Abstracting Markov Networks

L. Saitta
Università del Piemonte Orientale
Viale Teresa Michel 11
15121 Alessandria, Italy

C. Vrain
Université d'Orléans
LIFO - BP 6759
45067 Orléans, France

Abstract

In this paper we describe a preliminary investigation on the use of abstraction operators to reduce the complexity of inference in Markov Networks. More specifically, we are interested in Logic Markov Network, where the use of abstraction may be a complementary approach to lifted inference.

1 Introduction

Markov networks have proved to be a very useful tool to represent probability distributions over large domains (see for instance, Chapter 8 in (Bishop 2006)). A Markov Network is an undirected graphical model, where variables are represented by nodes and features on subsets of variables by cliques in the graph. The main characteristics of such a model can be described as follows:

- The joint probability distribution over the variables is defined as the normalized product of potential functions over the maximal cliques of the graph.
- Conditional independence: X is independent of Y given Z, written $X \coprod Y|Z$, when all possible paths between nodes of X and nodes of Y pass through at least one node of Z.

One of the major issue in graphical model is the computational complexity of reasoning, which can be decomposed into several inference problems, such as, for instance, finding the marginal distribution of a variable X, or computing the probability of a query q given evidence e ($P(q|e)$). Let us consider, for instance, the first problem. To compute $P(X)$, a straightforward but naive approach consists in summing the joint distribution over all variables except X: with N Boolean variables, this leads to a sum on 2^{N-1} terms. A complexity of the order $\mathcal{O}(2^N)$ is also required to compute the normalization factor of the probability distribution, namely the *partition function* Z.

Techniques have been developed to reduce the complexity of inference, either relying on the structure of the graph to compute exact inference, or using sampling techniques for computing approximate inference. For instance, (Bishop 2006) presents the *sum-product* algorithm (Kschischang, Frey, and Loeliger 2001) for undirected tree: it relies on the notion of factor graph that completes the graph with additional nodes, one for each maximal clique, and that allows, in some sense, to exchange the sum and the product in the expression of the marginal $P(X)$. Approximate inference usually relies on Markov Chain Monte Carlo sampling techniques, as for instance Gibbs sampling.

Nowawdays, the problem of inference complexity is all the more acute with the emergence of Statistical Relational Learning, which aims at combining probabilistic graphical models with first order logics representations. The work that we present in this paper has been motivated by Markov Logic Networks (MLN), introduced in (Richardson and Domingos 2006). A Markov Logic Network is defined by a set of weighted first-order formulas. A Markov network is built by associating a node to each ground atom[1] and two ground atoms are linked in the graph if they occur in the same instantiated formula. The complexity, in terms of the number of nodes in the graph, depends on the number of constants, the number of predicates and their arity. For instance, if we suppose that the domain is composed of n constants, and we have m binary predicates, then the number of ground atoms, and therefore the number of nodes in the graph, is $m \times n^2$.

To reduce the problem of inference complexity, mainly two families of approaches have been proposed. The first one considers the ground Makov network and uses either inference algorithms based on SAT algorithm (Poon and Domingos 2006) or Gibbs sampling (Richardson and Domingos 2006; Milch and Russell 2006). To overcome the complexity problem due to the size of the ground networks, methods have been developed to ground atoms and formulas only when needed (Singla and Domingos 2006; Riedel 2008; Poon, Domingos, and Sumner 2008).

The aim of the second approach is to avoid building the ground Markov networks. For instance first-order probabilistic inference extends the variable elimination algorithm to first-order logic (Poole 2003); first-order lifted inference extends Poole's work by introducing a counting elimination method (de Salvo Braz, Amir, and Roth 2005). In (Singla

[1]With function-free language, a ground atom is an expression $p(a_1, \dots, a_n)$, where n is the arity of the predicate p and $a_1, \dots, a_n$ are constants.

and Domingos 2008), the authors propose to build lifted networks: supernodes and superfeatures are introduced allowing to respectively group the ground atoms and the groundings of the formulas with the same behavior.

In this paper, we describe a preliminary investigation of a third line of attack, namely the use of *abstraction* for reducing the complexity of inference in Markov networks. Abstraction, in this context, aims at transforming the original network into a smaller/simpler one, which can be handled with a reduced complexity, still preserving some required properties. Graph abstraction for specific tasks have been proposed early in the literature. For instance, Holte et al. (Holte et al. 1996) have described a STAR abstraction operator that allows significant speed up in problem solving. Clustering of nodes in a graph, detection of "communities", and building up hierarchies and multi-scale views can all be considered as forms of abstraction (Epstein and Li 2009; Zhang, Ning, and Zhang 2007; Clauset, Moore, and Newman 2008; Arenas, Fernández, and Gómez 2008; Saitta, Henegar, and Zucker 2009; Harry and Lindquist 2004; Bulitko et al. 2007).

In this paper we concentrate on preserving properties related to the probabilistic inferences that are required in Markov networks, taking into account their specific characteristics.

2 Markov Networks with Boolean Variables

Given a vector $\vec{\mathbf{X}} = (X_1, X_2, ...X_N)$ of stochastic Boolean variables, which assumes values $\vec{\mathbf{x}} = (x_1, x_2, ..., x_N) \in \mathcal{X} = \{0,1\}^N$, a Markov network $\mathcal{G}$ represents the joint probability distribution over $\vec{\mathbf{X}}$. The network has N nodes and M cliques. An assignment of values to $\vec{\mathbf{x}}$ is a *world*. The set of all worlds, $\mathcal{X}$, contains 2^N elements.

The probability over $\vec{\mathbf{X}}$ can be expressed as a product of M *potential functions*, each one associated to a clique of $\mathcal{G}$:

$$P(\vec{\mathbf{X}} = \vec{\mathbf{x}}) =_{def} P(\vec{\mathbf{x}}) = \frac{1}{Z}\prod_{k=1}^{M} \varphi_k(\vec{\mathbf{x}}) \qquad (1)$$

In (1) Z is the *partition function*, which is a normalization factor defined by:

$$Z = \sum_{\vec{\mathbf{x}} \in \mathcal{X}} \prod_{k=1}^{M} \varphi_k(\vec{\mathbf{x}}) \qquad (2)$$

In order to estimate the computational complexity of evaluating Z, let us assume, as a unit, the complexity required to compute one potential function. Then:

$$\mathbf{C}(N, M) = M \cdot 2^N \qquad (3)$$

3 Partitioning a Markov Network

Given a Markov network $\mathcal{G}$, let us partition the *cliques* of $\mathcal{G}$ into two sets, corresponding to subgraphs $\mathcal{G}_1$ and $\mathcal{G}_2$, where $\mathcal{G}_1$ contains h cliques, and $\mathcal{G}_2$ contains $(M-h)$ cliques. The nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$ can be partitioned into three sets, namely, $\vec{\mathbf{U}}$, containing the nodes occurring only in $\mathcal{G}_1$, $\vec{\mathbf{V}}$, containing the nodes occurring only in $\mathcal{G}_2$, and $\vec{\mathbf{T}}$, containing the nodes shared by $\mathcal{G}_1$ and $\mathcal{G}_2$. The values taken on by $\vec{\mathbf{U}}$, grouping r variables, $\vec{\mathbf{T}}$, grouping s variables, and $\vec{\mathbf{V}}$, grouping $(N - r - s)$ variables, are contained in three subvectors of $\vec{\mathbf{x}}$, *i.e.*, $\vec{\mathbf{u}} \in \mathcal{X}_u$, $\vec{\mathbf{t}} \in \mathcal{X}_t$, and $\vec{\mathbf{v}} \in \mathcal{X}_v$, respectively. We have:

$$|\mathcal{X}_u| = 2^r \qquad |\mathcal{X}_t| = 2^s \qquad |\mathcal{X}_v| = 2^{N-r-s}$$

Without loss of generality we may assume that $\vec{\mathbf{u}}$ contains the first r variables $(X_1, ..., X_r)$, $\vec{\mathbf{t}}$ contains the subsequent s variables $(X_{r+1}, ..., X_{r+s})$, and $\vec{\mathbf{v}}$ contains the last $(N - r - s)$ variables $(X_{r+s+1}, ..., X_N)$:

$$\begin{aligned} \vec{\mathbf{u}} &= (x_1, x_2, ..., x_r) \\ \vec{\mathbf{t}} &= (x_{r+1}, ..., x_{r+s}) \\ \vec{\mathbf{v}} &= (x_{r+s+1}, ..., x_N) \end{aligned}$$

Let us now compute the probability of a generic world $\vec{\mathbf{x}}$, by taking into account the partition of $\mathcal{G}$ into $\mathcal{G}_1$ and $\mathcal{G}_2$:

$$P(\vec{\mathbf{x}}) = \frac{1}{Z}\prod_{k=1}^{h} \varphi_k(\vec{\mathbf{u}}, \vec{\mathbf{t}}) \prod_{k=h+1}^{M} \varphi_k(\vec{\mathbf{v}}, \vec{\mathbf{t}}) \qquad (4)$$

From (4) we want to compute the partition function Z:

$$Z = \sum_{\vec{\mathbf{x}} \in \mathcal{X}} \left(\prod_{k=1}^{h} \varphi_k(\vec{\mathbf{u}}, \vec{\mathbf{t}}) \prod_{k=h+1}^{M} \varphi_k(\vec{\mathbf{v}}, \vec{\mathbf{t}}) \right) \qquad (5)$$

We can split the sum in (5) as follows:

$$Z = \sum_{\vec{\mathbf{t}} \in \mathcal{X}_t} \left[\sum_{\vec{\mathbf{u}} \in \mathcal{X}_u} \prod_{k=1}^{h} \varphi_k(\vec{\mathbf{u}}, \vec{\mathbf{t}}) \sum_{\vec{\mathbf{v}} \in \mathcal{X}_v} \prod_{k=h+1}^{M} \varphi_k(\vec{\mathbf{v}}, \vec{\mathbf{t}}) \right] \qquad (6)$$

In expression (6) the two internal sums are independent, for each value of $\vec{\mathbf{t}}$. In order to compute Z from (6) we obtain a complexity (in the number of evaluations of potential functions):

$$\mathbf{C}(N, M, r, s, h) = \mathcal{O}\left(2^{r+s}h + 2^{N-r}(M-h)]\right)$$

In equation (6), let us define:

$$\Phi_1(\vec{\mathbf{u}}, \vec{\mathbf{t}}) = \prod_{k=1}^{h} \varphi_k(\vec{\mathbf{u}}, \vec{\mathbf{t}}) \;\text{ and }\; \Phi_2(\vec{\mathbf{v}}, \vec{\mathbf{t}}) = \prod_{k=h+1}^{M} \varphi_k(\vec{\mathbf{v}}, \vec{\mathbf{t}}) \quad (7)$$

Moreover:

$$\Psi_1(\vec{\mathbf{t}}) = \sum_{\vec{\mathbf{u}} \in \mathcal{X}_u} \Phi_1(\vec{\mathbf{u}}, \vec{\mathbf{t}}) \;\text{ and }\; \Psi_2(\vec{\mathbf{t}}) = \sum_{\vec{\mathbf{v}} \in \mathcal{X}_v} \Phi_2(\vec{\mathbf{v}}, \vec{\mathbf{t}}) \quad (8)$$

Finally, equation (6) can be rewritten as:

$$Z = \sum_{\vec{\mathbf{t}} \in \mathcal{X}_t} \Psi_1(\vec{\mathbf{t}}) \cdot \Psi_2(\vec{\mathbf{t}}) \qquad (9)$$

The above results can be generalized to the case in which the graph $\mathcal{G}$ is subdivided into R subgraphs $\mathcal{G}_k$ $(1 \leq k \leq R)$, each containing a different set of h_k cliques. We have then:

$$\mathcal{G} = \bigcup_{k=1}^{R} \mathcal{G}_k \quad \text{and} \quad \sum_{k=1}^{R} h_k = M$$

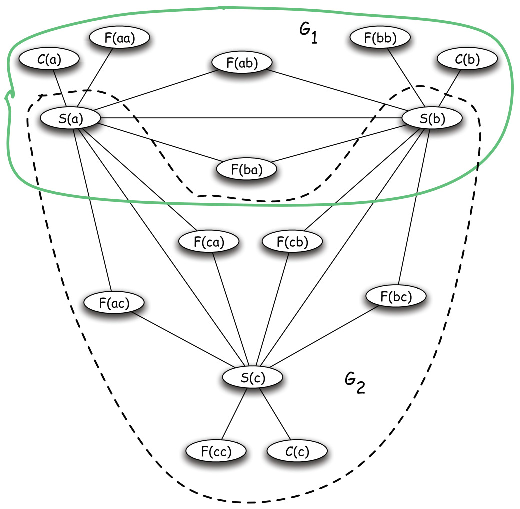

Figure 1: Example of complexity reduction. The graph is taken from (Richardson and Domingos 2006). Let $\mathcal{G}_1$ be the subgraph enclosed in the green line; then, $N = 15, M = 12, \vec{\mathbf{u}} = (C(a), F(a,a), F(a,b), F(b,a), F(b,b), C(b))$, $\vec{\mathbf{t}} = (S(a), S(b))$, $\vec{\mathbf{v}} = (F(a,c), F(c,a), F(c,b), F(b,c),$ $S(c), F(c,c), C(c))$, r = 8, s = 2, , h = 6. We have $\mathbf{C}(N,M) = 393,216$ and $\mathbf{C}(N,M,r,s,h) = 4608$ with a reduction in the number of evaluated potential functions of 98.8%.

The probability of a world $\vec{\mathbf{x}}$ can be written as follows:

$$P(\vec{\mathbf{x}}) = \frac{1}{Z}\prod_{k=1}^{R}\Phi_k(\vec{\mathbf{x}}^{(k)}) \tag{10}$$

where $\vec{\mathbf{x}}^{(k)}$ contains the variables that appear in $\mathcal{G}_k$. Each subgraph contributes to the probability distribution with the product:

$$\Phi_k(\vec{\mathbf{x}}^{(k)}) = \prod_{j=h_{k-1}+1}^{h_k}\varphi_j(\vec{\mathbf{x}}^{(k)}) \;\; (1 \le k \le R) \tag{11}$$

In order to compute Z, we have to use equation (16). In each $\Phi(\vec{\mathbf{x}}^{(k)})$ let us partition the set of variables into two groups: $\vec{\mathbf{u}}^{(k)} \in \mathcal{X}_u^{(k)}$ and $\vec{\mathbf{t}}^{(k)} \in \mathcal{X}_t^{(k)}$, such that the variables in $\vec{\mathbf{u}}^{(k)}$ only occur inside $\mathcal{G}_k$, whereas the variables in $\vec{\mathbf{t}}^{(k)}$ may also occur in some other $\mathcal{G}_j$. We may notice that the sets of variables $\vec{\mathbf{u}}^{(k)}$ are all disjoint, by definition, whereas the sets $\vec{\mathbf{t}}^{(k)}$ may share some variables among them. Then, let us define:

$$\vec{\mathbf{t}} = \bigcup_{k=1}^{R}\vec{\mathbf{t}}^{(k)} \in \mathcal{X}_t$$

As a consequence, Z can be rewritten as follows:

$$Z = \sum_{\vec{\mathbf{t}}\in\mathcal{X}_t}\left(\sum_{\vec{\mathbf{u}}^{(1)}\in\mathcal{X}_u^{(1)}}\Phi_1(\vec{\mathbf{u}}^{(1)},\vec{\mathbf{t}}) \;...\; \sum_{\vec{\mathbf{u}}^{(R)}\in\mathcal{X}_u^{(R)}}\Phi_R(\vec{\mathbf{u}}^{(R)},\vec{\mathbf{t}})\right)$$

For each assignment to the common variables $\vec{\mathbf{t}}$, the internal sums in Z are independent. We can define:

$$\Psi_k(\vec{\mathbf{t}}) = \sum_{\vec{\mathbf{u}}^{(k)}\in\mathcal{X}_{\vec{\mathbf{u}}^{(k)}}}\Phi_k(\vec{\mathbf{u}}^{(k)},\vec{\mathbf{t}}) \tag{12}$$

Finally, the partition function becomes:

$$Z = \sum_{\vec{\mathbf{t}}\in\mathcal{X}_t}\prod_{k=1}^{R}\Psi_k(\vec{\mathbf{t}}) \tag{13}$$

The complexity for computing Z through equation (13) can be evaluated by defining:

$$h = \max_{1\le k\le R} h_k \;\text{ and }\; s = |\vec{\mathbf{t}}| \;\text{ and }\; r = \max_{1\le k\le R}|\vec{\mathbf{u}}^{(k)}|$$

Then:

$$\mathbf{C}(N,M,R,r,s,h) = \mathcal{O}\left(2^s\sum_{k=1}^{R}h_k e^{|\vec{\mathbf{u}}^{(k)}|}\right) = \mathcal{O}(R\,h\;2^r)$$

Notice that $r < n$. The above formula also applies to the case of $R = 2$.

4 Abstraction in Markov Networks

As inference is exponential in the size of a Markov network, one way to make it less costly is to apply to it some abstraction operator, whose effect is to reduce its size. Let us consider a network $\mathcal{G}$ partitioned into R subnetworks, as in Section 3, and let $\mathcal{G}_k$ be one of such subnetwork. Let $\alpha(\vec{\mathbf{u}}^{(k)})$ be an *abstraction function* that lets the set of nodes that occur only in $\mathcal{G}_k$ collapse into a single Boolean node $\xi_k \in \{0,1\}$. More precisely:

$$\begin{aligned}\xi_k = \alpha(\vec{\mathbf{u}}^{(k)}) = 1 \quad &= \quad \text{if } \vec{\mathbf{u}}^{(k)} \in D_1^{(k)}\\ \xi_k = \alpha(\vec{\mathbf{u}}^{(k)}) = 0 \quad &= \quad \text{if } \vec{\mathbf{u}}^{(k)} \in D_0^{(k)}\end{aligned}$$

By applying the abstraction function α, the network $\mathcal{G}$ with N nodes becomes a network $\mathcal{G}'$ with $N' = (R+s)$ nodes; in fact, each set $\vec{\mathbf{u}}^{(k)}$ is mapped onto a single node ξ_k, whereas the nodes corresponding to $\vec{\mathbf{t}}$ remain the same. Let us notice that the subdivision of $\mathcal{G}$ into the $\mathcal{G}_k$'s was done by separating sets of cliques, not set of nodes. Then, each new node ξ_k still forms (possibly collapsed) cliques with the nodes in $\vec{\mathbf{t}}$. We may think that ξ_k and the nodes $\vec{\mathbf{t}}$ forms a unique clique which contributes to the abstract probability distribution $Q(\xi_1, ..., \xi_R, \vec{\mathbf{t}})$ by a single potential function $\Phi'_k(\xi_k, \vec{\mathbf{t}})$; then:

$$Q(\xi_1, ..., \xi_R, \vec{\mathbf{t}}) = \frac{1}{Z'}\prod_{k=1}^{R}\Phi'_k(\xi_k, \vec{\mathbf{t}}) \tag{14}$$

Let us now compute the marginal probability distribution of a generic ξ_k; from (14) we obtain:

$$\begin{aligned}Q(\xi_k) \quad &= \quad \frac{1}{Z'}\sum_{\vec{\mathbf{t}}\in\mathcal{X}_t}\sum_{\xi_j\ne\xi_k}\prod_{j=1}^{R}\Phi'_k(\xi_j,\vec{\mathbf{t}})\\ &= \quad \frac{1}{Z'}\sum_{\vec{\mathbf{t}}\in\mathcal{X}_t}\left(\Phi'_k(\xi_k,\vec{\mathbf{t}})\prod_{j=1,j\ne k}^{R}\Psi'_j(\vec{\mathbf{t}})\right)\end{aligned}$$

We may wonder whether there exists a definition of the $\Phi'_k(\xi_k, \vec{\mathbf{t}})$ $(1 \leq k \leq R)$ that makes $Q(\xi_k)$ (distribution of ξ_k computed in the abstract network) equal to $P(\xi_k)$ (distribution of ξ_k computed in the ground network). Using expression (10), we obtain:

$$P(\xi_k) = \frac{1}{Z} \sum_{\vec{\mathbf{t}} \in \mathcal{X}_t} \prod_{j=1, j \neq k}^{R} \Psi_j(\vec{\mathbf{t}}) \sum_{\vec{\mathbf{u}}^{(k)} \in D_{\xi_k}^{(k)}} \Phi_k(\vec{\mathbf{u}}^{(k)}, \vec{\mathbf{t}})$$

where $D_{\xi_k}^{(k)} = D_1^{(k)}$ if $\xi_k = 1$ and $D_{\xi_k}^{(k)} = D_0^{(k)}$ if $\xi_k = 0$.

By comparing expressions $P(\xi_k)$ and $Q(\xi_k)$ we see that $P(\xi_k) = Q(\xi_k)$ if we define:

$$\Phi'_k(\xi_k, \vec{\mathbf{t}}) = \sum_{\vec{\mathbf{u}}^{(k)} \in D_{\xi_k}^{(k)}} \Phi_k(\vec{\mathbf{u}}^{(k)}, \vec{\mathbf{t}}) \tag{15}$$

By summing equation (15) for $\xi_k = 0$ and $\xi_k = 1$, we obtain, by definition, that:

$$\Psi'_k(\vec{\mathbf{t}}) = \Psi_k(\vec{\mathbf{t}}) \quad \rightarrow \quad Z = Z' \tag{16}$$

We may notice that equality (16) allows Z' to be computed exactly on the abstract network, with a reduced complexity. The partitioning process may be repeated, obtaining thus multiple layers of abstraction. Moreover, the result (16) is independent from the specific abstraction function employed. Then, the selection of α, whose definition is domain-dependent, provides a degree of freedom to be exploited in order to preserve additional properties required by the task at hand.

As mentioned at the beginning, one of the goal of a Markov network is to allow the probability distribution of a query q to be computed in presence of an evidence e. For the sake of simplicity, let us assume that $q \equiv x_j$ (q is a single variable), and that the evidence $e \equiv (x_i = 1)$ is the assignment of the value *true* to another single variable x_i. We want to compute:

$$P(q|e) = \frac{P(q,e)}{P(e)} = \frac{P(x_j, x_i = 1)}{P(x_i = 1)} \tag{17}$$

Usually, it is too expensive to compute (17) directly, because it involves sums over possibly large set of worlds; then an approximate value is estimated through Gibbs sampling, which consists of sampling each node's truth value given its Markov blanket (Richardson and Domingos 2006). Using the abstract network $\mathcal{G}'$, either an exact value can be computed with reduced complexity, or an approximate estimation can be made, depending on the positions of the nodes x_j and x_i in $\mathcal{G}'$. There are three cases:

- The query q and the evidence e belong both to the same vector $\vec{\mathbf{u}}^{(k)}$ or to $\vec{\mathbf{t}}$
- The query q and the evidence e belong one to some $\mathcal{G}_k$ and one to $\vec{\mathbf{t}}$
- The query q and the evidence e belong to two different subgraphs $\mathcal{G}_k$ and $\mathcal{G}_\ell$.

The case in which both q and e belong to $\vec{\mathbf{t}} = (x_j, x_i, x_{i_1}, ..., x_{i_s})$ is the most favorable, because all the subgraphs $\mathcal{G}_k$ $(1 \leq k \leq R)$ can be abstracted to single nodes ξ_k. In this case we only need to compute the distribution $Q(\vec{\mathbf{t}})$ by marginalizing over the R variables ξ_k. The case in which both q and e belong to $\vec{\mathbf{u}}^{(k)} = (x_j, x_i, x_{i_1}, ..., x_{i_s})$ is analogous, but in this case we may only abstract $(R-1)$ subgraphs. In fact, subgraph $\mathcal{G}_k$ cannot be abstracted, because, in this case, we would only obtain from the abstract graph the probability $Q(\xi_k) = P(\xi_k)$, which is an agglomerated probability over the set of variable in $\vec{\mathbf{u}}^{(k)}$. Here is one of the points where the definition of the abstraction function may intervene: it could be defined (if possible) to be invertible.

If the query q and the evidence e belong one to some $\mathcal{G}_k$ and one to $\vec{\mathbf{t}}$, we cannot abstract $\mathcal{G}_k$, and we need the joint probability distribution over $\vec{\mathbf{u}}^{(k)}$ and $\vec{\mathbf{t}}$. Finally, when the query q and the evidence e belong to two different subgraphs $\mathcal{G}_k$ and $\mathcal{G}_\ell$, we need to abstract neither $\mathcal{G}_k$ nor $\mathcal{G}_\ell$, and we need the conditional probabilities of the variables $\vec{\mathbf{u}}^{(k)}$ and of the variables $\vec{\mathbf{u}}^{(\ell)}$ with respect to $\vec{\mathbf{t}}$. In fact, knowing $\vec{\mathbf{t}}$ makes the $\vec{\mathbf{u}}^{(k)}$'s and the $\vec{\mathbf{u}}^{(\ell)}$'s conditionally independent.

If the evidence and/or the query consists of the conjunction of more than one nodes, more subgraphs cannot be abstracted, by reducing the effectiveness of the approach. In this case it would be maybe better to accept an approximate answer to the query, which can be obtained in two ways: one is by sampling the relevant $\vec{\mathbf{u}}^{(k)}$, assuming known the $\vec{\mathbf{t}}$ (an extension of the method of the Markov blanket), or by defining an approximate inversion of the abstraction function.

In order to find a subdivision of the graph, any algorithm that searches for clusters of nodes, which are internally densely connected and externally loosely connected, may be used. Also algorithms for the minimum cut can be used. Both types of algorithms need a modification: the output disjoint vertex sets must be extended (rendering them no more disjoint) in order to include complete cliques; in this way the variables $\vec{\mathbf{t}}$ are created.

Another way to proceed (which we did not have yet pursued) is to generate the partition of the graph by demand, after the query and the evidence have been provided. In this case, the best would be to include both the query and the evidence in the same $\mathcal{G}_k$. This is possible by considering the minimum set of cliques that includes both. If the query and/or the evidence includes many nodes, this subgraph may turn out to be quite large.

5 Logical Markov Networks

Let us consider now the case in which the Markov network is a grounding of a First Order Logic knowledge base. More specifically, let $\mathcal{KB} = \{F_1, ..., F_K\}$ be a knowledge base, containing K formulas, and let $\mathcal{P}$ be the set of predicate names, with cardinality $|\mathcal{P}| = S$, occurring in it. Let $\mathcal{P}_k(F_k)$ be the set of atomic predicates occurring in F_k, with cardinality $|\mathcal{P}_k(F_k)| = q_k$; let moreover $\mathcal{V}_k(F_k)$ be the set of variables occurring in F_k, with cardinality $|\mathcal{V}_k(F_k)| = m_k$, extracted from a set $VAR = \{X, Y, Z, ...\}$.

Finally, let $\mathcal{A} = \{a_1, ..., a_n\}$ be a set of constants, and $\mathcal{G}$ be the graph corresponding to the totally instantiated knowledge base with respect to $\mathcal{A}$.

Graph $\mathcal{G}$ has N nodes and M cliques. Each node ν_i in $\mathcal{G}$ corresponds to a ground atom, which can be associated to a Boolean variable x_i. The state of $\mathcal{G}$, *i.e.*, a *world*, is an assignment of truth values to all the x_i's. $\mathcal{X} = \{0,1\}^N$ is set of all possible worlds.

Due to the way it is generated from a Logical Markov network, a ground Markov network has two characteristics: its structure is determined by the rules in $\mathcal{KB}$, and the potential functions assume a particular form. Concerning the structure, a generic formula $F_k(X_1, ..., X_{m_k})$ generates, in $\mathcal{G}$, a set of cliques Γ_k $(1 \leq k \leq K)$:

$$\Gamma_k = \{\gamma_1^{(k)}, ...\gamma_{h_k}^{(k)}\} \quad \text{with} \quad h_k = n^{m_k} \quad \text{and} \quad \sum_{k=1}^{K} h_k = M$$

Each clique has q_k vertices. Each clique $\gamma_j^{(k)}$ corresponds to one specific grounding of the formula F_k and has associated to it a potential function:

$$\Phi_{k,j}(\vec{\mathbf{x}}^{(k,j)}) = \exp\left[w_1 f(\vec{\mathbf{x}}^{(k,j)})\right]$$

where $\vec{\mathbf{x}}^{(k,j)}$ contains the variables occurring in the clique, and $f(\vec{\mathbf{x}}^{(k,j)})$ is a Boolean feature that assume value 1, if the grounding is *true*, and 0 if it *false* (in a particular world). The joint probability distribution can be expressed as follows:

$$\begin{aligned} P(x_1, ..., x_N) &= \prod_{k=1}^{K} \prod_{j=1}^{h_k} \exp\left[w_1 f(\vec{\mathbf{x}}^{(k,j)})\right] \quad (18) \\ &= \prod_{k=1}^{K} \exp\left[w_1 n(\vec{\mathbf{x}}^{(k)})\right] \quad (19) \end{aligned}$$

where $n(\vec{\mathbf{x}}^{(k)})$ is the number of true groundings of formula F_k in the particular world considered.

In a ground logical Markov network there is a natural subdivision of the global graph into subgraphs, by considering each formula at a time. Formula F_k generates a set of cliques, all with the same structure, differing only by the constants that appear in the groundings. It is then quite natural to consider the set of these cliques a unique subgraph $\mathcal{G}_k$. Moreover, taken any two formulas F_k and F_j it can be determined by the formulas themselves which nodes they have in common (the set of variables $\vec{\mathbf{t}}$). For instance, the graph in Figure 1 corresponds to the grounding, with respect to the set of constants $\mathcal{A} = \{a, b, c\}$, of the knowledge base $\mathcal{KB} = \{(F_1, w_1), (F_2, w_2)\}$, with $F_1(X,Y) = Smoke(X) \wedge Friend(X,Y) \rightarrow Smoke(Y)$ and $F_2(X) = Smoke(X) \rightarrow Cancer(X)$. The values w_1 and w_2 are the weights associated to the rules. By considering F_1 and F_2, it is immediate to see that the predicate $S(x)$ is in common for each of its instantiations.

Then, if by subdividing the graph reported in Figure 1 as described in Section 4, we obtain the abstract graph $\mathcal{G}_a$ reported in Figure 2. The consideration done in Section 4 for a generic Markov network applies to this case also, provided that the new potential functions are defined as in (15).

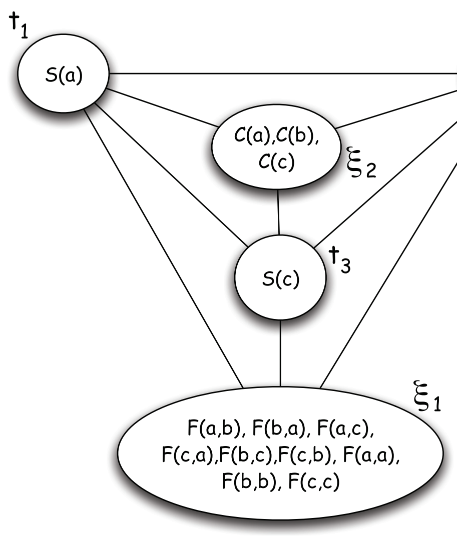
t1
t2
S(a)
S(b)
C(a),C(b), C(c)
ξ2
S(c)
t3
ξ1
F(a,b), F(b,a), F(a,c), F(c,a),F(b,c),F(c,b), F(a,a), F(b,b), F(c,c)

Figure 2: Abstract graph obtained from the ground one reported in Figure 1. The abstract graph has been obtained by considering each subgraph corresponding to the instantiation of every formula in the knowledge base. The nodes S(a), S(b) and S(c) are in common between the two subgraphs.

In $\mathcal{G}_a$ the Boolean variables ξ_1 and ξ_2 derived from two abstraction functions:

$$\begin{aligned} \xi_1 &= \alpha_1\big(F(a,a), F(a,b), F(a,c), F(b,a), F(b,b), \\ &\quad F(b,c), F(c,a), F(c,b), F(c,c)\big) \\ \xi_2 &= \alpha_2\big(C(a), C(b), C(c)\big) \end{aligned}$$

In this case the functions could be defined as the OR of the components, or their AND, or any other logical combination (or other type of function) suited for the task. Actually, by comparing the ground graph $\mathcal{G}$ and the abstract one $\mathcal{G}_a$, we may notice that the association of the variables in α_1 (α_2) with ξ_1 (ξ_2) corresponds to Hobbs' *domain abstraction*, based on the notion of *indistinguishability* (Hobbs 1985), with respect to the predicates $F(x,y)$ and $C(x)$. In this case, the variable ξ_1 is a representative for any of the component F(a,a), ... , F(c,c), in the sense that any of the latter assumes the truth value derived for ξ_1. The same holds for ξ_2. We may also notice that the graph $\mathcal{G}_a$ represents a kind of "lifted" graph, to perform lifted inference (Poole 2003).

View as a domain abstraction, the joint probability distribution in $\mathcal{G}_a$ can be expressed in a way analogous to (18). By giving an evidence $e \equiv (S(b) = 1)$ and a query $q \equiv$ F(a,b), we obtain, from $\mathcal{G}_a$:

$$P(F(a,b) = 1 | S(b) = 1) = 0.5$$

The true value, in the ground network, is indeed P(F(a,b) = 1) = 0.5. The equality between the abstract and the ground value of the probability of the query derives from the simmetry of the network. Experiments with generic topological structures show that the abstract value is, in general, only an approximation of the ground one. The estimation of the goodness of the approximation depends on the abstraction functions used.

6 Conclusions

In this paper we have described a preliminary investigation on the use of abstraction operators to reduce the complexity of inference in Markov network. We have shown that two kinds of abstraction (collapsing nodes and making constants indistinguishable) help reaching the goal. Other types of abstraction could be considered as well, for instance propositionalization. A related approach, applicable to generic CSPs, has been proposed by Lecoutre et al. (2000), who describe a method for abstracting CSPs. Even though not directly applicable to Markov networks, interesting links can be established. Other relations also exist with the paper by Shavlik and Natarajan (2009). In this paper the authors decribes a method for speeding up inference in ground Markov neworks by avoiding groundings that are certainly true/false.

The presented examples are only indicative of a methodology, and experiments on much larger networks and with a variety of abstraction operators are needed; this is the next step of this research. The methodology proposed is particularly well suited to logic Markov network, in which symmetry properties, due to the very way they are constructed, hold and can then be exploited.

References

Arenas, A.; Fernández, A.; and Gómez, S. 2008. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics* 10:053039.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Bulitko, V.; Sturtevant, N.; Lu, J.; and Yau, T. 2007. Graph abstraction in real-time heuristic search. *Journal of Artificial Intelligence Research* 30:51 – 100.

Clauset, A.; Moore, C.; and Newman, M. E. J. 2008. Hierarchical structure and the prediction of missing links in networks. *Nature* 453:98–101.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In *IJCAI*, 1319–1325.

Epstein, S. L., and Li, X. 2009. Cluster graphs as abstractions for constraint satisfaction problems. In 58-65., ed., *Proc. SARA-09*.

Harry, D., and Lindquist, D. 2004. Graph abstraction through centrality erosion and k-clique minimization.

Hobbs, J. R. 1985. Granularity. In 432-435., ed., *Proc. IJCAI-85*.

Holte, R. C.; Mkadmi, T.; Zimmer, R. M.; and MacDonald, A. J. 1996. Speeding up problem solving by abstraction: a graph oriented approach. *Artificial Intelligence* 85:321–361.

Kschischang, F. R.; Frey, B. J.; and Loeliger, H.-A. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47(2):498–519.

Milch, B., and Russell, S. J. 2006. General-purpose mcmc inference over relational structures. In *UAI*.

Poole, D. 2003. First-order probabilistic inference. In *IJCAI*, 985–991.

Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*.

Poon, H.; Domingos, P.; and Sumner, M. 2008. A general method for reducing the complexity of relational inference and its application to mcmc. In *AAAI*, 1075–1080.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Riedel, S. 2008. Improving the accuracy and efficiency of map inference for markov logic. In *UAI*, 468–475.

Saitta, L.; Henegar, C.; and Zucker, J.-D. 2009. Abstracting complex interaction networks. In 190-193., ed., *Proc. SARA-09*.

Shavlik, J., and Natarajan, S. 2009. Speeding up inference in markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *Proc. IJCAI 2009*, 1951–1956.

Singla, P., and Domingos, P. 2006. Memory-efficient inference in relational domains. In *AAAI*.

Singla, P., and Domingos, P. 2008. Lifted first-order belief propagation. In *AAAI*, 1094–1099.

Zhang, S.; Ning, X.; and Zhang, X. 2007. Graph kernels, hierarchical clustering, and network community structure: experiments and comparative analysis. *Eur. Phys. J. B* 57:67–74.