly present examples of $\text{CR}\mathcal{ALC}$ networks that have been described in previous publications.

First, Polastro and Correa [55] have examined the use of $\text{CR}\mathcal{ALC}$ networks in robot navigation, a task where high level reasoning is important [17,25]. Deterministic facts were encoded in sentences such as

$$\mathsf{Office} \equiv \mathsf{Room} \sqcap \exists \mathsf{contains}.\mathsf{Desk} \sqcap \exists \mathsf{contains}.\mathsf{Cabinet} \sqcap \exists \mathsf{contains}.\mathsf{Monitor}$$

together with probabilistic assessments obtained by experimental analysis. During operation, images were collected and inferences were run; for instance, an image containing 3 chairs, 1 table, 1 monitor, 1 cabinet and 1 door was taken and the inference $\mathbb{P}(\mathsf{Office}|\text{detected objects}) = 0.4278$ was computed. A similar application of $\text{CR}\mathcal{ALC}$ networks in spatial reasoning has been reported by Fenelon et al. [15].

A second example is link prediction, where the goal is to predict whether there is a link between two nodes in a social network [40]. Description logics are particularly well suited to handle social networks, because they deal with concepts (applied to single nodes) and roles (applied to pairs of nodes). Ochoa-Luna, Revoredo and Cozman [52] employed a $\text{CR}\mathcal{ALC}$ network to predict links in a social network consisting of researchers, where links indicate co-authorship. Approximate variational inference was used to compute the probability of various links between individuals. The combination of topological features and probabilistic reasoning led to accuracy of 89.5 % in link prediction, compared to 85.6 % accuracy with topological features only.

4 Approximate Variational Inference with $\text{CR}\mathcal{ALC}$ Networks

Given a $\text{CR}\mathcal{ALC}$ network, a finite domain $\mathcal{D}$, and an Abox $\mathcal{A}$, consider the query $Q = \mathbb{P}(C(a)|\mathcal{A})$. We refer to the computation of Q as an *inference*. One obvious idea is to ground the $\text{CR}\mathcal{ALC}$ network into a Bayesian network, and compute Q there by any existing algorithm [16,74]. However, the size of the grounded Bayesian network may become too large as the size of the domain grows. A solution is to approximate Q. One can ground the $\text{CR}\mathcal{ALC}$ network and run approximate inference there. It is not even necessary to generate the whole grounded Bayesian network, but only the groundings that are relevant to the query. However, for large domains it may be difficult to even generate the grounded network. A promising alternative is to avoid grounding concepts and roles, by noting that many of these grounding lead to identical computations.

So, consider running belief propagation [36] in the ground network. As many messages are actually identical, it makes sense to group them [32,68]. An additional step is to group nodes and run exact inference within each group, and to treat each group as a single variable as far as messages are concerned. That is, we have belief propagation among groups, and exact inference within groups; the messages can be derived using a variational scheme [75].

The idea then is to take the CR$\mathcal{ALC}$ network and to divide the ground Bayesian network in *slices*. Each slice congregates groundings with respect to a single individual: assertion $C(a)$ clearly belongs to the slice of a, and assertion $r(a, y)$ belongs to the slice of a for any $y \in \mathcal{D}$. For instance, in Fig. 1 we have two slices, one for individual a and another for individual b. We now consider messages in a variational scheme where each slice is a group [75]; to do so, we introduce a node for each function in the Bayesian network, and connect this node to the variables that affect the function (that is, we create the *factor graph*). Now we have messages sent to and from these function-nodes; for instance, suppose that node $A(a)$ is to send a message to the function-node that represents $f(c)$ where c is some individual that appears in the evidence. This message is $\prod_g m_{g,A(a)}(A(a))$, where $m_{g,A(a)}$ are the messages sent to $A(a)$ from function-nodes g, as g sweeps through the set of function-nodes connected to $A(a)$ except f. Similar messages are then sent to $A(a)$, and so on. Note that as we are grouping variables in each slice, and assuming that exact inference is run inside a slice, only messages between slices must be exchanged.

Now if all messages were exchanged amongst all possible slices until convergence, we would have a variant of belief propagation run in the fully grounded Bayesian network. But note that many slices are exchangeable. In particular, every element of the domain for which there is no evidence leads to an identical slice. Thus we can put together all such slices: everytime a message is to be sent from this "combined" slice, we simple compute the message that would be sent by a single slice, and raise it to the number of slices in the "combined" slice.

To summarize, approximate inferences are produced by generating a set of grounded Bayesian networks, one for each slice mentioned in the query and in the evidence, plus an additional Bayesian network for a "generic" individual; a detailed description of this algorithm can be found in Ref. [7]. Exact Bayesian network inference is performed in each one of these networks and messages are exchanged between the networks. The whole scheme resembles the RCR framework [73]. Experiments indicate that this approach leads to fast and accurate approximations [7].

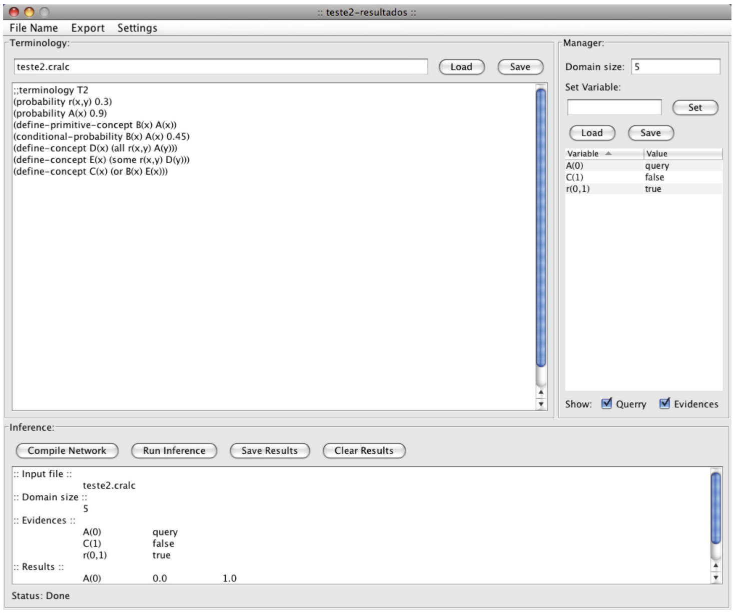
A package implementing this algorithm has been coded by the second author using the Java language (version 6.0), and can work either from the command prompt or through a graphical user interface.[3] Once the package is downloaded and uncompressed, it must be run together with the jar file JavaBayes.jar (in the folder Proj01\libs). The distribution comes with the file CrALC.java; this file must be compiled and run.

[3] The package is freely available, in compressed form, at the site http://sites.poli.usp.br/pmr/ltd/Software/CRALC/inf-cralc-v21may2012.zip.

The package can be run from the command prompt with some additional parameters:

- gui {on|off}: loads (or not) the graphical user interface (default is on);
- i input_file N: loads terminology in input_file and sets domain size to N;
- p output_file: saves a ground Bayesian network into output_file;
- e query_file: loads query and evidence in query_file;
- r name: saves inference in file name.txt, and each slice in auxiliary file.

To describe CR$\mathcal{ALC}$ networks as input, we have chosen to adapt the Knowledge Representation System Specification (KRSS).[4] We use the following constructs: `(and C1...Cn)` for conjunction; `(or C1...Cn)` for disjunction; `(not C)` for complement; `(all r C)` to indicate the quantifier $\forall r.C$; `(some r C)` to indicate the quantifier $\exists r.C$; `(define-concept C D)` for $C \equiv D$. The package also allows `(define-primitive-concept C D)` for $C \sqsubseteq D$, in case the user wishes to use this construct (note that doing so is somewhat risky as it may generate a terminology that cannot be grounded into a unique Bayesian network). Probabilistic assessments are specified as `(probability B` α`)`, denoting $P(B) = \alpha$. The package also allows `(conditional-probability B A` α`)` for $P(A|B) = \alpha$, even though such an assessment is not strictly necessary in CR$\mathcal{ALC}$ networks.



teste2.cralc

```
;;terminology T2
(probability r(x,y) 0.3)
(probability A(x) 0.9)
(define-primitive-concept B(x) A(x))
(conditional-probability B(x) A(x) 0.45)
(define-concept D(x) (all r(x,y) A(y)))
(define-concept E(x) (some r(x,y) D(y)))
(define-concept C(x) (or B(x) E(x)))
```

Domain size: 5

Variable	Value
A(0)	query
C(1)	false
r(0,1)	true

```
:: Input file ::
            teste2.cralc
:: Domain size ::
            5
:: Evidences ::
            A(0)        query
            C(1)        false
            r(0,1)      true
:: Results ::
            A(0)        0.0         1.0
```

Status: Done

Fig. 7. Terminologies are written in the larger panel, while assertions are set in the right panel; the lower panel reports on inferences.

[4] The standard specification of KRSS can be found at http://dl.kr.org/krss-spec.ps.

An example of valid input file is:

```
;; This is a comment...
(probability A(x) 0.7)
(probability B(x) 0.4)
(define-concept C(x) (and A(x) (not B(x))))
```

Assertions can be inserted through simple files as well; for instance:

```
A(1)    query
B(0)    true
r(0,1)  false
```

The graphical user interface depicted in Fig. 7 can be used to load/save files, to specify the size of the domain and the assertions, to ask for inferences, and to check results. The interface lets the user insert domain size and assertions. Each assertion is inserted either as $C(i)$, where C is a concept and i is an integer, or as $r(i, j)$, where r is a role and i and j are integers.

5 Exact Lifted Inference with CR$\mathcal{ALC}$ Networks

Because any CR$\mathcal{ALC}$ network is a relational Bayesian network, we can use any algorithm that runs *lifted* inference in relational Bayesian networks. Here lifted means that inference is performed without completely grounding all concepts and roles [58]. There has been significant effort in exact lifted inference [10–12,31,47,70,72]. In our case we are interested in lifted inference in the presence of "aggregation parfactors" [33]; that is, in the presence of terms that aggregate the effect of many random variables, such as quantifiers.

Kisynski and Poole's AC-FOVE algorithm, an extension of first-order variable elimination [33,34], is currently the state-of-art for lifted inference with aggregation parfactors. We still focus on finite domains, and in this case each construct in a CR$\mathcal{ALC}$ network can be translated either into parameterized functions or into aggregation parfactors. For instance, an existential quantifier $\exists r.C$ can be encoded into an aggregation parfactor that yields 1 when the number of instances of $r(x, y) \wedge C(y)$ is larger than one, and 0 otherwise. AC-FOVE then applies a set of rules to the functions and parfactors. Each rule transforms a function or parfactor until only the query is present. The basic rule is *lifted elimination*, where all instances of a relation (concept or role, in our setting) are eliminated at once, without any actual grounding. For lifted elimination to be applied, several conditions must be met. When these conditions do not hold, AC-FOVE has several options: it can split groundings into groups, unify several groundings, multiply functions after unification, or exponentiate probabilities so as to account for exchangeable elements of the domain. Finally, AC-FOVE can use *counting formulas* [47]; that is, it can use random variables whose values indicate how many individuals satisfy a given condition (rather than dealing with all individuals separately). AC-FOVE is a greedy algorithm that chooses, at each step, one of these operations, resorting to grounding as a last resource.

A package implementing the AC-FOVE algorithm has been coded by the third author using the Java language (version 6.0). The package consists of an API that must be called as needed. Not only the AC-FOVE algorithm is implemented, but also the variable elimination and the C-FOVE algorithms.[5]

To illustrate, consider a few code fragments. First, the construction of a domain:

```
// Creates a list of individuals
List<Constant> individuals = new ArrayList<Constant>();
individuals.add(Constant.getInstance("a1"));
individuals.add(Constant.getInstance("a2"));
individuals.add(Constant.getInstance("a3"));
// Creates a population based on the list
Population popul  = Population.getInstance(individuals);
// Creates a logical variable bound to the population
LogicalVariable x = StdLogicalVariable.getInstance("x", popul);
```

An existential quantifier leads to an aggregation parfactor as follows:

```
Parfactor g = new AggParfactorBuilder(p, c, Or.OR).build();
```

Assertions can be inserted as evidence:

```
List<BigDecimal> fEvidence = TestUtils.toBigDecimalList(0.0, 1.0);
Parfactor evidence = new StdParfactorBuilder().variables(sprinkler).
values(fEvidence).build();
```

Finally, to run the algorithm:

```
ACFOVE acfove = new ACFOVE(input);
Parfactor result = acfove.run();
```

6 Conclusion

In this paper we have presented a syntax for CR$\mathcal{ALC}$ networks, with two goals in mind. First, it simplifies knowledge representation when probabilistic assessments must be coupled with the $\mathcal{ALC}$ logic. Second, it offers a description language for a useful class of relational Bayesian networks. The resulting language avoids many complexities of CR$\mathcal{ALC}$ and of general relational Bayesian networks and can be easily grasped by a user.

We also described freely available packages that implement approximate and exact inference for CR$\mathcal{ALC}$ networks. The software packages we have presented still require substantial development, but they are steps in a direction we feel has not received enough attention. As efficient inference is a key to combinations of uncertainty and semantic information, we hope that these efforts may be useful in future applications. Clearly there are many paths for future work; for instance, the study of open-world reasoning, infinite domains, and interval probabilities.

[5] The package is freely available at https://github.com/ftakiyama/AC-FOVE, where source code and examples can be found.

Acknowledgements. The first author was partially supported by CNPq. The second author was supported by FAPESP. The work reported here has received substantial support by FAPESP grant 2008/03995-5.

References

1. Augustin, T., Coolen, F.P.A., de Cooman, G., Troffaes, M.C.M.: Introduction to Imprecise Probabilities. Wiley, Chichester (2014)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: Description Logic Handbook. Cambridge University Press, Cambridge (2002)
3. Boole, G.: The Laws of Thought. Dover edition, New York (1958)
4. Borgida, A.: On the relative expressiveness of description logics and predicate logics. Artif. Intell. **82**(1–2), 353–367 (1996)
5. Costa, P.C.G., Laskey, K.B.: Of Klingons and starships: Bayesian logic for the 23rd century. In: Conference on Uncertainty in Artificial Intelligence (2005)
6. Costa, P.C.G., Laskey, K.B.: PR-OWL: a framework for probabilistic ontologies. In: Conference on Formal Ontology in Information Systems (2006)
7. Cozman, F.G., Polastro, R.B.: Complexity analysis and variational inference for interpretation-based probabilistic description logics. In: Conference on Uncertainty in Artificial Intelligence, pp. 117–125. AUAI Press (2009)
8. d'Amato, C., Fanizzi, N., Lukasiewicz, T.: Tractable reasoning with Bayesian description logics. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 146–159. Springer, Heidelberg (2008)
9. de Finetti, B.: Theory of Probability, vol. 1–2. Wiley, New York (1974)
10. de Salvo Braz, R., Amir, E., Roth, D.: Lifted first-order probabilistic inference. In: International Joint Conference in Artificial Intelligence (2006)
11. de Salvo Braz, R., Amir, E., Roth, D.: Lifted first-order probabilistic inference. In: Getoor, L., Taskar, B. (eds.) An Introduction to Statistical Relational Learning, pp. 433–451. MIT Press, Cambridge (2007)
12. de Salvo Braz, R., Amir, E., Roth, D.: A survey of first-order probabilistic models. In: Holmes, D.E., Jain, L.C. (eds.) Innovations in Bayesian Networks. Studies in Computational Intelligence, pp. 289–317. Springer, Heidelberg (2008)
13. Ding, Z., Peng, Y., Pan, R.: BayesOWL: uncertainty modeling in semantic web ontologies. In: Ma, Z. (ed.) Soft Computing in Ontologies and Semantic Web, pp. 3–29. Springer, Heidelberg (2006)
14. Dürig, M., Studer, T.: Probabilistic ABox reasoning: preliminary results. In: Description Logics, pp. 104–111 (2005)
15. Fenelon, V., Hummel, B., Santos, P.E., Cozman, F.G.: Encoding spatial domains with relational Bayesian networks. In: Spatio-temporal Dynamics Workshop, pp. 49–54 (2010)
16. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: International Joint Conference on Artificial Intelligence, pp. 1300–1309 (1999)
17. Galindo, C., Fernandez-Madrigal, J.-A., Gonzalez, J., Saffiotti, A.: Robot task planning using semantic maps. Robot. Auton. Syst. **11**, 955–966 (2008)
18. Getoor, L., Friedman, N., Koller, D., Taskar, B.: Learning probabilistic models of relational structure. In: International Conference on Machine Learning, pp. 170–177 (2001)
19. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007)

20. Giugno, R., Lukasiewicz, T.: P-SHOQ(D): a probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web. In: Flesca, S., Greco, S., Leone, N., Lanni, G. (eds.) European Conference on Logics in Artificial Intelligence, pp. 86–97 (2002)
21. Gutierrez-Basulto, V., Jung, J.C., Lutz, C., Schröder, L.: A closer look at the probabilistic description logic prob-$\mathcal{EL}$. In: Burgard, W., Roth, D. (eds.) Conference on Artificial Intelligence, pp. 197–202 (2011)
22. Halpern, J.Y.: An analysis of first-order logics of probability. Artif. Intell. **46**, 311–350 (1990)
23. Halpern, J.Y.: Reasoning about Uncertainty. MIT Press, Cambridge (2003)
24. Heinsohn, J.: Probabilistic description logics. In: Conference on Uncertainty in Artificial Intelligence, pp. 311–318 (1994)
25. Hertzberg, J., Saffiotti, A.: Using semantic knowledge in robotics. Robot. Auton. Syst. **56**, 875–877 (2008)
26. Hung, E., Getoor, L., Subrahmanian, V.S.: Probabilistic interval XML. ACM Trans. Comput. Logic **8**(4), 1–38 (2007)
27. Jaeger, M.: Probabilistic reasoning in terminological logics. In: Principles of Knowledge Representation, pp. 461–472 (1994)
28. Jaeger, M.: Relational Bayesian networks. In: Geiger, D., Shenoy, P.P. (eds.) Conference on Uncertainty in Artificial Intelligence, pp. 266–273. Morgan Kaufmann (1997)
29. Jaeger, M.: Complex probabilistic modeling with recursive relational Bayesian networks. Ann. Math. Artif. Intell. **32**, 179–220 (2001)
30. Jaeger, M.: Relational Bayesian networks: a survey. Linkoping Electronic Articles in Computer and Information Science, 6 (2002)
31. Kersting, K.: Lifted probabilistic inference. In: De Raedt, L., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P. (eds.) European Conference on Artificial Intelligence. IOS Press (2012)
32. Kersting, K., Ahmadi, B., Natarajan, S.: Counting belief propagation. In: Conference on Uncertainty in Artificial Intelligence. AUAI Press (2009)
33. Kisynski, J.J., Poole, D.: Lifted aggregation in directed first-order probabilistic models. In: International Joint Conference on Artificial Intelligence, pp. 1922–1929 (2009)
34. Kisynski, J.J.: Aggregation and constraint processing in lifted probabilistic inference. Ph.D. thesis, Computer Science, University of British Columbia (2010)
35. Klinov, P., Parsia, B.: A hybrid method for probabilistic satisfiability. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 354–368. Springer, Heidelberg (2011)
36. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge (2009)
37. Koller, D., Pfeffer, A.: Object-oriented Bayesian networks. In: Conference on Uncertainty in Artificial Intelligence, pp. 302–313 (1997)
38. Koller, D., Pfeffer, A.: Probabilistic frame-based systems. In: AAAI, pp. 580–587 (1998)
39. Laskey, K.B.: MEBN: a language for first-order Bayesian knowledge bases. Artif. Intell. **172**(2–3), 140–178 (2008)
40. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. J. Am. Soc. Inform. Sci. Technol. **7**(58), 1019–1031 (2007)
41. Lukasiewicz, T.: Probabilistic description logic programs. Int. J. Approx. Reason. **45**(2), 288–307 (2007)

42. Lukasiewicz, T.: Expressive probabilistic description logics. Artif. Intell. **172**(6–7), 852–883 (2008)
43. Lukasiewicz, T., Predoiu, L., Stuckenschmidt, H.: Tightly integrated probabilistic description logic programs for representing ontology mappings. Ann. Math. Artif. Intell. **63**(3/4), 385–425 (2011)
44. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. J. Web Semant. **6**, 291–308 (2008)
45. Lutz, C., Schröder, L.: Probabilistic description logics for subjective uncertainty. In: Principles of Knowledge Representation and Reasoning, pp. 393–403. AAAI Press (2010)
46. Mahoney, S., Laskey, K.B.: Network engineering for complex belief networks. In: Conference on Uncertainty in Artificial Intelligence (1996)
47. Milch, B., Zettlemoyer, L.S., Kersting, K., Haimes, M., Kaelbling, L.P.: Lifted probabilistic inference with counting formulas. In: AAAI, pp. 1062–1068 (2008)
48. Ngo, L., Haddawy, P.: Answering queries from context-sensitive probabilistic knowledge bases. Theor. Comput. Sci. **171**(1–2), 147–177 (1997)
49. Niepert, M., Noessner, J., Stuckenschmidt, H.: Log-linear description logics. In: International Joint Conference on Artificial Intelligence (2011)
50. Nilsson, N.J.: Probabilistic logic. Artif. Intell. **28**, 71–87 (1986)
51. Nottelmann, H., Fuhr, N.: Adding probabilities and rules to OWL lite subsets based on probabilistic datalog. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **14**(1), 17–42 (2006)
52. Ochoa-Luna, J.E., Revoredo, K.C., Cozman, F.G.: An experimental evaluation of a scalable probabilistic description logics approach for semantic link prediction. In: International Workshop on Uncertainty Reasoning for the Semantic Web, Shangai, China, pp. 63–74 (2012). http://ceur-ws.org
53. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo (1988)
54. Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press, New York (2000)
55. Polastro, R., Corrêa, F., Cozman, F., Okamoto Jr., J.: Semantic mapping with a probabilistic description logic. In: da Rocha Costa, A.C., Vicari, R.M., Tonidandel, F. (eds.) SBIA 2010. LNCS (LNAI), vol. 6404, pp. 62–71. Springer, Heidelberg (2010)
56. Poole, D.: Probabilistic Horn abduction and Bayesian networks. Artif. Intell. **64**, 81–129 (1993)
57. Poole, D.: The independent choice logic for modelling multiple agents under uncertainty. Artif. Intell. **94**(1/2), 7–56 (1997)
58. Poole, D.: First-order probabilistic inference. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 985–991 (2003)
59. Predoiu, L., Stuckenschmidt, H.: Probabilistic models for the semantic web. In: Ma, Z., Wang, H. (eds.) The Semantic Web for Knowledge and Data Management: Technologies and Practices, pp. 74–105. IGI Global, Hershey (2009)
60. De Raedt, L.: Logical and Relational Learning. Springer, Heidelberg (2008)
61. Rettinger, A., Losch, U., Tresp, V., d'Amato, C., Fanizzi, N.: Mining the semantic web - statistical learning for next generation knowledge bases. Data Min. Knowl. Disc. **24**, 613–662 (2012)
62. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. **62**(1–2), 107–136 (2006)

63. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Epistemic and statistical probabilistic ontologies. In: International Workshop on Uncertainty Reasoning for the Semantic Web, pp. 1–12 (2012)
64. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. J. Artif. Intell. Res. **15**, 391–454 (2001)
65. Schild, K.: A correspondence theory for terminological logics: preliminary report. In: International Joint Conference on Artificial Intelligence, pp. 466–471 (1991)
66. Schmidt-Schauss, M., Smolka, G.: Attributive concept descriptions with complements. Artif. Intell. **48**, 1–26 (1991)
67. Sebastiani, F.: A probabilistic terminological logic for modelling information retrieval. In: Croft, W.B., van Rijsbergen, C.J. (eds.) International ACM Conference on Research and Development in Information Retrieval (SIGIR), Dublin, Ireland, pp. 122–130. Springer, London (1994)
68. Singla, P., Domingos, P.: Lifted first-order belief propagation. In: AAAI, pp. 1094–1099 (2008)
69. Staker, R.: Reasoning in expressive description logics using belief networks. In: International Conference on Information and Knowledge Engineering, pp. 489–495 (2002)
70. Taghipour, N., Fierens, D., Van den Broeck, G., Davis, J., Blockeel, H.: Completeness results for lifted variable elimination. In: International Conference on Artificial Intelligence and Statistics, pp. 572–580 (2013)
71. Thomas, A., Spiegelhalter, D., Gilks, W.: BUGS: a program to perform Bayesian inference using Gibbs sampling. In: Bernardo, J., Berger, J., Dawid, A., Smith, A. (eds.) Bayesian Statistics, vol. 4. Oxford University Press, Oxford (1992)
72. van den Broeck, G.: On the completeness of first-order knowledge compilation for lifted probabilistic inference. In: Neural Processing Information Systems (2011)
73. van den Broeck, G., Choi, A., Darwiche, A.: Lifted relax, compensate and then recover: from approximate to exact lifted probabilistic inference. In: Conference on Uncertainty in Artificial Intelligence (2012)
74. Wellman, M.P., Breese, J.S., Goldman, R.P.: From knowledge bases to decision models. Knowl. Eng. Rev. **7**(1), 35–53 (1992)
75. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free energy approximations and generalized belief propagation algorithms. IEEE Trans. Inf. Theory **51**, 2282–2312 (2005)
76. Yelland, P.M.: Market analysis using a combination of Bayesian networks and description logics. Technical Report SMLI TR-99-78, Sun Microsystems Laboratories (1999)