Learning to Assign Degrees of Belief in Relational Domains

Frédéric Koriche

Machine Learning, Volume 73(1), pages 25-53, 2008

Learning to Assign Degrees of Belief in Relational Domains

Frédéric Koriche

Abstract A recurrent problem in the development of reasoning agents is how to assign degrees of beliefs to uncertain events in a complex environment. The standard knowledge representation framework imposes a sharp separation between learning and reasoning; the agent starts by acquiring a "model" of its environment, represented into an expressive language, and then uses this model to quantify the likelihood of various queries. Yet, even for simple queries, the problem of evaluating probabilities from a general purpose representation is computationally prohibitive. In contrast, this study embarks on the learning to reason (L2R) framework that aims at eliciting degrees of belief in an inductive manner. The agent is viewed as an anytime reasoner that iteratively improves its performance in light of the knowledge induced from its mistakes. Indeed, by coupling exponentiated gradient strategies in learning and weighted model counting techniques in reasoning, the L2R framework is shown to provide efficient solutions to relational probabilistic reasoning problems that are provably intractable in the classical paradigm.

Keywords Learning to reason · Online learning · Relational probabilistic reasoning · Exponentiated gradient learning · Markov networks · Weighted model counting

1 Introduction

From early on (Cox, 1946), it has been recognized that degrees of belief, or epistemic probabilities, play an important role in commonsense reasoning. Consider for example a doctor equipped with a knowledge base containing factual information about patients, as well as general information about symptoms, diseases, diagnostic tests, and medical treatments. In most cases, the knowledge base is not complete enough to logically entail the illness of a particular patient. Since the efficacy of a treatment will almost certainly depend on the disease, it is thus important for the doctor to estimate the likelihood of different possibilities. More generally, if an agent wishes to employ the expected-utility paradigm of decision theory in order to guide its actions,

Frédéric Koriche
LIRMM, Université Montpellier II
161 Rue Ada, 34392 Montpellier Cedex 5 France
E-mail: Frederic.Koriche@lirmm.fr

it must assign probabilities to various assertions. Less obvious, however, is the key question of *how* to elicit such degrees of belief in a computationally effective manner.

The standard knowledge representation approach claims that beliefs are elicited in a *deductive* way: one starts by accepting a certain set of premises, and then accepts the conclusion that follows from the premises in accordance with certain inference rules. Actually, many approaches to uncertain reasoning follow this paradigm by extending logical inference to probabilistic inference. The agent indeed starts by acquiring a compact description of a target probability distribution, and then, utilizes this description to derive the probability of any possible query. The main assumption behind this paradigm is that, in general, the description is acquired independently of the queries that will be posed. To provide compact and reliable descriptions of complex probability measures, various representation formalisms have been proposed in the literature; some of them extend first-order logical representation languages to probabilistic inference (Poole, 1993; Bacchus et al., 1996; Muggleton, 1996; Ngo & Haddawy, 1997; Costa et al., 2003; Kersting, 2006), while others advocate a dual approach by extending graphical representation languages to relational inference (Jaeger, 1997; Friedman et al., 1999; Pfeffer, 2000; Taskar et al., 2002; Richardson & Domingos, 2006). For example, if the domain is represented by a knowledge base in first-order logic, the probability measure is induced by assigning equal likelihood to all interpretations that satisfy the knowledge base; the degree of belief of any query is simply the fraction of those interpretations which are consistent with the query (Bacchus et al., 1996; Halpern, 2003).

From a pragmatic viewpoint, the usefulness of a computational approach to probabilistic reasoning depends both on the *accuracy* of the belief estimates and the *efficiency* of belief estimation. Unfortunately, the task of deducing degrees of belief from a general purpose description is computationally prohibitive. In propositional logic, the problem of inferring the probability of any query from a propositional theory is #P-hard, and even the apparently easier question of approximating this probability in a very weak sense is NP-hard (Roth, 1996). The first-order version of this problem is highly undecidable in general (Abadi & Halpern, 1994). Under finite domain assumptions, though, it remains decidable. Nevertheless, even if any first-order theory defined over a finite domain can be transformed into a logically equivalent ground formula, the size of the resulting formula can grow exponentially with respect to the initial theory. Consequently, relational probabilistic inference turns out to be #Exp-hard to evaluate and NExp-hard to approximate. Similar results have been obtained for graphical representation languages (Jaeger, 2000).

In contrast, this study aims at eliciting degrees of belief in an *inductive* manner. The main departure from the deductive approach is that knowledge is not ascribed *a priori*, for the purpose of describing an environment, but instead acquired *a posteriori*, by experience, in order to improve the agent's ability to reason efficiently in its environment. Specifically, our approach follows the so-called *learning to reason* (L2R) framework that has recently emerged as an active research field of Inductive Logic Programming for dealing with the intractability of reasoning problems (Khardon, 1999; Khardon & Roth, 1997, 1999; Valiant, 1994, 2000a,b).

The inductive view of probabilistic inference is captured by a computational model of learning. The environment, or domain in question, is modeled as a probability distribution $\mathcal{P}$ on a space of relational structures, or interpretations. To acquire knowledge from its environment, the agent is given a "grace period" in which it can interact with its learning interface. The purpose of this learning interface is to help the agent in concentrating its effort toward finding a representation KB of $\mathcal{P}$ that

is useful for evaluating queries in some target query language $\mathcal{Q}$. The reasoning performance is measured only after the grace period, when the agent is presented with new queries from $\mathcal{Q}$ and has to estimate their probability according to its representation KB. Thus, by contrast with the deductive view of probabilistic inference, the agent is not required to achieve optimal performance by evaluating *any* possible query with perfect precision. Instead, the performance is measured with regard to a restricted though expressive query language.

Technically, our framework is based on the online mistake-driven learning model introduced by Littlestone (1988, 1989). In this setting, the L2R protocol is modeled as a repeated game between the reasoning agent and its learning interface. During each trial of the game, the agent receives a query Q from the language $\mathcal{Q}$ and assigns a degree of belief $\mathbf{Pr}_{KB}(Q)$ to it. The agent is charged a mistake only if its prediction loss is not judged satisfactory for the task at hand. In this case, the agent is supplied the correct probability $\mathbf{Pr}_{\mathcal{P}}(Q)$ and updates its knowledge base in light of this feedback information. In essence, the agent is an *anytime reasoner* that gradually improves its performance by interacting with its learning interface.

Based on this framework, the requirements for efficient reasoning are twofold. First, the length of the grace period needed to achieve full functionality must be polynomial in the dimension of the relational vocabulary. In other words, the agent's behavior must converge to yield accurate belief estimates after a polynomial number of interactions. Second, the computational cost needed to estimate the probability of any query from the language $\mathcal{Q}$ must also be polynomial in the input dimension.

To satisfy these desiderata, we develop an online L2R algorithm which combines techniques in regression learning and weighted model counting. The algorithm uses an exponentiated gradient strategy (Kivinen & Warmuth, 1997; Bylander, 1998; Cesa-Bianchi, 1999) adapted for assigning probabilities to relational queries. The cumulative number of mistakes made by the reasoner depends only *logarithmically* in the size of the target probability distribution, and hence *linearly* on the input dimension. Consequently, the learning curve of the reasoner is guaranteed to converge to yield accurate estimations after a polynomial number of interactions.

The key idea behind efficient relational probabilistic reasoning lies in a suitable encoding of the "mistake-driven" knowledge that allows tractable forms of weighted model counting (Sang et al., 2005; Chavira & Darwiche, 2008). Namely, for several restricted conjunctive query languages, the computational cost of assigning degrees of belief is polynomial in the number of mistakes made so far, and hence, the dimension of the input vocabulary. This result highlights the interest of the L2R framework by providing efficient solutions to relational probabilistic reasoning problems that are provably intractable in the classical framework.

The necessary background in relational probabilistic reasoning can be found in section 2. Based on this setting, the core of the paper introduces the L2R framework (section 3), next presents the exponentiated gradient L2R algorithm (section 4), and then examines tractable query languages (section 5). In section 6, we compare our framework with related work by establishing some interesting relationships with knowledge compilation and statistical relational learning. In particular, the connection with Markov logic networks (Richardson & Domingos, 2006) is examined in detail. As a dual contribution of this paper, our results can be viewed as introducing subclasses of Markov networks for which learning and inference are shown tractable. Finally, in section 7, we conclude by listing several perspectives of further research. For the sake of clarity, proofs of Theorems 1 and 2 are given in the Appendix.

2 Relational Probabilistic Reasoning

We consider reasoning problems where the environment is modeled as a probability distribution over a space of relational structures. This section briefly reviews the background for reasoning in such environments.

2.1 Relational Reasoning

A *relational vocabulary* consists in finite set of *relation symbols*, each equipped with its associated arity, and a finite set of *constant symbols*. In addition, a countable set of *variables* is used to construct quantified expressions. As usual, constant symbols represent objects in the domain of interest, and relation symbols represent properties of objects and relationships among objects. A *term* is a constant symbol or a variable.

As complex environments typically involve multiple kinds of objects, it is often useful to rely on a many-sorted vocabulary (Manzano, 2005), whose main advantage is to concisely represent the background knowledge about different sorts of objects. From this perspective, we shall assume that any relational vocabulary also includes a finite set of *sorts*. Each term is given a sort s, and each k-ary relation symbol is given a k-tuple of sorts $(s_1, \cdots, s_k)$. Sorts are essentially what are known as simple types in connection with programming languages; they represent indivisible kinds of entities. For example, the variable x might range over people (e.g. Ann, Bob, etc.), the constant C might represent a city (e.g. Seattle), and the binary relation symbol $\mathsf{At}(x, y)$ might indicate the location of person x at city y. Notice that an *untyped* vocabulary is just a vocabulary that contains a single sort.

An *atom* is an expression of the form $\mathsf{R}(t_1, \cdots, t_k)$, where R is a k-ary relation symbol and each t_i is a term of appropriate sort s_i in the type of R. A ground atom is an atom with no variables. *Formulas* are constructed in the usual way from the atoms, the connectives $\neg$ and $\wedge$, the quantifier $\forall$, and the logical constant $\top$.[1] *Sentences* are, as usual, formulas with no free variables. Unless stated otherwise, any formula under consideration in this study is a sentence. The *size* $|F|$ of a formula F is the number of occurrences of all relation symbols, constant symbols, and variables in its description. For example, the size of $\exists x \exists y \mathsf{At}(x, y)$ is 3.

The *Herbrand base* of a relational vocabulary, is the set $\mathcal{H}$ of all ground atoms that can be constructed from the relation symbols and the constant symbols according to their appropriate sorts in the vocabulary. The *dimension* d of a relational vocabulary is defined as the cardinality of its Herbrand base. Notably, if the vocabulary contains m relation symbols R_i or arity a_i, and n constant symbols, then $d \leq \sum_{i=1}^{m} n^{a_i}$. Clearly, d is equal to $\sum_{i=1}^{m} n^{a_i}$ when the vocabulary is reduced to a single sort. In this case, the Herbrand base is *full* in the sense that it includes all possible ground atoms formed from the relation symbols and constant symbols of the vocabulary.

Semantics is given to formulas using *relational structures* or *interpretations*. An interpretation I consists of a set of objects D, called the domain, and a way of associating each of the elements of the vocabulary the corresponding entities over the domain. Thus, a sort s is associated with a subset D_s of D, a constant symbol c of sort s is associated with an object of D_s, and a k-ary relation symbol R of type $(s_1, \cdots, s_k)$ is associated with a k-ary relation over $D_{s_1}, \cdots, D_{s_k}$. In this study,

[1] Other logical connectives like $\vee$, $\rightarrow$ and $\leftrightarrow$, and the quantifier $\exists$, are defined in the usual way in terms of $\neg$, $\wedge$, $\forall$, and $\top$.

we shall focus on the common class of *Herbrand interpretations* which assumes that the domain is precisely the set of constants of the vocabulary. Thus, a Herbrand interpretation is just a subset of $\mathcal{H}$. The set $\wp(\mathcal{H})$ of all Herbrand interpretations generated from $\mathcal{H}$ is called the *Herbrand space* of the vocabulary.

The notion of *model* is defined by recursion on the structure of formulas. As usual, I is always a model of $\top$. If F is a ground atom, then I is a model of F iff $F \in I$. If F is $\neg G$, then I is a model of F iff I is not a model of G. If F is $G_1 \wedge G_2$, then I is a model of F iff I is a model of both G_1 and G_2. Finally, if F is $\forall x\, G(x)$ where x is of sort s, then I is a model of F iff I is a model of $G(c)$ for every constant symbol c of sort s. The set of all models of F is denoted $\mathcal{M}(F)$.

Given a formula F, we say that F is *satisfiable* if $\mathcal{M}(F)$ is nonempty. Given two formulas F and G, we say that F *entails* G, denoted $F \models G$, if $\mathcal{M}(F) \subseteq \mathcal{M}(G)$. Furthermore, we say that F is *equivalent* to G, denoted $F \equiv G$, if $F \models G$ and $G \models F$.

2.2 Probabilistic Reasoning

In presence of uncertainty, logical entailment is often insufficient to determine the truth of a formula F; both F and its negation may be consistent with our model of the environment. In order to quantify uncertainty, we need to evaluate the probability that F is true in the environment. From this viewpoint, it is useful to represent the Herbrand space of a relational vocabulary by an indexed set $\{I_1, \cdots, I_N\}$, where $N = 2^d$ and d is the dimension of the vocabulary. Thus, a probability distribution over the Herbrand space $\wp(\mathcal{H})$ is specified by a N-tuple $\mathcal{P} = (p_1, \cdots, p_N)$ such that $p_i \in [0, 1]$ and $\sum_{i=1}^{N} p_i = 1$. The value p_i is the probability of the interpretation I_i according to the probability distribution $\mathcal{P}$.

The *projection* of a formula F onto the Herbrand space $\wp(\mathcal{H})$ is a N-tuple of features $\boldsymbol{\phi}(F) = (\phi_1(F), \cdots, \phi_N(F))$ where $\phi_i(F) = 1$ if I_i is a model of F and $\phi_i(F) = 0$ otherwise. Based on this notation, the probability that a formula F is true according to a probability distribution $\mathcal{P}$ over the Herbrand space $\wp(\mathcal{H})$ is

$$\mathbf{Pr}_{\mathcal{P}}(F) = \sum_{i=1}^{N} p_i \phi_i(F)$$

Definition 1 A *relational probabilistic reasoning problem* defined over a relational vocabulary is a pair $(\mathcal{P}, \mathcal{Q})$, where $\mathcal{P}$ is a probability distribution over the Herbrand space of the vocabulary, and $\mathcal{Q}$ is a countable set of formulas Q over the vocabulary. $\mathcal{P}$ is called the *environment* and $\mathcal{Q}$ the *query language*.

Example 1 Let us consider the *random blocks* domain introduced by Chavira et al. (2006). It describes the random placement of some blocks, viewed as obstacles, on the locations of a two-dimensional map. The domain involves two sorts of entities: blocks and locations. The spatial relationship among locations is described by two binary predicates $\mathsf{Left}(y_1, y_2)$, indicating that location y_1 is left of location y_2, and $\mathsf{Above}(y_1, y_2)$, indicating that location y_1 is above location y_2. The vocabulary also involves two other predicates: $\mathsf{At}(x, y)$, indicating the location y of a block x, and $\mathsf{Connected}(y_1, y_2)$ which describes whether, after the placement of the obstacles, there is an unblocked path between y_1 and y_2. The purpose of the agent is to identify the underlying "scene" by estimating the probability of various queries. For example, a query like $\exists y\, \mathsf{Connected}(y, \mathsf{l}_1) \wedge \forall x\, \neg \mathsf{At}(x, \mathsf{l}_1)$ might express the possibility that there is an unblocked path to reach l_1 given that this location is free. With an instance of the domain involving 4 blocks and 8 locations, the dimension is $(3 \times 64) + 32 = 224$.

Example 2 Let us turn to a variant of the so-called *blood type* domain introduced by Friedman et al. (1999). Occasionally, a person is unavailable for testing and yet, because of many reasons such as the clarification of crime, paternity test, or allocation of frozen semen, it is necessary to estimate the blood type of the person. The domain involves four kinds of entities: individuals, blood types $\{\mathsf{A}, \mathsf{B}, \mathsf{AB}, \mathsf{O}\}$, gene copies $\{1, 2\}$, and alleles $\{\mathsf{a}, \mathsf{b}, \mathsf{o}\}$. Each person x has a blood type y, represented by $\mathsf{Blood}(x, y)$. The blood type is a phenotype expressed by the two copies of the *abo* gene. Each copy z has an allele u, represented by $\mathsf{Gene}(x, z, u)$. The first copy is inherited from the mother $\mathsf{Mother}(x, x_1)$, and the second copy is inherited from the father $\mathsf{Father}(x, x_2)$. In this setting, the purpose of the agent is to estimate blood types from the parenthood context. To this end, we would like to express queries such as $\mathsf{Blood}(\mathsf{Ann}, \mathsf{A}) \wedge \mathsf{Mother}(\mathsf{Ann}, \mathsf{Mary}) \wedge \exists z\, \mathsf{Gene}(\mathsf{Mary}, z, \mathsf{a})$ indicating that the blood type of Ann is A given that one *abo* gene of her mother, Mary, has allele a. With and instance of blood type domain involving 10 individuals, the dimension of the vocabulary is $40 + 60 + (2 \times 100) = 300$.

3 The Learning to Reason Framework

In the learning to reason paradigm, the goal of the agent is to acquire a representation KB of the environment $\mathcal{P}$ for the query language $\mathcal{Q}$. From this perspective, the agent is given access to a learning interface that governs the occurrences of queries drawn from $\mathcal{Q}$. The interface most appropriate in our setting is a variant of the reasoning query oracle (Khardon & Roth, 1997, 1999) adapted for belief estimation. In essence, the oracle provides training queries and helps the agent in correcting its mistakes by supplying correct probabilities. To assess regression discrepancies, we use the typical quadratic loss function $L(x, y) = (x - y)^2$.

Definition 2 A *reasoning query oracle* for the problem $(\mathcal{P}, \mathcal{Q})$, with respect to a tolerance parameter $\gamma \in (0, 1]$, denoted $\mathrm{RQ}_\gamma(\mathcal{P}, \mathcal{Q})$, is an oracle that when accessed performs the following protocol with the reasoning agent A. *(1)* The oracle picks an arbitrary query Q in $\mathcal{Q}$ and returns it to A. *(2)* The agent A estimates the uncertainty of Q by assigning a degree of belief $\mathbf{Pr}_{KB}(Q)$. *(3)* The oracle responds by *"correct"* if $L(\mathbf{Pr}_{KB}(Q), \mathbf{Pr}_{\mathcal{P}}(Q)) \leq \gamma$, and *"incorrect"* otherwise. In case of mistake, the oracle also supplies the correct probability $\mathbf{Pr}_{\mathcal{P}}(Q)$ to A.

The interaction protocol is modeled as a repeated game between the agent and its interface. During each $t \geq 1$, the agent receives a query Q_t supplied by the oracle, makes a prediction according to its knowledge base KB_t and, in case of mistake, updates KB_t in light of the feedback information. A reasoning agent is *conservative* if it modifies its knowledge base only when it makes a mistake; in other words, a conservative agent leaves unchanged its hypothesis KB_t whenever any prediction using it is correct. The *mistake bound* for an agent A on the problem $(\mathcal{P}, \mathcal{Q})$, denoted $\mathrm{M}_{A,\gamma}(\mathcal{P}, \mathcal{Q})$, is the maximum number of mistakes that A can make by interacting with $\mathrm{RQ}_\gamma(\mathcal{P}, \mathcal{Q})$ over any arbitrary sequence of queries drawn from $\mathcal{Q}$.

Definition 3 Given a relational probabilistic reasoning problem $(\mathcal{P}, \mathcal{Q})$ defined over a vocabulary of dimension d, an algorithm A is an *efficient mistake bound learning to reason (MB-L2R)* algorithm for $(\mathcal{P}, \mathcal{Q})$, if there are polynomials p and q such that the following conditions hold for any parameter $\gamma \in (0, 1]$: (1) $\mathrm{M}_{A,\gamma}(\mathcal{P}, \mathcal{Q}) \leq p(d, \frac{1}{\gamma})$, and (2) A evaluates any query $Q \in \mathcal{Q}$ of size n in $q(n, d, \frac{1}{\gamma})$ time.

The online learning to reason framework provides a natural way to make explicit the dependence of the reasoning performance on the knowledge acquired from the environment. After a "grace period" of interaction between the agent and its oracle, the agent is expected to evaluate any query supplied by its interface without the help of the feedback response. Of course, we cannot force the agent to make all the mistakes within the grace period. However, we can estimate the asymptotic behavior of a conservative MB-L2R agent by converting it into a *Probably Approximately Correct (PAC)* L2R algorithm. In this setting, we make the assumption that all the queries supplied by $\mathrm{RQ}_\gamma(\mathcal{P},\mathcal{Q})$ are drawn independently at random according to a fixed, but unknown, probability distribution $\mathcal{D}$. Given parameters γ and ϵ, we say that a knowledge base KB is an (ϵ,γ)-*good hypothesis* if the probability of making a mistake when KB is used to predict on any query Q taken at random from $\mathcal{Q}$ according to $\mathcal{D}$ is at most ϵ, that is, $\mathbf{Pr}_{Q\sim\mathcal{D}}[L(\mathbf{Pr}_{KB}(Q),\mathbf{Pr}_{\mathcal{P}}(Q))>\gamma]\leq\epsilon$.

Definition 4 Given a relational probabilistic reasoning problem $(\mathcal{P},\mathcal{Q})$ defined over a vocabulary of dimension d, an algorithm A is an *efficient PAC-L2R* algorithm for $(\mathcal{P},\mathcal{Q})$, if there are polynomials p and q such that the following conditions hold for any probability distribution $\mathcal{D}$ on $\mathcal{Q}$ and any parameters $\delta,\epsilon\in(0,\frac{1}{2}]$ and $\gamma\in(0,1]$: (1) if A is supplied at least $p(d,\frac{1}{\delta},\frac{1}{\epsilon},\frac{1}{\gamma})$ queries that are drawn independently at random according to $\mathcal{D}$ then, with probability at least $1-\delta$, A returns an (ϵ,γ)-good hypothesis, and (2) A evaluates any query $Q\in\mathcal{Q}$ of size n in $q(n,d,\frac{1}{\gamma})$ time.

The simple conversion method due to Angluin (1988) can be adapted to our framework without modifying the online behavior of the agent. We use a variant of this method which takes into account the fact that, for an efficient MB-L2R algorithm, the total number of mistakes is bounded by a known polynomial p. The method takes as input a sample size $s=\frac{1}{\epsilon}(\ln\frac{m}{\delta})$, where $m=p(d,\frac{1}{\gamma})+1$, and converts A into a PAC analogue A_{PAC} as follows: A_{PAC} starts with KB_1 and calls $\mathrm{RQ}_\gamma(\mathcal{P},\mathcal{Q})$ exactly s times by keeping the same hypothesis. If all queries were correctly predicted during this period, then A_{PAC} halts and returns KB_1. Otherwise, A_{PAC} picks the first query that has led to a mistake together with its feedback information, and utilizes the update rule of A in order to generate a new hypothesis KB_2. Then KB_2 is tested s times, and so on. In essence, the algorithm A_{PAC} is a "cautious" variant of A that evaluates its current hypothesis on a fixed period before generating a new one.

Proposition 1 *Let $(\mathcal{P},\mathcal{Q})$ a relational probabilistic reasoning problem. Then, any efficient conservative MB-L2R algorithm A for $(\mathcal{P},\mathcal{Q})$ can be transformed into a an efficient PAC-L2R algorithm A_{PAC} for $(\mathcal{P},\mathcal{Q})$.*

Proof Since A is an efficient MB-L2R algorithm, there exists a polynomial p such that $\mathrm{M}_{A,\gamma}(\mathcal{P},\mathcal{Q})\leq p(d,\frac{1}{\gamma})$. Moreover, since A is conservative, A_{PAC} will generate at most $m=p(d,\frac{1}{\gamma})+1$ hypotheses $KB_1,\cdots,KB_m$. Now, suppose that none of these descriptions is an (ϵ,γ)-good hypothesis. Since each KB_i has error greater than ϵ, any query is correctly evaluated by KB_i with probability $(1-\epsilon)$. Thus, s independent random queries are correctly evaluated by KB_i with probability at most $(1-\epsilon)^s$. Since the probability of a union of events is at most the sum of their individual probabilities, the probability that all s queries are correctly evaluated by any of the hypotheses in $KB_1,\cdots,KB_m$ is at most $m(1-\epsilon)^s\leq m\,e^{-\epsilon s}$. Taking $s=\frac{1}{\epsilon}(\ln\frac{m}{\delta})$, the algorithm A_{PAC} is therefore guaranteed, with probability $1-\delta$, to return an (ϵ,γ)-good hypothesis by making at most $\frac{m}{\epsilon}(\ln\frac{m}{\delta})$ calls to $\mathrm{RQ}_\gamma(\mathcal{P},\mathcal{Q})$. Finally, since there exists a polynomial q such that A evaluates any query $Q\in\mathcal{Q}$ of size n in $q(n,d,\frac{1}{\gamma})$ time, by construction, A_{PAC} is also guaranteed to evaluate Q in $q(n,d,\frac{1}{\gamma})$ time. □

The most important message to be gleaned from the L2R framework is that the behavior of the reasoning agent is essentially governed by the target environment $\mathcal{P}$, the query language $\mathcal{Q}$, and the tolerance factor γ. To this very point, the choice of γ is crucially dependent on the domain in question. Consider for example an event such as $\mathsf{Blood}(\mathsf{Ann}, \mathsf{A}) \wedge \mathsf{Mother}(\mathsf{Ann}, \mathsf{Mary}) \wedge \forall z\, \mathsf{Gene}(\mathsf{Mary}, z, \mathsf{b})$. If γ is set to $\frac{1}{4}$, then any degree of belief less or equal than 50% will be judged satisfactory for this event, even if it is genetically impossible! On the other hand, if γ is set to $\frac{1}{2500}$, then any belief greater than 2% will be treated as a mistake.

4 Exponentiated Gradient Learning to Reason

The main idea behind this study is to combine exponentiated gradient strategies in online learning and weighted model counting techniques in probabilistic reasoning. For the sake of pedagogy, we begin to focus on the learning strategy by presenting a *direct* L2R algorithm that uses an explicit encoding of probability distributions over the Herbrand space of the input vocabulary. Based on weighted model counting techniques, we shall then derive an *indirect* L2R algorithm that simulates the direct method using an implicit, yet economic, representation of probability measures.

4.1 The Direct Algorithm

The direct L2R algorithm is presented in Figure 1. The key idea is to maintain a probability distribution $\hat{\mathcal{P}} = (\hat{p}_1, \cdots, \hat{p}_N)$ that approximates the target probability measure $\mathcal{P} = (p_1, \cdots, p_N)$. Initially, the hypothesis is set to the uniform distribution over the Herbrand space. On each trial t, the agent receives a query Q_t supplied by its learning interface, and makes a prediction $\hat{y}_t = \mathbf{Pr}_{\hat{\mathcal{P}}_t}(Q_t)$ with its hypothesis $\hat{\mathcal{P}}_t$. In case of mistake, the agent receives the correct value $y_t = \mathbf{Pr}_{\mathcal{P}}(Q_t)$ of the query Q_t. In light of this information, the agent adjusts the probabilities in $\hat{\mathcal{P}}_t$ according to the standard multiplicative weight update rule advocated in online regression algorithms (Kivinen & Warmuth, 1997). As usual, a normalization is also employed to guarantee that the resulting hypothesis $\hat{\mathcal{P}}$ belongs to the probability simplex.

The *entropy* of a target probability distribution $\mathcal{P}$ is defined in the usual way by $H(\mathcal{P}) = -\sum_{i=1}^{N} p_i \log_2 p_i$. Based on this notion, the behavior of the algorithm is captured by the following mistake-bound result.

Theorem 1 *For any relational probabilistic reasoning problem $(\mathcal{P}, \mathcal{Q})$ defined over a vocabulary of dimension d, on input $\gamma > 0$, when $\eta = 4$, the direct EG-L2R algorithm has the following mistake bound*

$$\mathrm{M}_{\gamma}(\mathcal{P}, \mathcal{Q}) \leq \frac{\ln 2}{2\gamma} \big(d - H(\mathcal{P})\big)$$

In other words, the worst-case total number of mistakes of the direct EG L2R algorithm is *logarithmic* in the size of the Herbrand space of the input vocabulary, and hence, *linear* in the input dimension. Interestingly, as we know that $H(\mathcal{P})$ is always nonnegative, the reasoning performance of the algorithm improves with the entropy of its environment. Indeed, the number of mistakes tends to zero as $\mathcal{P}$ is getting closer to the uniform distribution on the Herbrand space $\wp(\mathcal{H})$.

Fig. 1 The direct EG-L2R algorithm

Input:

learning rate $\eta > 0$

Initialization:

set $\hat{p}_{i,1} = \frac{1}{N}$ for $1 \le i \le N$

Trials: in each trial $t \ge 1$

receive a query Q_t

predict $\hat{y}_t = \mathbf{Pr}_{\hat{\mathcal{P}}_t}(Q_t)$

if the prediction is *correct* then

$\hat{\mathcal{P}}_{t+1} = \hat{\mathcal{P}}_t$

else

receive $y_t = \mathbf{Pr}_{\mathcal{P}}(Q_t)$

set $\hat{p}_{i,t+1} = \dfrac{\hat{p}_{i,t}\, e^{\eta \phi_i(Q_t)(y_t - \hat{y}_t)}}{Z_t}$ for $1 \le i \le N$

4.2 The Indirect Algorithm

In relational probabilistic reasoning, a direct implementation of the distribution $\hat{\mathcal{P}}$ is physically impossible as the agent would need to maintain a weight vector of size $\Omega(2^d)$. For example, in the blood type domain, with only ten individuals the agent would need to store about 10^{90} probability values! To circumvent this physical barrier, the key idea behind the indirect algorithm is to encode $\hat{\mathcal{P}}$ into a weighted knowledge base KB, whose size is polynomial in the number of mistakes made so far. In this representation, the prediction task is thus translated into a weighted model counting problem (Sang et al., 2005; Chavira & Darwiche, 2008).

To this end, we need additional definitions. A *weighted formula* is a pair (F, w) where F is a formula and w is a nonnegative real number. Intuitively, w reflects how strong a constraint it is: the higher the weight, the greater the difference in likelihood between an interpretation that satisfies the formula and one that does not. In this setting, any "unweighted" formula F is treated as an abbreviation of $(\neg F, 0)$; it denotes a hard constraint that restricts the space of possible interpretations. A *weighted knowledge base* KB is a finite set of weighted formulas. The *weight* of KB, denoted $\|KB\|$, is given by[2]

$$\|KB\| = \sum_{i=1}^{N} \prod_{(F,w) \in KB} w^{\phi_i(F)}$$

The basic idea behind this notion is to ascribe a weight to each interpretation $I_i\,(1 \le i \le N)$, which is specified by the product of weights w of all formulas (F, w) satisfied by I_i, i.e. $\phi_i(F) = 1$. Thus, the weight of the knowledge base KB is just the sum of weights of the interpretations that are models of KB.

[2] We take the usual convention that $0^0 = 1$ and $0^x = 0$ for any real number $x > 0$.

Fig. 2 The indirect EG-L2R algorithm

Input:

learning rate $\eta > 0$

Initialization:

set $KB_1 = \{(\top, 1)\}$

Trials: in each trial $t \geq 1$

receive a query Q_t

predict $\hat{y}_t = \mathbf{Pr}_{KB_t}(Q_t)$

if the prediction is *correct* then
  $KB_{t+1} = KB_t$
else
  receive $y_t = \mathbf{Pr}_{\mathcal{P}}(Q_t)$
  set $KB_{t+1} = KB_t \cup \{(Q_t, w_t)\}$ where $w_t = e^{\eta(y_t - \hat{y}_t)}$

Now, consider a formula Q of the query language $\mathcal{Q}$. Then, the *degree of belief* in Q given KB is defined by

$$\mathbf{Pr}_{KB}(Q) = \frac{\|KB \cup \{Q\}\|}{\|KB\|}$$

Recall that in the expanded knowledge base $KB \cup \{Q\}$ the formula Q is treated as an abbreviation of $(\neg Q, 0)$. Thus, $\|KB \cup \{Q\}\|$ denotes the sum of weights of the interpretations satisfying both KB and Q. By normalizing this sum using the weight of KB, the degree of belief in Q given KB can be regarded as the probability of choosing an interpretation I at random that satisfies Q according to the probability distribution induced by the weights of models of KB.

Interestingly, the *weighted-worlds* approach to degrees of belief suggested in this study provides a natural generalization of the *random-worlds* approach advocated for the particular case of unweighted knowledge bases (Grove et al., 1994; Bacchus et al., 1996; Halpern, 2003). Indeed, if KB is reduced to a set of hard constraints, then $\|KB\|$ corresponds to the number of models of KB, and hence, in this case the degree of belief in Q given KB is just the probability of choosing an interpretation at random that satisfies Q out of all the interpretations that satisfy KB.

We are now in position to examine the indirect EG-L2R algorithm. As described in Figure 2, the backbone of the algorithm is formed by maintaining a weighted knowledge base used to predict the likelihood of incoming queries. The reasoner starts with the tautology $(\top, 1)$. On each trial t, the agent assigns a degree of belief $\mathbf{Pr}_{KB_t}(Q_t)$ to the input query Q_t according to its knowledge base KB_t. In case of mistake, KB_t is simply expanded with the weighted formula (Q_t, w_t) that conveys, into a concise form, the knowledge gathered by the reasoner during the interaction with its interface. In particular, the weight w_t is adjusted by exponentiating the gradient of the agent's prediction loss with respect to its knowledge base.

As a representation theorem, the following result claims that the indirect L2R algorithm is an *exact simulation* of the direct L2R algorithm. Namely, on the same sequential prediction game, both algorithms assign the same degrees of belief.

Theorem 2 *Indirect EG-L2R exactly simulates the Direct EG-L2R.*

To summarize, the (indirect) EG-L2R algorithm is characterized by two important features. First, its mistake bound is linear in the dimension of the input vocabulary. This is indeed an immediate corollary of Theorems 1 and 2. Second, the size of the hypothesis KB maintained by the reasoner is also linear in the number of ground atoms. This follows from the fact the algorithm is conservative and hence, expands its knowledge base only if it makes a mistake.

Despite these encouraging properties, it remains to be seen how the algorithm can efficiently *evaluate* degrees of belief. This is the purpose of the next section.

5 Tractable Query Languages

After an excursion into the learning aspects of the framework, we now concentrate on the reasoning aspects of the framework by examining several query languages that are tractable for model counting. The utility of our architecture is thus highlighted by showing that relational probabilistic reasoning problems defined over some restricted yet expressive of conjunctive query languages are "mistake bound learnable". Based on the conservativeness property of the EG-L2R algorithm and the aforementioned conversion technique, these reasoning problems can also be shown "PAC learnable".

The basic building block of tractable languages lies in the notion of *decomposable* conjunctive query. Based on this notion, we begin to examine simple hitting query languages, next we turn to cluster query languages, and then we extend still further the expressiveness of the framework by exploring conjunctive query languages with parameterized cluster-width.

5.1 Decomposable Conjunctive Queries

A fundamental issue behind probabilistic reasoning is to evaluate in a reasonable amount of time the number of all interpretations satisfying a given formula F, that is $\|F\|$, if we regard F as a strong constraint of the form $(\neg F, 0)$. Before examining tractable classes of conjunctive queries, it is instructive to observe how the number of models of F can be evaluated using the ground atoms of F. Let $\mathcal{H}(F)$ denote the set of all ground atoms satisfying the following condition: $A \in \mathcal{H}(F)$ if and only if $A \in \mathcal{H}$ and there is a ground substitution θ such that A occurs in the instance $F\theta$. We remark that $\wp(\mathcal{H}(F))$ is the space of all Herbrand interpretations generated from $\mathcal{H}(F)$. The number of all interpretations in $\wp(\mathcal{H}(F))$ satisfying F is denoted $\langle\!\langle F \rangle\!\rangle$. Now, let d be the dimension of our background vocabulary. Based on the above notations, we can easily verify that $\|F\| = 2^{d-|\mathcal{H}(F)|} \langle\!\langle F \rangle\!\rangle$. This simple observation suggests that if $|\mathcal{H}(F)|$ and $\langle\!\langle F \rangle\!\rangle$ can be evaluated in polynomial time, then $\|F\|$ can be evaluated in polynomial time.

All query languages investigated in this study are conjunctive fragments of the class of *quantified propositions* described in (Cumby & Roth, 2000; Valiant, 2000b). More precisely, queries are defined to be restricted relational expressions in which (1) there is only a single predicate in the scope of each variable, and (2) there is only a single type of quantifier in the front of each predicate.

These restrictions can be formalized in the following way. An *existentially* (resp. *universally*) *quantified atom* is an expression of the form $\exists x_1, \cdots, \exists x_p \mathsf{R}(t_1, \cdots, t_k)$ (resp. $\forall x_1, \cdots, \forall x_p \mathsf{R}(t_1, \cdots, t_k)$), where $0 \leq p \leq k$ and all the p variables occurring among the k terms are within the scope of an existential (resp. universal) quantifier. An *uniformly quantified atom*, or *quantified atom* for short, is an existentially quantified atom or an universally quantified atom. A *quantified literal* is a quantified atom A or its negation $\neg A$. Finally, a *quantified conjunctive query* is a conjunction of quantified literals. For example, $\exists y_1 \, \mathsf{Connected}(\mathsf{l}_1, y_1) \wedge \neg \forall x \, \mathsf{At}(x, \mathsf{l}_2) \wedge \exists y_2 \, \mathsf{At}(\mathsf{b}_1, y_2)$ is a quantified conjunctive query.

Lemma 1 *Let A and B be two quantified atoms defined on a vocabulary of dimension d. Then deciding whether A entails B can be determined in $O(d|A||B|)$ time.*

Proof Since atoms can be either existentially quantified or universally quantified, four cases have to be considered. If both A and B are existentially quantified, then $A \vDash B$ if and only if $\mathcal{H}(A) \subseteq \mathcal{H}(B)$. Dually, if both A and B are universally quantified, then $A \vDash B$ if and only if $\mathcal{H}(B) \subseteq \mathcal{H}(A)$. Now, if A is universally quantified and B existentially quantified, then $A \vDash B$ if and only if we have $\mathcal{H}(A) \cap \mathcal{H}(B) \neq \varnothing$. Finally, if A is existentially quantified and B is universally quantified, then $A \vDash B$ if and only if $|\mathcal{H}(A)| = |\mathcal{H}(B)| = 1$, and $\mathcal{H}(A) = \mathcal{H}(B)$. Notice that the fourth case is *not* the contraposition of the third case. Obviously, the first three cases can be decided in $O(d|A||B|)$ time by listing the ground instances of one atom and checking whether they are instances of the other. The fourth case can be decided in $O(d|A| + d|B|)$ time by testing whether both atoms contain a single ground instance and, in case of success, comparing these instances. □

Based on the above lemma, the interest of quantified conjunctive queries lies in the fact that many logical operations can be realized in polynomial time. Given two formulas F and G, we say that F and G *match* if F entails G or G entails F. Dually, we say that F and G *clash* if $F \wedge G$ is unsatisfiable.

Proposition 2 *Let Q and Q' be two quantified conjunctive queries defined over a vocabulary of dimension d. Then deciding whether Q and Q' clash or match can be determined in $O(d|Q||Q'|)$ time.*

Proof As usual, the *sign* of a quantified literal L is positive if L can be rewritten as a quantified atom, and negative otherwise. Consider two quantified literals L and L', and let A and A' be the quantified atoms occurring in L and L' respectively. Then $L \vDash L'$ if and only if either both L and L' are positive and $A \vDash A'$, or both L and L' are negative and $A' \vDash A$. In both cases, this can be determined in $O(d|L||L'|)$ using Lemma 1. Now, consider two quantified conjunctive queries Q and Q'. Since literals are independently quantified, $Q \vDash Q'$ if and only if for each literal L' of Q' there is a literal L of Q such that $L \vDash L'$. Furthermore, $Q \wedge Q'$ is unsatisfiable if and only if there is a literal L of Q and a literal L' of Q' such that $L \wedge L'$ is unsatisfiable, which is equivalent to state that $L \vDash \neg L'$. So, deciding whether Q and Q' either match or clash can be determined in $O(d|Q||Q'|)$ time by comparing pairs of literals. □

Example 3 Consider an instance of the random block domain with the following queries $\exists y \, \mathsf{Connected}(\mathsf{l}_1, y) \wedge \exists x \, \mathsf{At}(x, \mathsf{l}_2)$ and $\exists y_1 \, \mathsf{Connected}(\mathsf{l}_2, y_1) \wedge \neg \exists x \exists y_2 \, \mathsf{At}(x, y_2)$. Clearly, these queries are clashing because $\exists x \exists y_2 \, \mathsf{At}(x, y_2)$ is entailed by $\exists x \, \mathsf{At}(x, \mathsf{l}_2)$.

To obtain tractable forms of model counting, we need to introduce an additional restriction on queries which states that any ground atom is covered by at most one quantified literal. The restriction can be formalized as follows. Two formulas F and G are said to *overlap* if $\mathcal{H}(F) \cap \mathcal{H}(G) \neq \varnothing$. Based on this notion, a *decomposable conjunctive query* is a conjunction of quantified literals which do not overlap each other. For example, the query $\exists y_1\, \mathsf{Connected}(\mathsf{l}_1, y_1) \wedge \neg\forall x\, \mathsf{At}(x, \mathsf{l}_2) \wedge \exists y_2\, \mathsf{At}(\mathsf{b}_1, y_2)$ is not decomposable since the last two literals overlap. On the other hand, the following expression $\exists y\, \mathsf{Connected}(\mathsf{l}_1, y) \wedge \exists x_1\, \mathsf{At}(x_1, \mathsf{l}_2) \wedge \neg\exists x_2\, \mathsf{At}(x_2, \mathsf{l}_3)$ is decomposable.

Lemma 2 *For any quantified literal L, the problem of counting the number of models of L can be determined in $O(|L|)$ time.*

Proof We know that $\|L\| = 2^{d-|\mathcal{H}(L)|}\langle\!\langle L \rangle\!\rangle$. So, we only need to show that both $|\mathcal{H}(L)|$ and $\langle\!\langle L \rangle\!\rangle$ can be evaluated in linear time. Let $\mathsf{R}(t_1, \cdots, t_k)$ denote the atomic expression occurring in L. Without loss of generality, we assume that for some integer p such that $0 \leq p \leq k$, all the first p terms $t_1, \cdots, t_p$ are variables. For $1 \leq i \leq p$ let n_i be the number of constant symbols of sort s_i. Obviously, $|\mathcal{H}(L)| = \prod_{i=1}^{p} n_i$. Therefore, $|\mathcal{H}(L)|$ can be evaluated in $O(|L|)$ time.

Now, let us turn to $\langle\!\langle L \rangle\!\rangle$. Since any literal is either positive or negative, and any atom is either universally quantified or existentially quantified, four cases have to be considered. In the following, l is an abbreviation of $|\mathcal{H}(L)|$. First, suppose that the quantified atom of L is $\forall x_1, \cdots, \forall x_p \mathsf{R}(t_1, \cdots, t_k)$. If L is positive, then $\langle\!\langle L \rangle\!\rangle = 1$ because the unique interpretation I satisfying L is $I = \mathcal{H}(L)$. If L is negative, then clearly $\langle\!\langle L \rangle\!\rangle = 2^l - 1$. Dually, suppose that the quantified atom of L is $\exists x_1, \cdots, \exists x_p \mathsf{R}(t_1, \cdots, t_k)$. If L is negative, then it can be rewritten as $\forall x_1, \cdots, \forall x_p \neg\mathsf{R}(t_1, \cdots, t_k)$, and hence $\langle\!\langle L \rangle\!\rangle = 1$, because the unique interpretation I satisfying L is $I = \varnothing$. Finally, if L is positive, then it follows that $\langle\!\langle L \rangle\!\rangle = 2^l - 1$. Note that in any case, if $\mathsf{R}(t_1, \cdots, t_k)$ is a ground atom (i.e. $p = 0$), then $\langle\!\langle L \rangle\!\rangle = 1$. Since in all the four cases $\langle\!\langle L \rangle\!\rangle$ can be evaluated in $O(|L|)$ time, this completes the proof. □

A *decomposable conjunctive query language* is a countable set $\mathcal{Q}$ of decomposable conjunctive queries. The key interest behind the notion of decomposability is clarified by the following property.

Proposition 3 *For any decomposable conjunctive query Q, the problem of counting the number of models of Q can be determined in $O(|Q|)$ time.*

Proof Let Q be a decomposable conjunctive query. Since $\|Q\| = 2^{d-|\mathcal{H}(Q)|}\langle\!\langle Q \rangle\!\rangle$ we only need to prove that both $|\mathcal{H}(Q)|$ and $\langle\!\langle Q \rangle\!\rangle$ can be evaluated in linear time. Suppose that Q contains q quantified literals. The decomposability property implies that $|\mathcal{H}(Q)| = \sum_{i=1}^{q} |\mathcal{H}(L_i)|$ and $\langle\!\langle Q \rangle\!\rangle = \prod_{i=1}^{q} \langle\!\langle L_i \rangle\!\rangle$, because literals do not overlap each other. Therefore, by application of Lemma 2, the result follows. □

Example 4 Consider again a simple instance of the random block domain involving 2 blocks and 5 locations. The dimension of the vocabulary is thus $(3 \times 25) + 10 = 85$. Initially, the knowledge base of the agent is $(\top, 1)$, so $\|KB\| = 2^{85}$. Now suppose that our agent receives the query $\exists y\, \mathsf{Connected}(\mathsf{l}_1, y) \wedge \neg\exists x\, \mathsf{At}(x, \mathsf{l}_2)$. We thus have

$$\mathbf{Pr}_{KB}(Q) = \frac{\|KB \wedge \exists y \mathsf{Connected}(\mathsf{l}_1, y) \wedge \neg\exists x \mathsf{At}(x, \mathsf{l}_2)\|}{\|KB\|} = \frac{2^{85-(5+2)} \times (2^5 - 1) \times 1}{2^{85}} = \frac{31}{128}$$

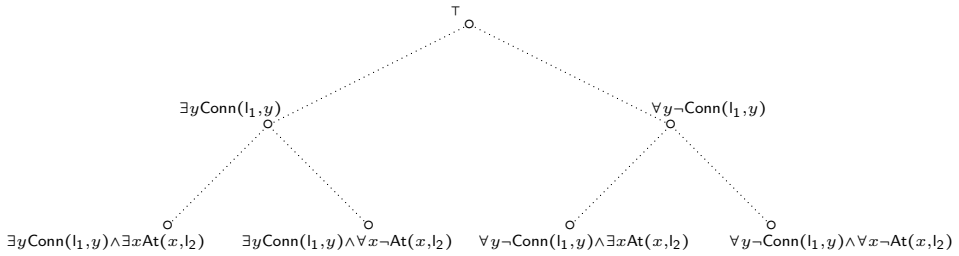

Fig. 3 A hitting query language for the random blocks domain.

We conclude this section with some useful definitions concerning knowledge bases. Given a query language $\mathcal{Q}$ and a weighted knowledge base KB, we say that KB is *generated* from $\mathcal{Q}$ if $F \in \mathcal{Q}$ for any weighted formula (F, w) occurring in KB. Furthermore, KB is said to be *rooted* if it contains the tautology $(\top, 1)$. Finally, KB called *irredundant* if it does not include a pair of distinct formulas (F_1, w_1) and (F_2, w_2) such that F_1 is equivalent to F_2. When considering the behavior of the EG-L2R algorithm, we remark that the knowledge base maintained by the agent is rooted and generated by the query language of its interface. Clearly, this knowledge base can be kept irredundant by simply replacing any pair (F_1, w_1) and (F_2, w_2) such that $F_1 \equiv F_2$, with the formula $(F_1, w_1 w_2)$. If $\mathcal{Q}$ is a decomposable query language, this operation can be realized in polynomial time.

5.2 Hitting Query Languages

In propositional logic, an important class of formulas for which model counting can be determined in polynomial time is the class of hitting formulas (Büning & Zhao, 2001). Based on the notion of decomposable queries, we extend this class to the relational setting and show that relational probabilistic reasoning problems defined over hitting query languages are mistake-bound learnable. In the following, a *hitting set* is a set S of formulas where any two of its formulas either match or clash.

Definition 5 A *hitting query language* is a hitting set of decomposable conjunctive queries.

Example 5 Any hitting query language can be represented by a tree, where each vertex is labeled by an equivalence class of formulas. Two queries Q and Q' that match are linked by a path in the tree. Conversely, two queries Q and Q' that are not joined by any path in the tree must clash. For example, consider in Figure 3 a labeled tree for the random blocks domain. Then, the set of vertices in the tree forms a hitting query language. Note that if this language would be expanded with a query such as $\exists x \mathsf{At}(x, \mathsf{l}_2)$, then the hitting property would be violated because $\exists x \mathsf{At}(x, \mathsf{l}_2)$ does not match or clash with the formulas $\exists y \mathsf{Conn}(\mathsf{l}_1, y)$ and $\neg \exists y \mathsf{Conn}(\mathsf{l}_1, y)$.

The salient feature of hitting query languages lies in the fact that any weighted knowledge base generated from a hitting language can be represented into an tree-like data structure for which query evaluation is essentially linear in the size of the knowledge base and the dimension of the input vocabulary.

To capture this property within a formal setting, we need to introduce additional definitions. Consider a hitting set S and a formula F in S. Borrowing the terminology of trees, an *ancestor* of F is a distinct formula G of S such that $F \models G$. A *parent* of F is an immediate ancestor of F, that is, an ancestor G of F such that $F \models G$ and $G' \not\models G$ for any distinguished ancestor G' of F. From a dual viewpoint, a *descendant* of F is a distinct formula G in S such that $G \models F$, and a *child* of F is an immediate descendant of F, that is, a descendant G of F such that and $G \not\models G'$ for any distinguished descendant G' of F. Finally, a formula F is called a *root* if it does not have any parent, and a *leaf* if it does not have any child.

For example, using the hitting set in Figure 3, we observe that $\top$ and $\exists y \mathsf{Conn}(\mathsf{l}_1, y)$ are ancestors of $\exists y \mathsf{Conn}(\mathsf{l}_1, y) \wedge \exists x \mathsf{At}(x, \mathsf{l}_2)$ and $\top$ is the parent of $\exists y \mathsf{Conn}(\mathsf{l}_1, y)$.

Now, let KB be a rooted, irredundant knowledge base generated from a hitting query language $\mathcal{Q}$. Then the *tree* of KB, denoted $\mathbf{Tr}(KB)$, is a labeled rooted tree defined in the following way. First, to each formula (F_i, w_i) in KB we associate a vertex i in $\mathbf{Tr}(KB)$. Second, to each pair of formulas (F_i, w_i) and (F_j, w_j) in KB such that F_i is a child of F_j in the set of formulas of KB, we associate an edge (i, j). By $a(i)$ (resp. $d(i)$) we denote the collection of all vertices j such that F_j is an ancestor (resp. a descendant) of F_i in KB. Third and finally, to each vertex i, we associate a label $(F_i, \widehat{w}_i, \widehat{c}_i)$ where $\widehat{w}_i$ and $\widehat{c}_i$ are defined as follows

$$\widehat{w}_i = w_i \cdot \prod_{j \in a(i)} w_j \quad \text{and} \quad \widehat{c}_i = \begin{cases} \|F_i\| \text{ if } i \text{ is a leaf, and} \\ \|F_i\| - \sum_{j \in d(i)} \widehat{c}_j \text{ otherwise} \end{cases}$$

The key intuition behind this data structure lies in the fact that $\mathbf{Tr}(KB)$ induces a natural partition of the Herbrand space: any interpretation is associated with the most specific formula in KB that covers it. From this viewpoint, $\widehat{c}_i$ is a counter that captures the number of interpretations that satisfy F_i but none of its implicants in the knowledge base. Analogously, $\widehat{w}_i$ is a real value that captures the weight of these interpretations. We remark that $\mathbf{Tr}(KB)$ is indeed a rooted tree since KB includes the tautology $(\top, 1)$. For convenience, the root node is simply denoted $\top$.

With this data structure in hand, the degree of belief of a query Q can be evaluated by expanding $\mathbf{Tr}(KB)$ with a new vertex associated to Q. The expansion procedure presented in Figure 4 takes as input the tree $\mathbf{Tr}(KB)$ of a knowledge base KB and a query Q, and returns as output the new tree $\mathbf{Tr}(KB \cup \{Q\})$. The likelihood of Q is thus obtained by exploring all descendants of Q in the tree.

Lemma 3 *Let $\mathcal{Q}$ be a hitting query language, KB be a rooted irredundant knowledge base generated from $\mathcal{Q}$, and Q a query in $\mathcal{Q}$. Let $\mathbf{Tr}(KB \cup \{Q\})$ be the structure obtained from $\mathbf{Tr}(KB)$ and Q by the tree expansion procedure. Then*

$$\mathbf{Pr}_{KB}(Q) = \frac{\widehat{w}_q \widehat{c}_q + \sum_{i \in d(q)} \widehat{w}_i \widehat{c}_i}{\widehat{w}_\top \widehat{c}_\top + \sum_{j \in d(\top)} \widehat{w}_j \widehat{c}_j}$$

Proof Consider an arbitrary vertex i in $\mathbf{Tr}(KB \cup \{Q\})$ and let $\widehat{F}_i$ denote the formula $F_i \wedge \bigwedge_{j \in d(i)} \neg F_j$. Let $S_i = \{\widehat{F}_j : j \in d(i)\} \cup \{\widehat{F}_i\}$. The key idea of the proof is to show that S_i induces a partition of $\mathcal{M}(F_i)$. To establish this assertion, consider an arbitrary model I of F_i. We first prove that I is model of *at most* one formula in S_i. Suppose this is not the case. Then, there are at least two formulas $\widehat{G}$ and $\widehat{G}'$ in S_i such that I is a model of $\widehat{G} \wedge \widehat{G}'$. Obviously, G and G' cannot clash. So, G and G' must match, which implies that either G is a descendant of G' or the converse. In both cases, $\widehat{G} \wedge \widehat{G}'$ is unsatisfiable and hence, this leads to a contradiction.

Fig. 4 The tree expansion procedure

Input:

The tree $\mathbf{Tr}(KB)$ of a weighted knowledge base KB and a query Q

Begin

find the parent p of Q in $\mathbf{Tr}(KB)$

if $F_p \equiv Q$ then return $\mathbf{Tr}(KB)$

add a new a vertex q and the edge (q, p)

for each child i of p such that $F_i \models Q$, remove the edge (i, p) and add (i, q)

label q with $(Q, \widehat{w}_q, \widehat{c}_q)$ where $\widehat{w}_q = \widehat{w}_p$ and $\widehat{c}_q = \|Q\| - \sum_{i \in d(q)} \widehat{c}_i$

set $\widehat{c}_p$ to $\widehat{c}_p - \widehat{c}_q$

return $\mathbf{Tr}(KB)$

End

Now, we prove that I is model of *at least* one formula of S_i. To this end, let j be the deepest node in the tree for which I is a model of G_j. This node is unique because otherwise I should be a model of two formulas G_j and $G_{j'}$ that would clash. Now, suppose that I is not a model of $\widehat{G}_j$. Then, by construction of $\widehat{G}_j$, there is a descendant k of j in the tree such that I is not a model of $\neg G_k$. So, I is a model of G_k but this contradicts the assumption that j is the deepest node in the tree.

To summarize, for any model I of F_i, there is exactly one $\widehat{G}$ in S_i such that I is a model of $\widehat{G}$. Based on this partition, it follows that $\|F_i\| = \|\widehat{F}_i\| + \sum_{j \in d(i)} \|\widehat{F}_j\|$, and hence $\|\widehat{F}_i\| = \widehat{c}_i$. Furthermore, for any formula F, we know that $\|KB \cup \{F\}\|$ is the sum of weights of models of KB that are also models of F. It follows that

$$\|KB \cup \{F_i\}\| = \|KB \cup \{\widehat{F}_i\}\| + \sum_{j \in d(i)} \|KB \cup \{\widehat{F}_j\}\|$$

By construction, the weight of any model of $\widehat{F}_i$ is the product of weights of each formula (F_k, w_k) in KB such that F_k is entailed by F_i. Thus by definition of $\widehat{w}_i$, it follows that $\|KB \cup \widehat{F}_i\| = \widehat{w}_i \widehat{c}_i$. Therefore,

$$\|KB \cup \{F_i\}\| = \widehat{w}_i \widehat{c}_i + \sum_{j \in d(i)} \widehat{w}_j \widehat{c}_j$$

Finally, since $\mathbf{Pr}_{KB}(Q)$ is given by $\dfrac{\|KB \cup \{Q\}\|}{\|KB \cup \{\top\}\|}$, the result follows. □

Based on the above lemma and the complexity analysis of the tree expansion procedure, we can derive the following property.

Theorem 3 *Let $(\mathcal{P}, \mathcal{Q})$ be a relational probabilistic reasoning problem where $\mathcal{Q}$ is a hitting query language over a vocabulary of dimension d, m be the total number of mistakes made by the EG-L2R algorithm, and n be the largest size of any formula in $\mathcal{Q}$. Then for any query Q in $\mathcal{Q}$, $\mathbf{Pr}_{KB}(Q)$ can be evaluated in $O(dmn^2)$ time.*

Proof The result can be established by assuming that the agent maintains the tree of its knowledge base, during each trial t. Note that if $t = 1$ then $\mathbf{Tr}(KB_t)$ consists in a single node labeled by $(\top, 1, 2^d)$. Now, consider that $t \geq 1$ and let Q_t be an incoming query. The agent starts to transform $\mathbf{Tr}(KB_t)$ into $\mathbf{Tr}(KB_t \cup \{Q_t\})$ according to the tree expansion procedure, which takes $O(dmn^2)$ time. Indeed, searching the parent of Q takes dmn^2 time in the worst case by comparing Q with all formulas in the tree, using the comparison method suggested in Proposition 2. Updating edges simply takes $O(m)$ time, and updating vertices takes $O(|Q| + m)$ time. Notably, to determine $\widehat{c}_q$, we simply need to evaluate the number of the models of Q according to Proposition 3, and subtract from it each counter in the descendants of Q. Based on Lemma 3, the agent can ascribe a degree of belief in $O(m)$ time by exploring the descendants of Q and $\top$ in its tree. If the prediction is correct, then the agent can reset the original tree in $O(m)$ time by updating at most m edges. Otherwise, the agent simply needs to propagate the update value $e^{\eta(y_t - \hat{y}_t)}$ to Q and its ascendants in order to derive the new tree $\mathbf{Tr}(KB_{t+1})$. Again, this takes $O(m)$ time. Based on these operations, the overall cost per trial requires $O(dmn^2)$ time. □

We have previously shown that the cumulative number of mistakes of the EG-L2R algorithm is linear in the input dimension. Since the size of the weighted knowledge base is linear in the number of mistakes, we obtain the following result.

Corollary 1 *There exists an efficient mistake bound L2R algorithm for any relational probabilistic reasoning problem $(\mathcal{P}, \mathcal{Q})$ where $\mathcal{Q}$ is a hitting query language.*

5.3 Cluster Query Languages

The expressiveness of query languages can be increased by forming clusters of hitting languages. In propositional logic, a cluster formula is a union of non-overlapping hitting formulas (Nishimura et al., 2006). This notion is ported to relational query languages in the following way. A set S of formulas is *connected* if for any pair of formulas $\{F, G\}$ in S, there is a sequence of formulas $F_1, \cdots, F_n$ such that $F = F_1$, $G = F_n$ and F_i and F_{i+1} overlap for all i such that $1 \leq i \leq n-1$. A *cluster set* is a set S of formulas where any connected subset of S is a hitting set.

Definition 6 A *cluster query language* is a cluster set of decomposable conjunctive queries.

Example 6 By analogy with hitting languages, any cluster query language can be represented by a tree where each vertex is labeled by an equivalence class of formulas. Two queries that match are joined by a path in the tree. Dually, two queries that are not joined by a path in the tree are necessarily clashing or non-overlapping. For example, consider in Figure 5 a labeled tree for the random blocks domain. Then the set of vertices in the tree forms a cluster query language.

Theorem 4 *Let $(\mathcal{P}, \mathcal{Q})$ be a relational probabilistic reasoning problem where $\mathcal{Q}$ is a cluster query language over a vocabulary of dimension d, m be the total number of mistakes made by the EG-L2R algorithm, and n be the largest size of any formula in $\mathcal{Q}$. Then for any query Q in $\mathcal{Q}$, $\mathbf{Pr}_{KB}(Q)$ can be evaluated in $O(dmn^2)$ time.*

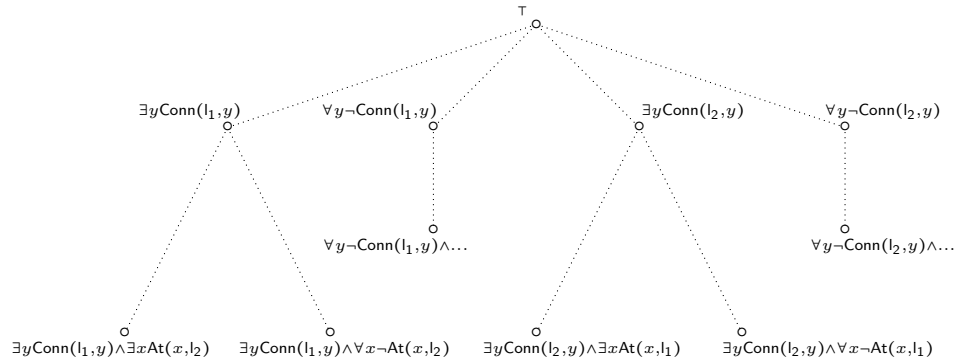

Fig. 5 A cluster query language for the random blocks domain.

Proof Any cluster query language $\mathcal{Q}$ is formed by the union $\mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_d$ of up to d non-overlapping hitting languages. Thus, any rooted irredundant knowledge base KB_t generated from $\mathcal{Q}$ can be represented by a forest $\{\mathbf{Tr}(KB_{t,1}), \cdots, \mathbf{Tr}(KB_{t,d})\}$, where $KB_{t,i}$ is generated from $\mathcal{Q}_i$. To identify clusters, each tree $\mathbf{Tr}(KB_{t,i})$ can be associated with the d-dimensional characteristic boolean vector $\mathbf{h}_{t,i}$ of $\mathcal{H}(KB_{t,i})$. Now consider an incoming query Q_t and suppose that $Q_t \in \mathcal{Q}_i$. The task of finding $KB_{t,i}$ takes $O(dmn)$ time by testing whether $h_{t,i}(j) = 1$ for at least one A_j in $\mathcal{H}(Q_t)$. Furthermore, since clusters are non-overlapping, $\mathbf{Pr}_{KB_t}(Q_t) = \frac{\|KB_{t,i} \cup \{Q_t\}\|}{\|KB_{t,i}\|}$. By application of Theorem 3, this takes $O(dmn^2)$ time. Additionally, the cost of updating $\mathbf{Tr}(KB_{t,i})$ and $\mathbf{h}_{t,i}$ only takes $O(dm + mn)$ time. □

Corollary 2 *There exists an efficient mistake-bound L2R algorithm for any relational probabilistic reasoning problem* $(\mathcal{P}, \mathcal{Q})$ *where* $\mathcal{Q}$ *is a cluster query language.*

5.4 Parameterized Cluster-Width

As a glimpse beyond cluster languages, we can relax the cluster assumption and explore the larger family of conjunctive query languages with bounded *cluster-width*. Very roughly, the cluster-width of a propositional formula is an upper bound on the minimum number of boolean variables that must be assigned in order to transform the initial formula into a cluster expression, for which model counting can be decided in polynomial time (Nishimura et al., 2006).

The notion of cluster-width is ported to relational query languages using the following definitions. Given a set S of formulas, a *hitting obstruction* is a pair of formulas $\{F, G\}$ in S such that F and G are connected, but they do not match or clash. Note that S is a cluster set if and only if it does not contain any hitting obstruction. With a hitting obstruction we associate the following pair of sets of ground atoms: $\{\mathcal{H}(F), \mathcal{H}(G)\}$. Based on this notion, let $\mathbf{Gr}(S)$ denote the graph with vertex set $\mathcal{H}$; two ground atoms A and B are joined by an edge in the graph $\mathbf{Gr}(S)$ if and only if there is an hitting obstruction $\{F, G\}$ in S where $A \in \mathcal{H}(F)$ and $B \in \mathcal{H}(G)$. We call $\mathbf{Gr}(S)$ the *obstruction graph* of S.

Recall that a *vertex cover* of a graph is a set of vertices that contains at least one end of every edge of the graph. The *cluster-width* of a set S of formulas is defined by the size of a smallest vertex cover in the obstruction graph $\mathbf{Gr}(S)$ of S.

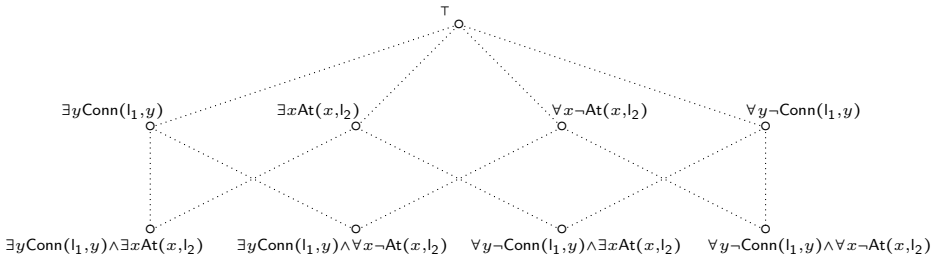

Fig. 6 A k-cluster query language for the random blocks domain with $k = 2$

Definition 7 A *k-cluster-width* query language is a decomposable conjunctive query language with cluster width at most k.

Example 7 Consider again a simple instance of the random block domain with 2 blocks and 5 locations. Suppose that we would like to employ the query language illustrated in Figure 6. As usual, two queries that match are joined by a path in the graph. We can observe that there are actually two hitting obstructions: the first obstruction is given by the pair $\{\exists y \mathsf{Conn}(\mathsf{l}_1, y), \exists x \mathsf{At}(x, \mathsf{l}_2)\}$, and the second obstruction is the dual pair $\{\forall y \neg \mathsf{Conn}(\mathsf{l}_1, y), \forall x \neg \mathsf{At}(x, \mathsf{l}_2)\}$. The resulting obstruction graph of this language is represented in Figure 7. We can easily verify that the size of a minimal vertex cover for this graph is 2.

Example 8 Now consider an instance of the blood type domain involving four persons: Ann, Bob, Chris and Mary. We would like to estimate the blood type of these persons from parenthood or genetic observations. The query language is represented in Figure 8. Again, two queries that match are joined by a path in the graph. We can distinguish two groups of hitting obstructions. The first group includes queries relative to Ann and Mary, while the second includes queries relative to Bob and Chris. The obstruction graph associated to the language is illustrated in Figure 9. We can easily observe that the size of a minimal vertex cover for this graph is 6.

The key interest of query languages with fixed cluster-width lies in the so-called notion of *backdoor set*. In propositional logic, a (strong) backdoor of a satisfiability problem or a model counting problem is a set of boolean variables which, however they are assigned, provide a simplified formula that can be solved in polynomial time (Williams et al., 2003; Ruan et al., 2004). This notion can be extended to our framework in the following way.

Consider a subset $\mathcal{B}$ of the set $\mathcal{H}$ of all ground atoms defined over the background vocabulary. A *partial Herbrand interpretation* over $\mathcal{B}$ is a map I that assigns to each atom in $\mathcal{B}$ a truth value and leaves undefined the atoms in $\mathcal{H} - \mathcal{B}$. Given a decomposable conjunctive formula F, the *simplification* of F by I, denoted $F[I]$, is defined as follows: for any literal L in F that is true in I, L is replaced with $\top$, and for any literal L in F that is false in I, F is replaced with $\bot$. More generally, given a knowledge base KB generated from a decomposable conjunctive query language, the simplification of KB by I, denoted $KB[I]$, is defined as follows. First, each weighted formula (F, w) in KB is replaced with $(F[I], w)$, next each resulting formula of the form $(\bot, w)$ is removed from $KB[I]$, and then the set of all formulas of the form $(\top, w_i)$ is replaced with a single root $(\top, \prod_i w_i)$ in $KB[I]$.

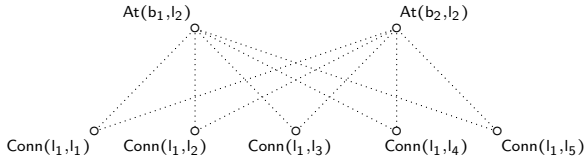

Fig. 7 The obstruction graph associated with random blocks domain in Fig. 6.

A set of ground atoms $\mathcal{B}$ is called a *cluster-backdoor set* for a knowledge base KB if for any partial Herbrand interpretation I over $\mathcal{B}$, the simplification $KB[I]$ of KB by I is a cluster set of weighted formulas. The following states that the obstruction graph of our knowledge base can be constructed in polynomial time.

Lemma 4 *Let $\mathcal{Q}$ be a decomposable query language, where n is the largest size of any formula in $\mathcal{Q}$. Let KB be a knowledge base containing m formulas generated from $\mathcal{Q}$ and $\mathbf{Gr}(KB)$ the obstruction graph of KB. Then for any $Q \in \mathcal{Q}$, the obstruction graph $\mathbf{Gr}(KB \cup \{Q\})$ can be constructed in $O(d^2m + dmn^2)$ time.*

Proof Without loss of generality, we assume that KB is decomposed into a set $\{KB_1, \cdots, KB_d\}$ of up to d clusters, where each cluster is associated with the characteristic vector of its set of ground atoms. Note that any cluster is not necessarily a hitting set. Given a query $Q \in \mathcal{Q}$, we begin to check whether Q is connected to some cluster KB_i. As shown in the proof of Theorem 4, this can be done in $O(dmn)$ time using the characteristic vector of each cluster. If Q is not connected to any cluster, then we return $\mathbf{Gr}(KB)$. Otherwise, for each formula F_i occurring in KB_i, we first test whether F_i and Q clash or match, and if they are incomparable, we expand the obstruction graph with the list of edges (A_i, B) where $A_i \in \mathcal{H}(F_i)$ and $B \in \mathcal{H}(Q)$. By proposition 2, the first operation takes $O(dn^2)$ time. The second operation takes $O(dn + d^2)$ time by generating and connecting the ground atoms of F_i and Q. The result follows by repeating the process to the m formulas in KB_i. □

It is well-known that the vertex cover problem is fixed-parameter tractable. In other words, given a fixed parameter k, we can determine whether a graph containing n vertices has a vertex cover of size at most k in time bounded by $2^k n^{O(1)}$. By a direct application of the current best worst-case time complexity obtained by Chen et al. (2005) for the vertex cover problem, we can derive the following property.

Lemma 5 *Let KB be a knowledge generated from a decomposable conjunctive query language over a vocabulary of dimension d, and $\mathbf{Gr}(KB)$ its obstruction graph. Then one can find in time $O(1.273^k + dk)$ a vertex cover of $\mathbf{Gr}(KB)$ of size at most k, or determine that no such vertex cover exists.*

With these notions in hand, the connection between vertex covers and cluster-backdoor sets is captured by the following property.

Lemma 6 *Let KB be a knowledge generated from a decomposable conjunctive query language, and $\mathbf{Gr}(KB)$ its obstruction graph. Then any vertex cover of $\mathbf{Gr}(KB)$ is a cluster-backdoor set of KB.*

Proof Consider a vertex cover $\mathcal{B}$ of the obstruction graph $\mathbf{Gr}(KB)$. For each hitting obstruction $\{F, F'\}$ in the set of formulas of KB, there is an associated bipartite subgraph $\mathcal{G}$ of $\mathbf{Gr}(KB)$ that joins each $A \in \mathcal{H}(F)$ to each $A' \in \mathcal{H}(F')$. Now, consider the set $\mathcal{B}_\mathcal{G}$ of vertices in $\mathcal{G}$ that belong to $\mathcal{B}$. Obviously, $\mathcal{B}_\mathcal{G}$ is a vertex cover of $\mathcal{G}$.

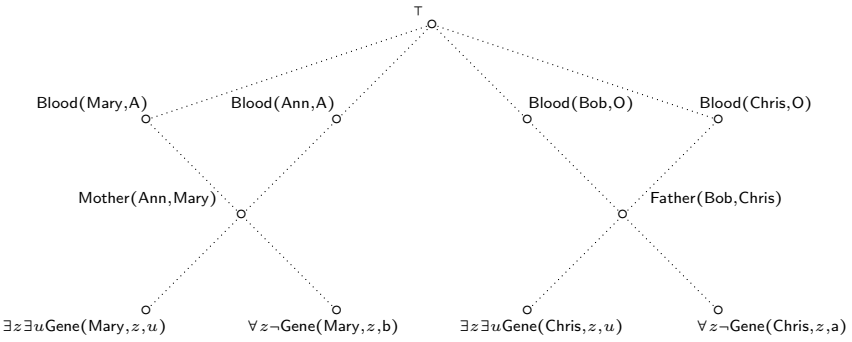

Fig. 8 A k cluster-width query language for the blood type domain. In this compact representation, each formula F associated to a vertex s is given by the conjunction of the labels occurring in any minimal path from $\top$ to s.

Let I be a partial interpretation defined over $\mathcal{B}_\mathcal{G}$. We shall prove that *at least one* of the simplified formulas $F[I]$ and $F'[I]$ is either $\top$ or $\bot$.

First, suppose that F and F' are two quantified literals denoted L and L'. By construction, any quantified literal can be rewritten into a finite disjunction of ground literals or a finite conjunction of ground literals. If both $L[I]$ and $L'[I]$ are left undefined, then in any case, there are at least two ground literals $L_i \in L$ and $L'_j \in L'$ such that the corresponding ground atoms A_i and A'_j are left undefined by I. It follows that $\mathcal{B}_\mathcal{G} \cap \{A_i, A'_j\} = \varnothing$, and hence, there is no vertex in $\mathcal{B}_\mathcal{G}$ that covers the edge (A_i, A'_j) in $\mathcal{G}$. This, however, contradicts the fact that $\mathcal{B}_\mathcal{G}$ is a vertex cover of $\mathcal{G}$. Now, consider the general case where F and F' are conjunctions of quantified literals. Again, if $F[I]$ and $F'[I]$ are left undefined, then there are at least two quantified literals $L \in F$ and $L' \in F'$ such that both $L[I]$ and $L'[I]$ are left undefined. However, as shown above, this contradicts the fact that $\mathcal{B}_\mathcal{G}$ is a vertex cover of $\mathcal{G}$. Therefore, at least one of $F[I]$ and $F'[I]$ is simplified to $\top$ or $\bot$.

Finally, since $\mathcal{B}$ is a vertex cover of each bipartite subgraph $\mathcal{G}$ of $\mathbf{Gr}(KB)$ associated to a hitting obstruction in KB, it follows that any partial interpretation I over $\mathcal{B}$ will remove each obstruction by deleting at least one formula in the obstruction. Consequently, $\mathcal{B}$ is indeed a cluster-backdoor set of KB. □

Theorem 5 *Let $(\mathcal{P}, \mathcal{Q})$ be a relational probabilistic reasoning problem where $\mathcal{Q}$ is a k-cluster-width language over a vocabulary of dimension d, m be the total number of mistakes made by EG-L2R, and n be the largest size of any query in $\mathcal{Q}$. Then for any query Q, $\mathbf{Pr}_{KB}(Q)$ can be evaluated in $O\big(1.273^k + dk + 2^k(d^2m + dm^2n^2)\big)$ time.*

Proof This can be established by assuming that the agent maintains the conflict graph $\mathbf{Gr}(KB_t)$ of its knowledge base during each trial. Consider an arbitrary trial $t \geq 1$ and let Q_t be the supplied query. By Lemma 4, $\mathbf{Gr}(KB \cup \{Q_t\})$ can be constructed in $O(d^2m + dmn^2)$ time. Furthermore, by Lemma 5, a vertex cover $\mathcal{B}$ of size at most k for this graph can be found in $O(1.273^k + dk)$ time. Finally, by Lemma 6, we know that $\mathcal{B}$ is a cluster-backdoor set of $KB \cup \{Q_t\}$. Now, let I be a partial interpretation over $\mathcal{B}$. Then the set of formulas in $KB[I]$ is a cluster set. Each cluster in $KB[I]$ can be transformed into a tree structure by repeated application of the tree expansion procedure. To this point, we only need to compile the cluster that includes Q_t. Since there are at most m formulas in that cluster, the tree compilation requires $O(dm^2n^2)$ time, and query evaluation takes $O(dmn)$ time. Finally, since there are 2^k distinct partial interpretations generated from $\mathcal{B}$, the result follows. □

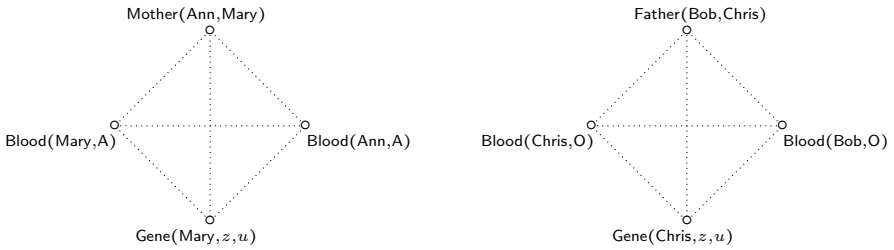

Fig. 9 The obstruction graph associated to the support set of the language in Fig. 8. The ground atoms of Gene are compactly represented by $\mathsf{Gene}(\mathsf{Mary}, z, u)$ and $\mathsf{Gene}(\mathsf{Chris}, z, u)$

Corollary 3 *There exists an efficient mistake-bound L2R algorithm for any probabilistic reasoning problem* $(\mathcal{P}, \mathcal{Q})$ *where* $\mathcal{Q}$ *is a* k*-cluster-width query language.*

6 Related Work

The framework described here has connections with work in many other areas related to reasoning and learning. Relational queries and their probabilistic estimation are central, for example, to both databases and Artificial Intelligence, while learning in uncertain and relational domains has been studied in numerous settings. Our framework is distinguished by the close way in which it integrates learning and reasoning, and by the insistence on having the measures of both computational complexity and of accuracy bounded by polynomial functions. Among the various alternative approaches that are relevant to ours, we concentrate on three.

6.1 Statistical Relational Learning

Statistical relational learning (SRL) is concerned with learning in structural and uncertain environments. As many real-world applications require to handle both aspects, this research field has received a great deal of attention in the literature.

There are at least three key dimensions that must be taken into account in any SRL framework. The first dimension is concerned with the *type* of learning task. In a *learning to classify* problem, the overall goal is to produce a relational probabilistic classifier capable of separating examples according to their class; the performance of the learner is measured by how well it classifies future examples. By contrast, in a *learning to reason* problem, the goal is to produce a relational probabilistic reasoner capable of quantifying the uncertainty of various events according to its knowledge; the performance of the learner is here measured by its ability to reason about the environment. The second dimension is characterized by the *representation language* into which hypotheses are described. Various representation languages have emerged in the literature, using either probabilistic extensions of logical formalisms, or dually, relational extensions of graphical formalisms; see e.g. (Getoor & Taskar, 2007) for a recent overview on these formalisms. The third dimension is concerned by the *cover relation* between hypotheses and examples (De Raedt & Kersting, 2004). For instance, in *learning from interpretations*, examples are described by relational structures, such as Herbrand interpretations, and the cover relation is defined by

the probability that an interpretation is a model of the given hypothesis. In *learning from entailment*, examples are queries, and the cover relation is specified by the degree of belief assigned to a query according to the hypothesis.

Most current SRL frameworks are devoted to classification tasks; they essentially differ in the choice of the representation language and the type of cover relation. In contrast, our framework is concerned with reasoning problems, in the spirit of learning to reason approaches. Despite this conceptual difference, our framework has connections with Markov networks, which have often been advocated as a compact and flexible representation language in statistical learning (Taskar et al., 2002, 2003, 2004; Kok & Domingos, 2005; Richardson & Domingos, 2006; Mihalkova et al., 2007). A *ground Markov network* represents a probability distribution over the sample space $\wp(\mathcal{H})$ of a set $\mathcal{H}$ of ground atoms. It is composed of an undirected graph that associates to each ground atom a vertex, and a set of potential functions associated to each clique of the graph. A potential function is a non-negative real function of the state of its corresponding clique. According to the terminology of this paper, a potential function F can be specified by its projection $\phi(F)$ and its weight w. The probability distribution associated to a Markov network M is given by

$$\mathbf{Pr}_M(I_i) = \frac{1}{Z} \prod_{(F,w)\in M} w^{\phi_i(F)}$$

where $Z = \sum_{i=1}^{N} \prod_{(F,w)\in M} w^{\phi_i(F)}$ is the normalizing partition function. First-order extensions of Markov networks, such as Markov relational networks (Taskar et al., 2002) and Markov logic networks (Richardson & Domingos, 2006), can be viewed as *templates* for constructing ground networks. For example, a Markov logic network is just a set of weighted formulas (F, w), where each ground instance of F represents a potential function with weight w. The probability distribution associated to a Markov logic network M is therefore given by

$$\mathbf{Pr}_M(I_i) = \frac{1}{Z} \prod_{(F,w)\in M} w^{\psi_i(F)}$$

where $\psi_i(F)$ is the number of ground instances of F that are true in I_i. From this viewpoint, Markov relational networks constitute a subclass of Markov logic networks where formulas are restricted to conjunctive queries.

The expressiveness of Markov logic networks and their variants comes, however, with two important sources of complexity. First, the problem of determining the number $\psi_i(F)$ of ground instances of a formula that are true in an interpretation I_i is $\#P$-hard (Richardson & Domingos, 2006). Second, the problem of inferring the degree of belief of any query Q from a Markov logic network M is also $\#P$-hard, even if Q is a simple atom and M a ground network (Roth, 1996).

In contrast, our framework introduces a new tractable class of Markov networks that handles both sources of complexity. Indeed, any set of weighted decomposable conjunctions KB gives rise to a Markov network, in a natural way, by associating to each weighted formula (F, w) in KB a corresponding potential function. The graphical structure of the network is formed by associating a vertex to each quantified atom occurring in KB, and an edge to each pair of quantified atoms occurring in the same formula of KB. Thus, any clique in which the conjunction of nodes is a formula F of KB, is associated with the potential function (F, w). The computational cost of determining the value $\phi_i(F)$ for any decomposable conjunction F is linear in the input dimension of the vocabulary. In addition, the cost of inferring the degree of belief of any decomposable query Q from the network KB is also polynomial in the

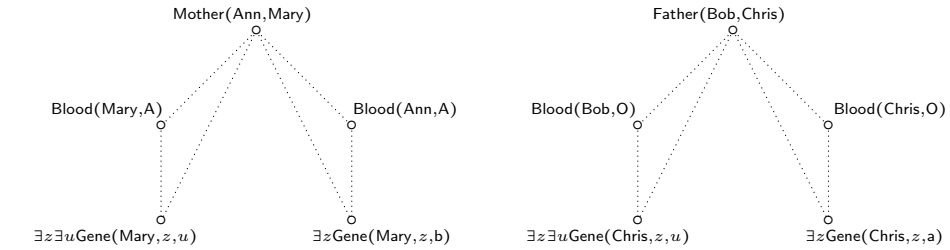

Fig. 10 The Markov network obtained from the query language in Fig. 8.

input dimension, provided that the set of formulas occurring in $KB \cup \{Q\}$ is a cluster set. More generally, if the cluster-width of this set is fixed by a parameter k, the running time is then polynomial in the input dimension.

Example 9 Consider again the blood type domain, and suppose that our agent has made a mistake on each query presented in Figure 8. The graphical structure of the resulting Markov network is illustrated in Figure 10. For example, the clique $\{\mathsf{Blood}(\mathsf{Ann},\mathsf{A}), \mathsf{Blood}(\mathsf{Mary},\mathsf{A}), \exists z \mathsf{Gene}(\mathsf{Mary},z,\mathsf{b})\}$ is associated with the potential function $\big(\mathsf{Blood}(\mathsf{Ann},\mathsf{A}) \wedge \mathsf{Blood}(\mathsf{Mary},\mathsf{A}) \wedge \forall z \neg \mathsf{Gene}(\mathsf{Mary},z,\mathsf{b}), w\big)$, where w is the weight given by the exponentiated gradient update rule.

6.2 Knowledge Compilation

Knowledge compilation has also emerged as an active research field for addressing computational difficulties in reasoning (Selman & Kautz, 1996; Liberatore, 1998; Darwiche & Marquis, 2002; Del Val, 2005). The idea is to transform a model of the domain into a compiled structure, which is then used online to answer queries in polynomial time. The key motivation is to push as much of the computational overhead into the compilation phase, which is amortized over many queries. Although most of the work has focused on logical inference, knowledge compilation has recently been ported to the probabilistic setting, using arithmetic circuits (Darwiche, 2003). In particular, this approach has been employed for compiling relational Bayesian networks (Chavira et al., 2006). Technically, a relational model is first instantiated into a ground Bayesian network which is then compiled into an arithmetic circuit. Ground conjunctive queries are evaluated by differentiating the compiled circuits in time linear in their size.

Despite its undoubted success on improving the effectiveness of reasoning tasks, an important drawback of knowledge compilation is that the compiled structure is not guaranteed to be polynomial in the size of the input dimension, even in the restricted case where the compiled structure only approximates the initial theory (Del Val, 2005). Actually, most of computational problems that are interesting in commonsense reasoning are not "compilable"" (Liberatore, 1998). Notably, the size of a compiled circuit for a relational Bayesian network can grow exponentially in the size of its ground instance, and hence, doubly exponentially in the input dimension.

Our framework can be viewed as a form of *inductive knowledge compilation*. Specifically, the initial model of the domain plays the part of the reasoning query oracle; based on this interface, the learner iteratively compiles the queries that have led to a mistake. Thus, the compilation is imposed not on the theory of the domain but on frequent queries posed to the agent. We argue that our framework is a plausible one for commonsense situations, since in general, an uncertain and relational

domain can be very complex but the queries are relatively simple. Notably, in the setting of cluster query languages, we have shown that query evaluation is essentially linear in the number of mistakes.

6.3 Learning to Reason

Naturally, this study is reminiscent to previous approaches on learning to reason. The foundations of this paradigm have been established by Khardon & Roth (1997, 1999) in the setting of propositional reasoning. In a similar spirit, the neuroidal architecture of Valiant (2000a,b) handles the relational setting by characterizing a variant of the language $\mathcal{R}$ for which learning and entailment are polynomially bounded. Our framework attempts to move one step further by showing that expressive fragments of $\mathcal{R}$ are also tractable for relational probabilistic reasoning.

The approach closest to ours is the "Bayesian" learning to reason framework developed by Greiner et al. (1997). Recall that a propositional Bayesian network is formed by two parts: a *qualitative* part, or structure, which consists in a directed acyclic graph where vertices represent variables and edges represent dependencies, and a *quantitative* part that associates to each vertex a conditional probability table that quantifies the effects of the parents of the vertex on itself. The aim of the Bayesian L2R framework is to induce from a sample of queries, each valued by some hidden target distribution, a Bayesian network that compactly encodes the target distribution and accurately evaluates the degree of belief of future queries.

In the Bayesian L2R framework, the hypothesis language is the class of all Bayesian networks that can be built from the vocabulary. This structural choice has, however, two important consequences. First, as shown by the authors, the problem of finding a Bayesian network that minimizes the quadratic loss on a given set of queries is NP-hard even in the restricted setting where the structure of the network is fixed and the learner has just to fill the CP-tables. Second, the tractability of query evaluation in Bayesian networks is only guaranteed for very restricted query languages, such as Markov-blanket queries (Pearl, 1988). In most cases, including even atomic queries, probabilistic reasoning is #P-hard (Roth, 1996). By contrast, in the setting suggested by our framework, the hypothesis language is dependent on the choice of the target query language. Namely, the hypothesis maintained by the learner is formed by associating a weight to each query that has led to a mistake. This dependence, coupled with an exponentiated weight update learning strategy, enables us to handle more expressive relational query languages that are tractable for weighted model counting.

7 Conclusions

Along the lines of making degrees of belief applicable under tractable conditions, this study has stressed the importance of combining learning and reasoning processes together. Specifically, we have shown that some restricted though expressive classes of relational probabilistic reasoning problems are mistake-bound learnable.

Clearly, there are many directions in which one might attempt extensions of this framework. A first direction is to extend the expressiveness of query languages while maintaining computational efficiency. For example, a conjunctive query such

as $\exists x \exists z \, (\mathsf{Mother}(\mathsf{Ann}, x) \wedge \mathsf{Gene}(x, z, \mathsf{a}))$ can be transformed into a linear number of non-overlapping decomposable conjunctions; model counting can hence be evaluated in polynomial time. A second direction is to explore other learning strategies such as, for example, quasi-additive algorithms (Grove et al., 2001; Gentile, 2003) which might open the door to new learnability results. A third direction is to examine other types of learning interfaces. In fact, the results obtained in this paper rely on a "quantitative" oracle capable of supplying the correct probability of input queries. Yet, the interface is not always a perfect image of the environment, and it might be interesting to explore "qualitative" oracles, limited to supply a propositional attitude, or "noisy" interfaces, susceptible to make perception errors.

Finally, we have restricted this study to degrees of belief, sometimes referred to as unconditional epistemic probabilities (Halpern, 2003). This assumption is justifiable in many situations where the learner is allowed to decompose conditional queries of the form $G|F$. Namely, suppose that during the grace period, antecedents and consequents of conditional queries are received separately. Then, after this period, the learner can estimate $\mathbf{Pr}_{\mathcal{P}}(G|F)$, using $\mathbf{Pr}_{KB}(G \wedge F)$ and $\mathbf{Pr}_{KB}(F)$, provided that the formulas occurring in conditional queries form a decomposable language with bounded cluster-width. Nevertheless, in some situations we are not necessarily informed about the context of events, and it is legitimate to learn directly from arbitrary queries. From this perspective, the problem of learning to reason about conditionals looks challenging.

Appendix

Theorem 1 *For any relational probabilistic reasoning problem $(\mathcal{P}, \mathcal{Q})$ defined over a vocabulary of dimension d, on input $\gamma > 0$, when $\eta = 4$, the direct EG-L2R algorithm has the following mistake bound*

$$\mathrm{M}_{\gamma}(\mathcal{P}, \mathcal{Q}) \leq \frac{\ln 2}{2\gamma} \big(d - H(\mathcal{P})\big)$$

Proof The analysis follows Kivinen & Warmuth (1997) with a slight difference in the derived bound. Let $S = (Q_1, \cdots, Q_T)$ be a sequence of queries supplied by the oracle. We proceed by showing that the divergence between $\hat{\mathcal{P}}_t$ and $\mathcal{P}$ decreases after each mistake. The divergence function employed for the analysis is the Kullback-Leibler (KL) distance

$$D(\mathcal{P}, \hat{\mathcal{P}}_t) = \sum_{i=1}^{N} p_i \ln(p_i / \hat{p}_{t,i})$$

It is well known that the KL distance is minimal and equal to 0 if and only if $\mathcal{P} = \hat{\mathcal{P}}_t$. Based on this divergence function, we first observe that

$$\sum_{t=1}^{T} D(\mathcal{P}, \hat{\mathcal{P}}_t) - D(\mathcal{P}, \hat{\mathcal{P}}_{t+1}) = D(\mathcal{P}, \hat{\mathcal{P}}_1) - D(\mathcal{P}, \hat{\mathcal{P}}_{T+1}) \leq \big(d - H(\mathcal{P})\big) \ln 2 \qquad (1)$$

We thus need to prove that the total number of mistakes is bounded by this sum. In doing so, let us examine the difference of divergences after each mistake. We have

$$\begin{aligned} D(\mathcal{P},\hat{\mathcal{P}}_t) - D(\mathcal{P},\hat{\mathcal{P}}_{t+1}) &= \sum_{i=1}^{N} p_i \ln\left(\frac{e^{\eta\phi_i(Q_t)(y_t-\hat{y}_t)}}{\sum_{j=1}^{N}\hat{p}_{t,j}e^{\eta\phi_j(Q_t)(y_t-\hat{y}_t)}}\right) \\ &= \eta(y_t-\hat{y}_t)y_t - \ln\left(\sum_{i=1}^{N}\hat{p}_{t,i}e^{\eta\phi_i(Q_t)(y_t-\hat{y}_t)}\right) \\ &= \eta(y_t-\hat{y}_t)y_t - \ln\left(1-\hat{y}_t\left(1-e^{\eta(y_t-\hat{y}_t)}\right)\right) \end{aligned}$$

The last equality is derived from the fact that the exponentiated term is a convex function of the form $\alpha^{\phi_i(Q_t)}$ with $\alpha = e^{\eta(y_t-\hat{y}_t)} > 0$ and $\phi_i(Q_t) \in \{0,1\}$. Now, by Lemma 1 in (Helmbold et al., 1997), we know that $\ln(1-z(1-e^x)) \le xz + \frac{1}{8}x^2$. By substituting this inequality into the main equation, we have

$$\begin{aligned} D(\mathcal{P},\hat{\mathcal{P}}_t) - D(\mathcal{P},\hat{\mathcal{P}}_{t+1}) &\ge \eta y_t(y_t-\hat{y}_t) - \eta\hat{y}_t(y_t-\hat{y}_t) - \frac{\eta^2(y_t-\hat{y}_t)^2}{8} \\ &= \eta\left(1-\frac{\eta}{8}\right)(y_t-\hat{y}_t)^2 \end{aligned}$$

Summing over all trials and using the upper bound (1), we obtain

$$\sum_{t=1}^{T}(y_t-\hat{y}_t)^2 \le \frac{8\ln 2}{\eta(8-\eta)}\big(d - H(\mathcal{P})\big) \tag{2}$$

The cumulative loss is minimized when $\eta = 4$. For this value, we have

$$\sum_{t=1}^{T}(y_t-\hat{y}_t)^2 \le \frac{\ln 2}{2}\big(d - H(\mathcal{P})\big) \tag{3}$$

Finally, we know that a mistake arises only when $(y_t-\hat{y}_t)^2 > \gamma$. By substituting this condition into inequality (3) we obtain the desired bound. □

Theorem 2 *The indirect EG-L2R algorithm exactly simulates the direct EG-L2R algorithm.*

Proof We assume that both algorithms are run on the same parameter η and the same sequence $S = (Q_1,\cdots,Q_T)$ of queries supplied by the same oracle $\mathrm{RQ}_\gamma(\mathcal{P},\mathcal{Q})$. We must show that $\mathbf{Pr}_{\hat{\mathcal{P}}_t}(Q_t) = \mathbf{Pr}_{KB_t}(Q_t)$ on each trial $t = 1,\cdots,T$. Given an interpretation I in $\wp(\mathcal{H})$, let $\widehat{I}$ be the conjunction of all ground literals that are true in I. Note that $\widehat{I}$ has a unique model, which is I. It follows that $\mathbf{Pr}_{\hat{\mathcal{P}}_t}(\widehat{I}_i) = \hat{p}_{t,i}$ for $1 \le i \le N$. Now, let us consider the following invariant: for any trial $t \ge 1$ and any interpretation I in $\wp(\mathcal{H})$, $\mathbf{Pr}_{\hat{\mathcal{P}}_t}(\widehat{I}) = \mathbf{Pr}_{KB_t}(\widehat{I})$. This is a sufficient condition to prove that $\mathbf{Pr}_{\hat{\mathcal{P}}_t}(Q_t) = \mathbf{Pr}_{KB_t}(Q_t)$. Indeed, we have

$$\begin{aligned} \mathbf{Pr}_{\hat{\mathcal{P}}_t}(Q_t) &= \sum_{i=1}^{N}\hat{p}_{t,i}\phi_i(Q_t) = \sum_{I\in\mathcal{M}(Q_t)}\mathbf{Pr}_{\hat{\mathcal{P}}_t}(\widehat{I}) = \sum_{I\in\mathcal{M}(Q_t)}\mathbf{Pr}_{KB_t}(\widehat{I}) \\ &= \sum_{I\in\mathcal{M}(Q_t)}\frac{\|KB_t\cup\{\widehat{I}\}\|}{\|KB_t\|} = \frac{\|KB_t\cup\{Q_t\}\|}{\|KB_t\|} = \mathbf{Pr}_{KB_t}(Q_t) \end{aligned}$$

The existence of this invariant is proven by induction on the number of trials. Let $t = 1$. Here, the knowledge base KB_1 is reduced to $\{(\top, 1)\}$. We have

$$\mathbf{Pr}_{\hat{\mathcal{P}}_1}(\widehat{I}) = \frac{1}{N} = \frac{\|\widehat{I}\|}{\sum_{i=1}^{N} \|\widehat{I}_i\|} = \frac{\|KB_1 \cup \{\widehat{I}\}\|}{\|KB_1\|} = \mathbf{Pr}_{KB_1}(\widehat{I})$$

Now, consider that $t > 1$ and assume by induction hypothesis that the property holds at the beginning of the trial. We have $\hat{y}_t = \mathbf{Pr}_{\hat{\mathcal{P}}_t}(Q_t) = \mathbf{Pr}_{KB_t}(Q_t)$. On receiving the oracle's response, two cases are possible. First, suppose that $L(y_t, \hat{y}_t) \leq \epsilon$. In this case, the property trivially holds at the end of the trial. Now suppose that $L(y_t, \hat{y}_t) > \epsilon$. Recall that $KB_{t+1} = KB_t \cup \{(Q_t, w_t)\}$, where $w_t = e^{\eta(y_t - \hat{y}_t)}$. The property is proved by the following chain of equalities.

$$\begin{aligned}
\mathbf{Pr}_{\hat{\mathcal{P}}_{t+1}}(\widehat{I}_i) &= \frac{\hat{p}_{t,i} e^{\eta \phi_i(Q_t)(y_t - \hat{y}_t)}}{\sum_{j=1}^{N} \hat{p}_{t,j} e^{\eta \phi_j(Q_t)(y_t - \hat{y}_t)}} = \frac{\mathbf{Pr}_{\hat{\mathcal{P}}_t}(\widehat{I}_i) w_t^{\phi_i(Q_t)}}{\sum_{j=1}^{N} \mathbf{Pr}_{\hat{\mathcal{P}}_t}(\widehat{I}_j) w_t^{\phi_j(Q_t)}} \\
&= \frac{\mathbf{Pr}_{KB_t}(\widehat{I}_i) w_t^{\phi_i(Q_t)}}{\sum_{j=1}^{N} \mathbf{Pr}_{KB_t}(\widehat{I}_j) w_t^{\phi_j(Q_t)}} = \frac{\mathbf{Pr}_{KB_{t+1}}(\widehat{I}_i)}{\sum_{j=1}^{N} \mathbf{Pr}_{KB_{t+1}}(\widehat{I}_j)} = \mathbf{Pr}_{KB_{t+1}}(\widehat{I}_i)
\end{aligned}$$

It follows that $\mathbf{Pr}_{\hat{\mathcal{P}}_t}(Q_t) = \mathbf{Pr}_{KB_t}(Q_t)$ on any trial $t = 1, \cdots, T$, as desired. □

References

Abadi, M., & Halpern, J. Y. (1994). Decidability and expressiveness for first-order logics of probability. *Information and Computation*, *112*(1), 1–36.

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*(4), 319–342.

Bacchus, F., Grove, A. J., Halpern, J. Y., & Koller, D. (1996). From statistical knowledge bases to degrees of belief. *Artificial Intelligence*, *87*(1-2), 75–143.

Büning, H. K., & Zhao, X. (2001). Satisfiable formulas closed under replacement. *Electronic Notes in Discrete Mathematics*, *9*, 48–58.

Bylander, T. (1998). Worst-case analysis of the perceptron and exponentiated update algorithms. *Artificial Intelligence*, *106*(2), 335–352.

Cesa-Bianchi, N. (1999). Analysis of two gradient-based algorithms for on-line regression. *Journal of Computer and System Sciences*, *59*(3), 392–411.

Chavira, M., & Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence*. To appear.

Chavira, M., Darwiche, A., & Jaeger, M. (2006). Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning*, *42*(1-2), 4–20.

Chen, J., Kanj, I. A., & Xia, G. (2005). Simplicity is beauty: Improved upper bounds for vertex cover. Tech. Rep. TR05-008, De Paul University, Chicago, IL.

Costa, V. S., Page, D., Qazi, M., & Cussens, J. (2003). CLP(BN): Constraint logic programming for probabilistic knowledge. In *Proc. of the Nineteenth Conference in Uncertainty in Artificial Intelligence*, (pp. 517–524). Acapulco, Mexico: Morgan Kaufmann.

Cox, R. T. (1946). Probability, frequency, and reasonable expectation. *American Journal of Physics*, *14*, 1–13.

Cumby, C. M., & Roth, D. (2000). Relational representations that facilitate learning. In *Proceedings or the Seventeenth International Conference on the Principles of Knowledge Representation and Reasoning*, (pp. 425–434). Breckenridge, CO: Morgan Kaufmann.

Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *Journal of the ACM*, *50*(3), 280–305.

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, *17*, 229–264.

De Raedt, L., & Kersting, K. (2004). Probabilistic inductive logic programming. In *Proceedings ot the Fifteenth International Conference on Algorithmic Learning Theory*, (pp. 19–36). Padova, Italy: Springer.

Del Val, A. (2005). First order LUB approximations: characterization and algorithms. *Artificial Intelligence*, *162*(1-2), 7–48.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, (pp. 1300–1309). Stockholm, Sweden: Morgan Kaufmann.

Gentile, C. (2003). The robustness of the p-norm algorithms. *Machine Learning*, *53*(3), 265–299.

Getoor, L., & Taskar, B. (2007). *Introduction to Statistical Relational Learning*. Cambridge, MA: The MIT Press.

Greiner, R., Grove, A. J., & Schuurmans, D. (1997). Learning Bayesian nets that perform well. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, (pp. 198–207). Providence, RI: Morgan Kaufmann.

Grove, A. J., Halpern, J. Y., & Koller, D. (1994). Random worlds and maximum entropy. *Journal of Artificial Intelligence Research*, *2*, 33–88.

Grove, A. J., Littlestone, N., & Schuurmans, D. (2001). General convergence results for linear discriminant updates. *Machine Learning*, *43*(3), 173–210.

Halpern, J. Y. (2003). *Reasoning about Uncertainty*. Cambridge, MA: MIT Press.

Helmbold, D. P., Schapire, R. E., Singer, Y., & Warmuth, M. K. (1997). A comparison of new and old algorithms for a mixture estimation problem. *Machine Learning*, *27*(1), 97–119.

Jaeger, M. (1997). Relational Bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, (pp. 266–273). Providence, RI: Morgan Kaufmann.

Jaeger, M. (2000). On the complexity of inference about probabilistic relational models. *Artificial Intelligence*, *117*(2), 297–308.

Kersting, K. (2006). *An Inductive Logic Programming Approach to Statistical Relational Learning*, vol. 148 of *Frontiers in Artificial Intelligence and Applications*. Amsterdam, Netherlands: IOS Press.

Khardon, R. (1999). Learning to take actions. *Machine Learning*, *35*(1), 57–90.

Khardon, R., & Roth, D. (1997). Learning to reason. *Journal of the ACM*, *44*(5), 697–725.

Khardon, R., & Roth, D. (1999). Learning to reason with a restricted view. *Machine Learning*, *35*(2), 95–116.

Kivinen, J., & Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, *132*(1), 1–63.

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. In *Proceedings of the Twenty-Second International Conference in Machine Learning*, (pp. 441–448). Bonn, Germany: ACM.

Liberatore, P. (1998). *Compilation of intractable problems and its application to artificial intelligence*. Ph.D. thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Rome, Italy.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*(4), 285–318.

Littlestone, N. (1989). *Mistake bounds and logarithmic linear-threshold learning algorithms*. Ph.D. thesis, University of California, Santa Cruz, CA.

Manzano, M. (2005). *Extensions of First-Order Logic*. Cambridge, UK: Cambridge University Press.

Mihalkova, L., Huynh, T., & Mooney, R. J. (2007). Mapping and revising Markov logic networks for transfer learning. In *Proceedings of the Twenty-Second AAAI Conference*

on Artificial Intelligence, (pp. 608–614). Vancouver, Canada: AAAI Press.

Muggleton, S. (1996). Stochastic logic programs. In L. D. Readt (Ed.) *Advances in Inductive Logic Programming*, (pp. 254–264). Amsterdam, Netherlands: IOS Press.

Ngo, L., & Haddawy, P. (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, *171*(1-2), 147–177.

Nishimura, N., Ragde, P., & Szeider, S. (2006). Solving #SAT using vertex covers. In *Proceedings of the Ninth International Conference in Theory and Applications of Satisfiability Testing*, (pp. 396–409). Seattle, WA: Springer.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

Pfeffer, A. (2000). *Probabilistic Reasoning for Complex Systems*. Ph.D. thesis, Computer Science Department, Stanford University, CA.

Poole, D. (1993). Probabilistic horn abduction and Bayesain networks. *Artificial Intelligence*, *64*, 81–129.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*(1-2), 107–136.

Roth, D. (1996). On the hardness of approximate reasoning. *Artificial Intelligence*, *82*(1-2), 273–302.

Ruan, Y., Kautz, H. A., & Horvitz, E. (2004). The backdoor key: A path to understanding problem hardness. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, (pp. 124–130). San Jose, CA: AAAI Press.

Sang, T., Beame, P., & Kautz, H. A. (2005). Performing Bayesian inference by weighted model counting. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, (pp. 475–482). Pittsburgh, PA: AAAI Press.

Selman, B., & Kautz, H. A. (1996). Knowledge compilation and theory approximation. *Journal of the ACM*, *43*(2), 193–224.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference in Uncertainty in Artificial Intelligence*, (pp. 485–492). Edmonton, Alberta, Canada: Morgan Kaufmann.

Taskar, B., Chatalbashev, V., & Koller, D. (2004). Learning associative Markov networks. In *Proceedings of the Twenty-first International Conference in Machine Learning*. Banff, Alberta, Canada: ACM.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*. Vancouver, British Columbia, Canada: MIT Press.

Valiant, L. G. (1994). *Circuits of the Mind*. New-York, NY: Oxford University Press.

Valiant, L. G. (2000a). A neuroidal architecture for cognitive computation. *Journal of the ACM*, *47*(5), 854–882.

Valiant, L. G. (2000b). Robust logics. *Artificial Intelligence*, *117*(2), 231–253.

Williams, R., Gomes, C. P., & Selman, B. (2003). Backdoors to typical case complexity. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, (pp. 1173–1178). Acapulco, Mexico: Morgan Kaufmann.