Lifted Aggregation in Directed First-order Probabilistic Models

Jacek Kisyński and **David Poole**
Department of Computer Science
University of British Columbia
{kisynski,poole}@cs.ubc.ca

Abstract

As exact inference for first-order probabilistic graphical models at the propositional level can be formidably expensive, there is an ongoing effort to design efficient lifted inference algorithms for such models. This paper discusses directed first-order models that require an aggregation operator when a parent random variable is parameterized by logical variables that are not present in a child random variable. We introduce a new data structure, aggregation parfactors, to describe aggregation in directed first-order models. We show how to extend Milch *et al.*'s C-FOVE algorithm to perform lifted inference in the presence of aggregation parfactors. We also show that there are cases where the polynomial time complexity (in the domain size of logical variables) of the C-FOVE algorithm can be reduced to logarithmic time complexity using aggregation parfactors.

1 Introduction

Probabilistic graphical models, such as belief networks, are a popular tool for representing dependencies between random variables. However, such standard representations are propositional, hence are not well suited for describing relations between individuals or quantifying over sets of individuals. First-order logic has the capacity for representing relations and quantification of variables, but it does not treat uncertainty. Representations that mix graphical models and first-order logic (*first-order probabilistic models*) were proposed more than fifteen years ago [Horsch and Poole, 1990; Breese, 1992]. In these models, random variables are parameterized by logical variables that are typed with populations of individuals.

Among the appeals of the first-order probabilistic models is that one should be able to fully specify a model, that is, its structure and the accompanying probability distributions, before knowing the individuals in the modeled domain. This means in particular that, even though we might not know the sizes of the populations, we still should be able to specify the model. To make this possible, the length of a specification of a first-order probabilistic model must be independent of the sizes of the populations in the model.

Although many first-order probabilistic languages have since emerged [Getoor and Taskar, 2007; De Raedt *et al.*, 2008], the most common exact inference technique has been based on dynamical propositionalization (grounding) of the portion of the first-order model that is relevant to the query, followed by probabilistic inference performed at the propositional level. The problem with these propositional representations is that they may be extremely large, rendering inference intractable even for very simple first-order models. Other approaches exploit redundant computation [Koller and Pfeffer, 1997; Pfeffer and Koller, 2000], or compile the problem into an arithmetic circuit [Chavira *et al.*, 2006].

The idea of *lifted inference* is to carry out as much inference as possible without propositionalizing. The correctness of this approach is judged by having the same result as if we had first grounded and then carried out standard inference. An exact lifted inference procedure for first-order probabilistic, directed models was proposed by Poole [2003]. One of the obstacles in avoiding propositionalization occurs when a first-order model contains adjacent parameterized random variables that have different parameterizations. This problem was investigated by de Salvo Braz *et al.* [2007]. Further work resulted in the C-FOVE algorithm [Milch *et al.*, 2008], which is currently the state of the art in exact lifted inference.

While Poole considered directed models, the later work by de Salvo Braz *et al.* and Milch *et al.* focused on undirected models. Their results can be used for directed models, which have the advantage of allowing pruning of the part of the model that is irrelevant to the query. Also, conditional probability distributions in directed models can be interpreted and learned locally, which is important for models that are specified by people or need to be understood by people.

One aspect that arises in directed models is the need for aggregation that occurs when a parent random variable is parameterized by logical variables that are not present in a child random variable. Currently available first-order inference algorithms do not allow a description of aggregation in first-order models that is independent of the sizes of the populations. In this paper we introduce a new data structure, *aggregation parfactors*, describe how to use it to represent aggregation in first-order models, and show how to perform lifted inference in its presence. Experiments presented in Section 5 show that aggregation parfactors can lead to gains in efficiency.

2 Preliminaries

Like previous work on lifted probabilistic inference, this paper is not tied to any particular first-order probabilistic language. We reason at the level of data structures and assume that various first-order languages (or their subsets) will compile to these data structures. First-order probabilistic languages share a concept of a *parameterized random variable*. We introduce related terms in Section 2.1. The idea of parameterized random variables is similar to the notion of plates [Buntine, 1994]; we use plates notation in our figures. In Section 2.2 we discuss aggregation in directed first-order probabilistic models.

2.1 Parameterized Random Variables

If $\mathcal{S}$ is a set, we denote by $|\mathcal{S}|$ the size of the set $\mathcal{S}$.

A *population* is a set of *individuals*. A population corresponds to a domain in logic. For example, a population may be a set of all soccer players involved in a soccer game, where *rossi* is one of the individuals and the population size is 22.

A *parameter* corresponds to a logical variable and is typed with a population. For example, parameter *Player* may be typed with the population of all players involved in a soccer game. Given parameter A, we denote its population by $\mathcal{D}(A)$. Given a set of constraints $\mathcal{C}$, we denote a set of individuals from $\mathcal{D}(A)$ that satisfy constraints in $\mathcal{C}$ by $\mathcal{D}(A) : \mathcal{C}$.

A *substitution* is of the form $\{X_1/t_1 \ldots, X_k/t_k\}$, where the X_i are distinct parameters, and each *term* t_i is a parameter typed with a population or a constant denoting an individual from a population. A *ground substitution* is a substitution, where each t_i is a constant.

A *parameterized random variable* is of the form $f(t_1, \ldots, t_k)$, where f is a functor (either a function symbol or a predicate symbol) and t_i are terms. We denote a set of parameters of the parameterized random variable $f(t_1, \ldots, t_k)$ by $\mathbb{P}(f(t_1, \ldots, t_k))$. Each functor has a set of values called the *range* of the functor. We denote the range of the functor f by $range(f)$. Examples of parameterized random variables are $inj(Player)$ and $inj(rossi)$. We have $\mathbb{P}(inj(Player)) = \{Player\}$ and $\mathbb{P}(inj(rossi)) = \emptyset$.

A parameterized random variable $f(t_1, \ldots, t_k)$ represents a set of random variables, one for each possible ground substitution to all of its parameters. For example, if *Player* is typed with a population consisting of all 22 individuals playing the game, then $inj(Player)$ represents 22 random variables: $inj(rossi)$, $inj(panucci)$, ..., $inj(desailly)$ corresponding to ground substitutions $\{Player/rossi\}$, $\{Player/panucci\}$, ..., $\{Player/desailly\}$, respectively. The range of the functor of the parameterized random variable is the domain of random variables represented by the parameterized random variable.

Let $\mathbf{v}$ denote an *assignment of values* to random variables; $\mathbf{v}$ is a function that takes a random variable and returns its value. We extend $\mathbf{v}$ to also work on parameterized random variables, where we assume that free parameters are universally quantified. For example, if $\mathbf{v}(inj(Player)) = true$, then each of the random variables represented by $inj(Player)$, namely $inj(rossi)$, $inj(panucci)$, ..., $inj(desailly)$, is assigned the value *true* by $\mathbf{v}$.

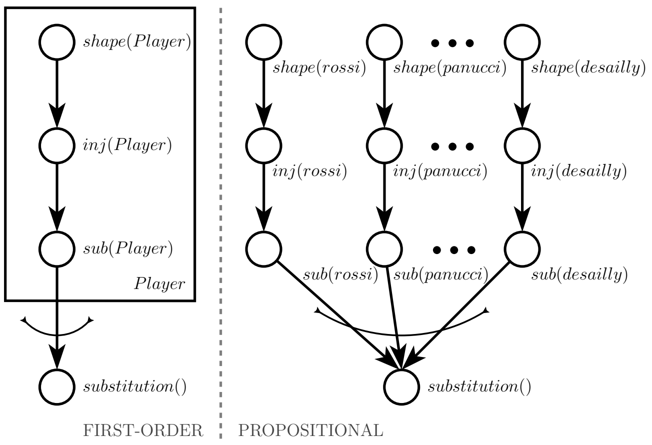

Figure 1: A first-order model from Example 1 and its equivalent belief network. Aggregation is denoted by curved arcs.

2.2 Aggregation in Directed First-order Probabilistic Models

First-order probabilistic models describe probabilistic dependencies between parameterized random variables. A *grounding* of a first-order probabilistic model is a propositional probabilistic model obtained by replacing each parameterized random variable with the random variables it represents and replicating appropriate probability distributions.

Example 1. Consider the directed first-order probabilistic model and its grounding presented in Figure 1. The model is meant to represent that whether a player playing a soccer game is substituted or not during a single soccer game depends on whether he gets injured. The probability of an injury in turn depends on a physical condition of the player. The model has four nodes: a parameterized random variable $shape(Player)$ with range $\{fit, unfit\}$, a parameterized random variable $inj(Player)$ with range $\{true, false\}$, a parameterized random variable $sub(Player)$ with range $\{true, false\}$, and a variable $substitution()$ with range $\{true, false\}$ that is *true* if a player was substituted during the game and *false* otherwise. We have $\mathcal{D}(Player) = \{rossi, panucci, \ldots, desailly\}$ and $|\mathcal{D}(Player)| = 22$.

A parameterized random variable $shape(Player)$ represents the 22 random variables in the corresponding propositional model, as do variables $inj(Player)$ and $sub(Player)$. Therefore, in the propositional model, the number of parent nodes influencing the node $substitution()$ is equal to 22. Their common effect aggregates in the child variable. In the discussed model we use the logical OR as an aggregation operator to describe the (deterministic) conditional probability distribution $\mathcal{P}(substitution()|sub(Player))$. Note that $substitution()$ is a noisy-OR of $inj(Player)$.

In a directed first-order model, when a child variable has a parent variable with extra parameters, in the grounding the child variable has an unbounded number of parents. We need some aggregation operator to describe how the child variable depends on the parent variable. Following Zhang and Poole [1996], we assume that the range of the parent variable is a subset of the range of the child variable, and use a commutative and associative deterministic binary operator over the range of the child variable as an aggregation operator $\otimes$.

Given probabilistic input to the parent variable, we can construct any causal independence model covered by the definition of causal independence from Zhang and Poole [1996], which in turn covers common causal independence models such as noisy-OR [Pearl, 1986] and noisy-MAX [Díez, 1993] as special cases. In other words, this allows any causal independence model to act as underlying mechanism for aggregation in directed first-order models. For other types of aggregation in first-order models, see Jaeger [2002].

In this paper we require that the directed first-order probabilistic models satisfy the following conditions:

(1) for each parameterized random variable, its parent has at most one extra parameter
(2) if a parameterized random variable $c(\dots)$ has a parent $p(\dots,A,\dots)$ with an extra parameter A, then:
    (a) $p(\dots,A,\dots)$ is the only parent of $c(\dots)$
    (b) the range of p is a subset of the range of c
    (c) $c(\dots)$ is a deterministic function of the parent: $c(\dots) = p(\dots,a_1,\dots) \otimes \dots \otimes p(\dots,a_n,\dots) = \bigotimes_{a\in\mathcal{D}(A)} p(\dots,a,\dots)$, where $\otimes$ is a commutative and associative deterministic binary operator over the range of c.

At first the above conditions seem to be very restrictive, but they in fact are not. There is no need to define the aggregation over more than one parameter due to the associativity and commutativity of the $\otimes$ operator. We can obtain more complicated distributions by introducing auxiliary variables and combining multiple aggregations.

Example 2. Consider a parent variable $p(A,B,C)$ and a child variable $c(C)$. We can describe a $\otimes$-based aggregation over A and B, $c(C) = \bigotimes_{(a,b)\in\mathcal{D}(A)\times\mathcal{D}(B)} p(A,B,C)$ using an auxiliary parameterized random variable $c'(B,C)$ such that c' has the same range as c. Let $c'(B,C) = \bigotimes_{a\in\mathcal{D}(A)} p(A,B,C)$, then $c(C) = \bigotimes_{b\in\mathcal{D}(B)} c'(B,C)$.

Similarly, with the use of auxiliary nodes, we can construct a distribution that combines an aggregation with influence from other parent nodes or even combines multiple aggregations generated with different operators.

In the rest of the paper, we assume that the discussed models satisfy conditions (1) and (2), for ease of presentation and with no loss of generality.

3 Existing Algorithm

In this section, we introduce *counting formulas* [Milch *et al.*, 2008] and *parfactors* [Poole, 2003] and give an overview of the C-FOVE algorithm [Milch *et al.*, 2008].

3.1 Counting Formulas

A *counting formula* is $\#_{A:\mathcal{C}}[f(\dots,A,\dots)]$, where A is a parameter that is *bound* by the # sign, $\mathcal{C}$ is a set of inequality constraints involving A and $f(\dots,A,\dots)$ is a parameterized random variable.

The value of $\#_{A:\mathcal{C}}[f(\dots,A,\dots)]$, given an assignment of values to random variables $\mathbf{v}$, is the *histogram* h that maps the range of f to natural numbers such that

$$h(x) = |\{a \in (\mathcal{D}(A):\mathcal{C}) : \mathbf{v}(f(\dots,a,\dots)) = x\}|.$$

The range of such a counting formula is the set of histograms having a bucket for each element x in the range of f with entries adding up to $|\mathcal{D}(A):\mathcal{C}|$. The number of such histograms is $\binom{|\mathcal{D}(A):\mathcal{C}|+|range(f)|-1}{|range(f)|-1}$, which for small values of $|range(f)|$ is $\mathcal{O}(|\mathcal{D}(A):\mathcal{C}|^{|range(f)|-1})$. Thus, any extensional representation of a function on a counting formula $\#_{A:\mathcal{C}}[f(\dots,A,\dots)]$ requires amount of space at least linear in $|\mathcal{D}(A):\mathcal{C}|$.

Counting formulas allow us to exploit interchangeability within factors. They were inspired by work on *cardinality potentials* [Gupta *et al.*, 2007] and *counting elimination* [de Salvo Braz *et al.*, 2007]. They are a new form of parameterized random variables. Unless otherwise stated, by parameterized random variables we understand both forms: the standard, defined in Section 2.1, and counting formulas.

3.2 Parametric Factors

A *factor* on a set of random variables represents a function that, given an assignment of a value to each random variable from the set, returns a real number. Factors are used in the variable elimination algorithm [Zhang and Poole, 1994] to store initial conditional probabilities and intermediate results of computation during probabilistic inference in graphical models. Operations on factors include multiplication of factors and summing out random variables from a factor.

Let $\mathbf{v}$ be an assignment of values to random variables and let $\mathcal{F}$ be a factor on a set of random variables $\mathcal{S}$. We extend $\mathbf{v}$ to factors and denote by $\mathbf{v}(\mathcal{F})$ the value of the factor $\mathcal{F}$ given $\mathbf{v}$. If $\mathbf{v}$ does not assign values to all of the variables in $\mathcal{S}$, then $\mathbf{v}(\mathcal{F})$ denotes a factor on other variables.

A *parametric factor* or *parfactor* is a triple $\langle\mathcal{C},\mathcal{V},\mathcal{F}\rangle$ where $\mathcal{C}$ is a set of inequality constraints on parameters (between a parameter and a constant or between two parameters), $\mathcal{V}$ is a set of parameterized random variables and $\mathcal{F}$ is a factor from the Cartesian product of ranges of parameterized random variables in $\mathcal{V}$ to the reals.

A parfactor $\langle\mathcal{C},\mathcal{V},\mathcal{F}\rangle$ represents a set of factors, one for each ground substitution $\mathcal{G}$ to all free parameters in $\mathcal{V}$ that satisfies constraints in $\mathcal{C}$. Each such factor $\mathcal{F}_\mathcal{G}$ is a factor on the set of random variables obtained by applying a substitution $\mathcal{G}$. Given an assignment $\mathbf{v}$ to the random variables represented by $\mathcal{V}$, we define $\mathbf{v}(\mathcal{F}_\mathcal{G}) = \mathbf{v}(\mathcal{F})$.

We use parfactors to represent probability distributions for parameterized random variables in first-order models and intermediate computation results during lifted inference.

Normal-Form Constraints

Let X be a parameter in $\mathcal{V}$ from a parfactor $\langle\mathcal{C},\mathcal{V},\mathcal{F}_\mathcal{F}\rangle$. In general, the size of the set $\mathcal{D}(X):\mathcal{C}$ depends on other parameters in $\mathcal{V}$ (see discussions on *uniform solution counting partitions* in de Salvo Braz *et al.* [2007] and *normal form constraints* in Milch *et al.* [2008]).

Milch *et al.* [2008] introduced a special class of sets of inequality constraints. Let $\mathcal{C}$ be a set of inequality constraints on parameters and X be a parameter. We denote by $\mathcal{E}_X^{\mathcal{C}}$ the set of terms t such that $(X \neq t) \in \mathcal{C}$. Set $\mathcal{C}$ is in *normal form* if for each inequality $(X \neq Y) \in \mathcal{C}$, where X and Y are parameters, $\mathcal{E}_X^{\mathcal{C}} \setminus \{Y\} = \mathcal{E}_Y^{\mathcal{C}} \setminus \{X\}$.

Consider a parfactor $\langle \mathcal{C}, \mathcal{V}, \mathcal{F}_{\mathcal{F}} \rangle$, where $\mathcal{C}$ is in normal form. For all parameters X in $\mathcal{V}$, $|\mathcal{D}(X) : \mathcal{C}| = |\mathcal{D}(X)| - |\mathcal{E}_X^{\mathcal{C}}|$.

We require that for a parfactor $\langle \mathcal{C}, \mathcal{V}, \mathcal{F}_{\mathcal{F}} \rangle$ involving counting formulas, the union of $\mathcal{C}$ and the constraints in all the counting formulas in $\mathcal{V}$ is in normal form. Other parfactor do not need to be in normal form.

3.3 C-FOVE

Let Φ be a set of parfactors. Let $\mathcal{J}(\Phi)$ denote a factor equal to the product of all factors represented by elements of Φ. Let $\mathbf{U}$ be the set of all random variables represented by parameterized random variables present in parfactors in Φ. Let $\mathbf{Q}$ be a subset of $\mathbf{U}$. The *marginal of* $\mathcal{J}(\Phi)$ *on* $\mathbf{Q}$, denoted $\mathcal{J}_{\mathbf{Q}}(\Phi)$, is defined as $\mathcal{J}_{\mathbf{Q}}(\Phi) = \sum_{\mathbf{U} \setminus \mathbf{Q}} \mathcal{J}(\Phi)$.

Given Φ and $\mathbf{Q}$, the C-FOVE algorithm computes the marginal $\mathcal{J}_{\mathbf{Q}}(\Phi)$ by summing out random variables from $\mathbf{U} \setminus \mathbf{Q}$, where possible in a lifted manner. Evidence can be handled by adding to Φ additional parfactors on observed random variables.

As lifted summing out is only possible under certain conditions, the C-FOVE algorithm uses elimination enabling operations, such as applying substitutions to parfactors and multiplication. Below we show when and how these operations can be applied to aggregation parfactors. We refer the reader to Milch *et al.* [2008] for more details on C-FOVE.

4 Incorporating aggregation in C-FOVE

In Section 4.1, we show how to represent aggregation in first-order models using a simple form of *aggregation parfactors*. In Section 4.2, we show how these aggregation parfactors can be converted to parfactors that in turn can be used during inference with C-FOVE. In Section 4.3, we describe when and how reasoning directly in terms of these aggregation parfactors can achieve improved efficiency. In Section 4.4 we outline how a generalized version of aggregation parfactors increases the number of cases for which efficiency is improved.

4.1 Aggregation Parfactors

Example 3. Consider the model presented in Figure 1. We cannot represent the conditional probability distribution $\mathcal{P}(substitution()|sub(Player))$ with a parfactor $\langle \emptyset, \{sub(Player), substitution()\}, \mathcal{F} \rangle$ as even simple noisy-OR cannot be represented as a product. A parfactor $\langle \emptyset, \{sub(rossi), \ldots, sub(desailly), substitution()\}, \mathcal{F} \rangle$ is not an adequate input representation of this distribution because its size would depend on $|\mathcal{D}(Player)|$. The same applies to a parfactor $\langle \emptyset, \{\#_{Player:\emptyset}[sub(Player)], substitution()\}, \mathcal{F} \rangle$ as the size of the range of $\#_{Player:\emptyset}[sub(Player)]$ depends on $|\mathcal{D}(Player)|$.

Definition 1. An *aggregation parfactor* is a hex-tuple $\langle \mathcal{C}, p(\ldots,A,\ldots), c(\ldots), \mathcal{F}_p, \otimes, \mathcal{C}_A \rangle$, where

- $p(\ldots,A,\ldots)$ and $c(\ldots)$ are parameterized random variables
- the range of p is a subset of the range of c
- A is the only parameter in $p(\ldots,A,\ldots)$ that is not in $c(\ldots)$
- $\mathcal{C}$ is a set of inequality constraints not involving A
- $\mathcal{F}_p$ is a factor from the range of p to real numbers
- $\otimes$ is a commutative and associative deterministic binary operator over the range of c
- $\mathcal{C}_A$ is a set of inequality constraints involving A.

An aggregation parfactor $\langle \mathcal{C}, p(\ldots,A,\ldots), c(\ldots), \mathcal{F}_p, \otimes, \mathcal{C}_A \rangle$ represents a set of factors, one for each ground substitution $\mathcal{G}$ to parameters $\mathbb{P}(p(\ldots,A,\ldots)) \cup \mathbb{P}(\ldots)) \setminus \{A\}$ that satisfies constraints in $\mathcal{C}$. Each factor $\mathcal{F}_{\mathcal{G}}$ is a mapping from the Cartesian product $\left(\times_{a \in \mathcal{D}(A):\mathcal{C}_A} range(p)\right) \times range(c)$ to the reals, which, given an assignment of values to random variables $\mathbf{v}$, is defined as follows:

$$\mathbf{v}(\mathcal{F}_{\mathcal{G}}) = \begin{cases} \prod_{a \in \mathcal{D}(A):\mathcal{C}_A} \mathcal{F}_p(\mathbf{v}(p(\ldots,A,\ldots))), \\ \quad \text{if } \bigotimes_{a \in \mathcal{D}(A):\mathcal{C}_A} \mathbf{v}(p(\ldots,a,\ldots)) = \mathbf{v}(c(\ldots)); \\ 0, \text{otherwise.} \end{cases}$$

It is important to notice that $\mathcal{D}(A) : \mathcal{C}_A$ might vary for different ground substitutions $\mathcal{G}$ if the set $\mathcal{C} \cup \mathcal{C}_A$ is not in normal form (see Section 3.2). The space required to represent an aggregation parfactor does not depend on the size of the set $\mathcal{D}(A) : \mathcal{C}_A$. It is also at most quadratic in the size of $range(c)$, as the operator $\otimes$ can be represented as a factor from $range(c) \times range(c)$ to $range(c)$.

When an aggregation parfactor $\langle \mathcal{C}, p(\ldots,A,\ldots), c(\ldots), \mathcal{F}_p, \otimes, \mathcal{C}_A \rangle$ is used to describe aggregation in a first-order model, the factor $\mathcal{F}_p$ will be a constant function with the value 1. An aggregation parfactor created during inference may have a non-trivial $\mathcal{F}_p$ component (see Section 4.3).

Example 4. Consider the first-order model and its grounding presented in Figure 1. We can represent the conditional probability distribution $\mathcal{P}(substitution()|sub(Player))$ with an aggregation parfactor $\langle \emptyset, sub(Player), substitution(), \mathcal{F}_{sub}, \text{OR}, \emptyset \rangle$, where $\mathcal{F}_{sub}$ is a constant function with the value 1. The size of the representation does not depend on the population size of the parameter *Player*.

In the rest of the paper, Φ denotes a set of parfactors and aggregation parfactors. The notation introduced in Section 3.3 remains valid under the new meaning of Φ.

4.2 Conversion to Parfactors

Conversion using counting formulas

Consider an aggregation parfactor $\langle \mathcal{C}, p(\ldots,A,\ldots), c(\ldots), \mathcal{F}_p, \otimes, \mathcal{C}_A \rangle$. Since $\otimes$ is an associative and commutative operator, given an assignment of values to random variables $\mathbf{v}$, it does not matter which of the variables $p(\ldots,a,\ldots)$, $a \in \mathcal{D}(A):\mathcal{C}_A$ are assigned each value from $range(p)$, but only how many of them are assigned each value. This property was a motivation for the *counting elimination* algorithm [de Salvo Braz *et al.*, 2007] and counting formulas [Milch *et al.*, 2008], and allows us to convert aggregation parfactors to a product of two parfactors, where one of the parfactors involves a counting formula.

Proposition 1. Let $g_A = \langle \mathcal{C}, p(\ldots,A,\ldots), c(\ldots), \mathcal{F}_p, \otimes, \mathcal{C}_A \rangle$ be an aggregation parfactor from Φ such that set $\mathcal{C} \cup \mathcal{C}_A$ is in normal form. Let $\mathcal{F}_\#$ be a factor from the Cartesian product $range(\#_{A:\mathcal{C}_A}[p(\dots,A,\dots)]) \times range(c)$ to $\{0,1\}$. Given an assignment of values to random variables $\mathbf{v}$, the function is defined as follows:

$$\mathcal{F}_\#(h(),\mathbf{v}(c(\dots))) = \begin{cases} 1, \text{ if } \bigotimes_{x\in range(p)} \bigotimes_{i=1}^{h(x)} x \\ \qquad = \mathbf{v}(c(\dots)); \\ 0, \text{ otherwise,} \end{cases}$$

where $h()$ is a histogram from $range(\#_{A:\mathcal{C}_A}[p(\dots,A,\dots)])$. Then $\mathcal{J}(\Phi) = \mathcal{J}(\Phi \setminus \{g_A\} \cup \{\langle \mathcal{C}\cup\mathcal{C}_A, \{p(\dots,A,\dots)\}, \mathcal{F}_p\rangle, \langle \mathcal{C}, \{\#_{A:\mathcal{C}_A}[p(\dots,A,\dots)], c(\dots)\}, \mathcal{F}_\#\rangle\})$.

If the set $\mathcal{C}\cup\mathcal{C}_A$ is not in normal form we will need to use splitting operation described in Section 4.3 to convert the aggregation parfactor to a set of aggregation parfactors with constraint sets in normal form.

Conversion for MAX **and** MIN **operators**

If in an aggregation parfactor $\otimes$ is the MAX operator (which includes the OR operator as a special case), we can use a factorization presented by Díez and Galán [2003] to convert the aggregation parfactor to parfactors without counting formulas. The factorization is an example of the *tensor rank-one decomposition* of a conditional probability distribution [Savicky and Vomlel, 2007].

Proposition 2. Let $g_A = \langle \mathcal{C}, p(\dots,A,\dots), c(\dots), \mathcal{F}_p, \text{MAX}, \mathcal{C}_A\rangle$ be an aggregation parfactor from Φ, where MAX operator is induced by a total ordering $\prec$ of $range(c)$. Let $\mathbf{s}()$ be a successor function induced by $\prec$. Let $c'(\dots)$ be an auxiliary parameterized random variable with the same parameterization and the same range as c. Let $\mathcal{F}_c$ be a factor from the Cartesian product $range(p) \times range(c)$ to real numbers that, given an assignment of values to random variables $\mathbf{v}$, is defined as follows:

$$\mathcal{F}_c(\mathbf{v}(p(\dots,A,\dots)),\mathbf{v}(c'(\dots))) =$$
$$\begin{cases} \mathcal{F}_p(\mathbf{v}(p(\dots,A,\dots))), & \text{if } \mathbf{v}(p(\dots,A,\dots)) \preccurlyeq \mathbf{v}(c'(\dots)); \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathcal{F}_\Delta$ be a factor from the Cartesian product $range(p) \times range(c)$ to real numbers that, given $\mathbf{v}$, is defined as follows:

$$\mathcal{F}_\Delta(\mathbf{v}(c(\dots)),\mathbf{v}(c'(\dots))) = \begin{cases} 1, \text{ if } \mathbf{v}(c(\dots)) = \mathbf{v}(c'(\dots)); \\ -1, \text{ if } \mathbf{v}(c(\dots)) = \mathbf{s}(\mathbf{v}(c'(\dots))); \\ 0, \text{ otherwise.} \end{cases}$$

Then $\mathcal{J}(\Phi) = \mathcal{J}(\Phi \setminus \{g_A\} \cup \{\langle \mathcal{C}\cup\mathcal{C}_A, \{p(\dots,A,\dots), c'(\dots)\}, \mathcal{F}_c\rangle, \langle \mathcal{C}, \{c(\dots), c'(\dots)\}, \mathcal{F}_\Delta\rangle\})$.

An analogous proposition holds for the MIN operator. In both cases, as shown in Section 5, the above conversion is advantageous to the conversion described in Proposition 1, which uses counting formulas.

4.3 Operations on Aggregation Parfactors

In the previous section we showed that aggregation parfactors can be used during a modeling phase and then, during inference with the C-FOVE algorithm, once populations are known, aggregation parfactors can be translated to parfactors. Such a solution allows us to take advantage of the modeling properties of aggregation parfactors and C-FOVE inference capabilities. It is also possible to exploit aggregation parfactors during inference. In this section we describe operations on aggregation parfactors that can be added to the C-FOVE algorithm. These operation can delay or even avoid translation of aggregation parfactors to parfactors involving counting formulas. This in turn, as we will see in Section 5, can result in more efficient inference.

Splitting

The C-FOVE algorithm applies substitutions to parfactors to handle observations and queries and to enable the multiplication of parfactors. As this operation results in the creation of a residual parfactor, it is called splitting. Below we present how aggregation parfactors can be split on substitutions.

Proposition 3. Let $g_A = \langle \mathcal{C}, p(\dots,A,\dots), c(\dots), \mathcal{F}_p, \otimes, \mathcal{C}_A\rangle$ be an aggregation parfactor from Φ. Let $\{X/t\}$ be a substitution such that $(X \neq t) \notin \mathcal{C}$ and $X \in \mathbb{P}(p(\dots,A,\dots)) \setminus \{A\}$ and term t is a constant such that $t \in \mathcal{D}(X)$, or a parameter such that $t \in \mathbb{P}(p(\dots,A,\dots)) \setminus \{A\}$. Let $g_A[X/t]$ be a parfactor g_A with all occurrences of X replaced by term t. Then $\mathcal{J}(\Phi) = \mathcal{J}(\Phi \setminus \{g_A\} \cup \{g_A[X/t], \langle \mathcal{C}\cup\{X \neq t\}, p(\dots,A,\dots), c(\dots), \mathcal{F}_p, \otimes, \mathcal{C}_A\rangle\})$.

Proposition 3 allows us to split an aggregation parfactor on a substitution that does not involve the aggregation parameter. Below we show how to split on a substitution that involves the aggregation parameter A and a constant. Such an operation divides the individuals from $\mathcal{D}(A) : \mathcal{C}$ in two data structures: an aggregation parfactor and a standard parfactor. We have to make sure that after splitting $c(\dots)$ is still equal to a $\otimes$-based aggregation over the whole $\mathcal{D}(A) : \mathcal{C}$.

Proposition 4. Let $g_A = \langle \mathcal{C}, p(\dots,A,\dots), c(\dots), \mathcal{F}_p, \otimes, \mathcal{C}_A\rangle$ be an aggregation parfactor from Φ. Let $\{A/t\}$ be a substitution such that $(A \neq t) \notin \mathcal{C}_A$ and term t is a constant such that $t \in \mathcal{D}(A)$, or a parameter such that $t \in \mathbb{P}(p(\dots,A,\dots)) \setminus \{A\}$. Let $c'(\dots)$ be an auxiliary parameterized random variable with the same parameterization and range as $c(\dots)$. Let $\mathcal{C}_A[A/t]$ be a set of constraints $\mathcal{C}_A$ with all occurrences of A replaced by term t. Let $\mathcal{F}_1$ be a factor from the Cartesian product $range(p) \times range(c') \times range(c)$ to real numbers. Given an assignment of values to random variables $\mathbf{v}$, the function is defined as follows:

$$\mathcal{F}_1(\mathbf{v}(p(\dots,A,\dots)),\mathbf{v}(c'(\dots)),\mathbf{v}(c(\dots))) =$$
$$\begin{cases} \mathcal{F}_p(p(\dots,t,\dots)), \text{ if } \mathbf{v}(p(\dots,t,\dots)) \otimes \mathbf{v}(c'(\dots)) \\ \qquad\qquad = \mathbf{v}(c(\dots)) \\ 0, \text{ otherwise.} \end{cases}$$

Then $\mathcal{J}(\Phi) = \mathcal{J}(\Phi \setminus \{g_A\} \cup \{\langle \mathcal{C}, p(\dots,A,\dots), c'(\dots), \mathcal{F}_p, \otimes, \mathcal{C}_A \cup \{A \neq t\}\rangle, \langle \mathcal{C}\cup\mathcal{C}_A[A/t], \{p(\dots,t,\dots), c'(\dots), c(\dots)\}, \mathcal{F}_1\rangle\})$.

Splitting presented in Proposition 4 corresponds to the expansion of a counting formula in C-FOVE. The case where a substitution is of the form $\{X/A\}$ can be handled in a similar fashion as described in Proposition 4. If a substitution has more than one element, then we split recursively on its elements using the above propositions.

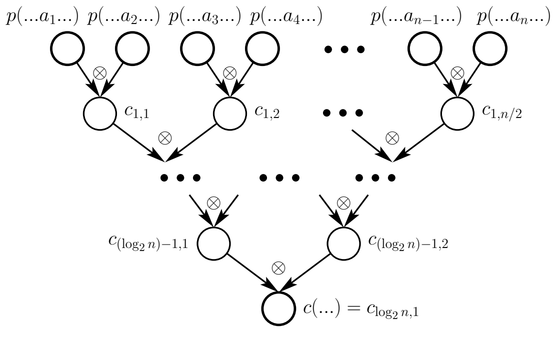

Figure 2: Decomposed aggregation.

Multiplication

The C-FOVE algorithm multiplies parfactors to enable elimination of parameterized random variables. An aggregation parfactor can be multiplied by a parfactor on $p(\ldots,A,\ldots)$.

Proposition 5. Let $g_A = \langle \mathcal{C},\ p(\ldots,A,\ldots),\ c(\ldots),\ \mathcal{F}_p,\ \otimes,\ \mathcal{C}_A\rangle$ be an aggregation parfactor from Φ and $g_1 = \langle \mathcal{C}_1,\ \{p(\ldots,A,\ldots)\},\ \mathcal{F}_1\rangle$ be a parfactor from Φ such that $\mathcal{C}_1 = \mathcal{C} \cup \mathcal{C}_A$. Let $g_2 = \langle \mathcal{C},\ p(\ldots,A,\ldots),\ c(\ldots),\ \mathcal{F}_p\mathcal{F}_1,\ \otimes,\ \mathcal{C}_A\rangle$. Then $\mathcal{J}(\Phi) = \mathcal{J}(\Phi \setminus \{g_A, g_1\} \cup \{g_2\})$.

We call g_2 the product of g_A and g_1.

Summing out

The C-FOVE algorithm sums out random variables to compute the marginal. Below we show how in some cases we can sum out $p(\ldots,A,\ldots)$ directly from an aggregation parfactor.

When $p(\ldots,A,\ldots)$ represents a set of random variables that can be treated as independent, aggregation decomposes into a binary tree of applications of the aggregation operator. Figure 2 illustrates this for a case where $n = |\mathcal{D}(A):\mathcal{C}_A|$ is a power of two. The results at each level of the tree are identical, therefore we need to compute them only once. In the general case, covered by Proposition 6, we use a *square-and-multiply* method [Piṅgala, 200 BC], whose time complexity is logarithmic in $|\mathcal{D}(A):\mathcal{C}_A|$, to eliminate $p(\ldots,A,\ldots)$ from an aggregation parfactor.

Proposition 6. Let $g_A = \langle \mathcal{C},\ p(\ldots,A,\ldots),\ c(\ldots),\ \mathcal{F}_p,\ \otimes,\ \mathcal{C}_A\rangle$ be an aggregation parfactor from Φ. Assume that $\mathbb{P}(c(\ldots)) = \mathbb{P}(p(\ldots,A,\ldots)) \setminus \{A\}$ and that set $\mathcal{C} \cup \mathcal{C}_A$ is in normal form. Let $\mathbf{S}$ denote a set of random variables represented by $p(\ldots,A,\ldots)$. Assume that no other parfactor or aggregation parfactor in Φ involves parameterized random variables that represent random variables from $\mathbf{S}$. Let $m = \lfloor \log_2 |\mathcal{D}(A):\mathcal{C}_A| \rfloor$ and $b_m \ldots b_0$ be the binary representation of $|\mathcal{D}(A):\mathcal{C}_A|$. Let $(\mathcal{F}_0,\ldots,\mathcal{F}_m)$ be a sequence of factors from range of c to the reals, defined recursively as follows:

$$\mathcal{F}_0(x) = \begin{cases} \mathcal{F}_p(x), & \text{if } x \in \mathit{range}(p); \\ 0, & \text{otherwise,} \end{cases}$$

$$\mathcal{F}_k(x) = \begin{cases} \sum_{\substack{y,z \in \mathit{range}(c) \\ y \otimes z = x}} \mathcal{F}_{k-1}(y)\,\mathcal{F}_{k-1}(z), & \text{if } b_{m-k} = 0; \\ \sum_{\substack{w,y,z \in \mathit{range}(c) \\ w \otimes y \otimes z = x}} \mathcal{F}_p(w)\,\mathcal{F}_{k-1}(y)\,\mathcal{F}_{k-1}(z), & \\ & \text{otherwise.} \end{cases}$$

Then $\mathcal{J}_{\mathbf{S}}(\Phi) = \mathcal{J}(\phi \setminus \{g_a\} \cup \{\langle \mathcal{C}, \{c(\ldots)\}, \mathcal{F}_m\rangle\})$.

Proposition 6 does not allow variable $c(\ldots)$ in an aggregation parfactor to have extra parameters that are not present in variable $p(\ldots,A,\ldots)$. The C-FOVE algorithm handles extra parameters by introducing counting formulas on these parameters. Then it can proceed with standard summation. We cannot apply the same approach to aggregation parfactors as newly created counting formulas could have ranges incompatible with the range of the aggregation operator. We need a special summation procedure, described below in Proposition 7.

Proposition 7. Let $g_A = \langle \mathcal{C},\ p(\ldots,A,\ldots),\ c(\ldots,E,\ldots),\ \mathcal{F}_p,\ \otimes,\ \mathcal{C}_A\rangle$ be an aggregation parfactor from Φ. Assume that $\mathbb{P}(c(\ldots)) \setminus \{E\} = \mathbb{P}(p(\ldots,A,\ldots)) \setminus \{A\}$, that set $\mathcal{C} \cup \mathcal{C}_A$ is in normal form. Let $\mathbf{S}$ denote a set of random variables represented by $p(\ldots,A,\ldots)$. Assume that no other parfactor or aggregation parfactor in Φ involves parameterized random variables that represent random variables from $\mathbf{S}$. Let $m = \lfloor \log_2 |\mathcal{D}(A):\mathcal{C}_A| \rfloor$ and $b_m \ldots b_0$ be the binary representation of $|\mathcal{D}(A):\mathcal{C}_A|$. Let $(\mathcal{F}_0,\ldots,\mathcal{F}_m)$ be a sequence of factors from range of c to real numbers, defined recursively as follows:

$$\mathcal{F}_0(x) = \begin{cases} \mathcal{F}_p(x), & \text{if } x \in \mathit{range}(p); \\ 0, & \text{otherwise,} \end{cases}$$

$$\mathcal{F}_k(x) = \begin{cases} \sum_{\substack{y,z \in \mathit{range}(c) \\ y \otimes z = x}} \mathcal{F}_{k-1}(y)\,\mathcal{F}_{k-1}(z), & \text{if } b_{m-k} = 0; \\ \sum_{\substack{w,y,z \in \mathit{range}(c) \\ w \otimes y \otimes z = x}} \mathcal{F}_p(w)\,\mathcal{F}_{k-1}(y)\,\mathcal{F}_{k-1}(z), & \\ & \text{otherwise.} \end{cases}$$

Let $\mathcal{C}_E$ be a set of constraints from $\mathcal{C}$ that involve E. Let $\mathcal{F}_\#$ be a factor from the range of counting formula $\#_{E:\mathcal{C}_E}[c(\ldots,E,\ldots)]$ to real numbers defined as follows:

$$\mathcal{F}_\#(h()) = \begin{cases} \mathcal{F}_m(x), & \text{if } \exists x \in \mathit{range}(c)\ h(x) = |\mathcal{D}(E):\mathcal{C}_E|; \\ 0, & \text{otherwise.} \end{cases}$$

Then $\mathcal{J}_{\mathbf{S}}(\Phi) = \mathcal{J}(\phi \setminus \{g_a\} \cup \{\langle \mathcal{C} \setminus \mathcal{C}_E, \{\#_{E:\mathcal{C}_E}[c(\ldots,E,\ldots)]\}, \mathcal{F}_\#\rangle\})$.

The above proposition can be generalized to the cases where $c(\ldots)$ has more than one extra parameter.

If set $\mathcal{C} \cup \mathcal{C}_A$ is not in normal form, then $|\mathcal{D}(A):\mathcal{C}_A|$ might vary for different ground substitutions to parameters in $p(\ldots,A,\ldots)$ and we will not be able to apply Propositions 6 and 7. We can follow Milch *et al.* [2008] and bring constraints in the aggregation parfactor to a *normal form* by splitting it on appropriate substitutions. Once the constraints are in normal form, $|\mathcal{D}(A):\mathcal{C}_A|$ does not change for different ground substitutions. The other approach is to compute *uniform solution counting partitions* [de Salvo Braz *et al.*, 2007] using a constraint solver and use this information when summing out $p(\ldots,A,\ldots)$.

4.4 Generalized aggregation parfactors

Propositions 6 and 7 require that random variables represented by $p(\ldots,A,\ldots)$ are independent. They are only dependent if they either have a common ancestor in the grounding or a common observed descendant. If during inference we eliminate the common ancestor or condition on the observed descendant before we eliminate $p(\dots,A,\dots)$ through aggregation, we may introduce a counting formula on $p(\dots,A,\dots)$. This would prevent us from applying results of Propositions 6 and 7 and performing efficient lifted aggregation.

We need to delay such conditioning and summing out until we eliminate $p(\dots,A,\dots)$. It requires a generalized version of the aggregation parfactor data structure, a septuple $\langle \mathcal{C}, p(\dots,A,\dots), c(\dots,E,\dots), \mathcal{V}, \mathcal{F}_{p\cup\mathcal{V}}, \otimes, \mathcal{C}_A\rangle$ where V is a set of *context parameterized random variables* and $\mathcal{F}_{p\cup\mathcal{V}}$ is a factor from the Cartesian product of ranges of parameterized random variables in $\{p(\dots,A,\dots)\}\cup\mathcal{V}$ to the reals. The factor $\mathcal{F}_{p\cup\mathcal{V}}$ stores the dependency between $p(\dots,A,\dots)$ and context variables.

Generalization of propositions from Sections 4.2 and 4.3 is straightforward. Proposition 5 has to be generalized so aggregation parfactors can be multiplied by parfactors on variables other than $p(\dots,A,\dots)$, and generalized versions of Propositions 6 and 7 have to manipulate larger factors.

The third experiment from Section 5 involved inference with generalized aggregation parfactors.

5 Experiments

In our experiments, we investigated how the population size of parameters and the size of the range of parameterized random variables affect inference in the presence of aggregation.

We compared the performance of variable elimination (VE), variable elimination with the noisy-MAX factorization [Díez and Galán, 2003] (VE-FCT), C-FOVE, C-FOVE with the lifted noisy-MAX factorization described in Section 4.2 (C-FOVE-FCT), and C-FOVE with aggregation parfactors (AC-FOVE). We used Java implementations of the above algorithms on an Intel Core 2 Duo 2.66GHz processor with 1GB of memory made available to the JVM.

In the first experiment, we tested the above algorithms on the model introduced in Example 1 and depicted in Figure 1. In the second experiment, used a modified version of this model, in which $range(inj) = range(sub) = range(substitution) = \{0,1,2\}$, and $substitution()$ is a noisy-MAX of $inj(Player)$. In both experiments, we measured the time necessary to compute the marginal of the variable $substitution()$ using a top-down elimination ordering. We varied the population size n of the parameter *Players* from 1 to 100,000.

Figures 3 and 4 show the results of the experiments. The time complexity for VE is exponential in n, and the algorithm did not scale as n increased. The time complexity for VE-FCT is linear in n. In the model with noisy-OR, the time complexity for C-FOVE is also linear in n, but C-FOVE does lifted inference and it achieved better results then VE-FCT, which performs inference at the propositional level. In the model with noisy-MAX, the time complexity for C-FOVE is quadratic in n, and C-FOVE was outperformed by VE-FCT. C-FOVE-FCT and AC-FOVE, for which the time complexity is logarithmic in n, performed best in both cases (for clarity we did not show the C-FOVE-FCT performance in Figures 3 and 4). The difference between their performance and the performance of C-FOVE was apparent even for small populations in the second experiment , which involved aggregation over non-binary random variables.

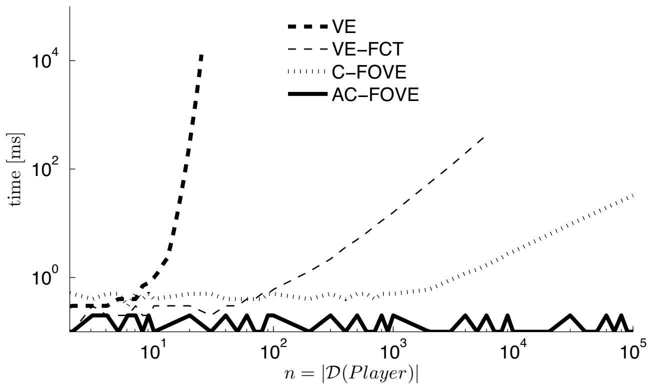

Figure 3: Performance on the model with noisy-OR aggregation.

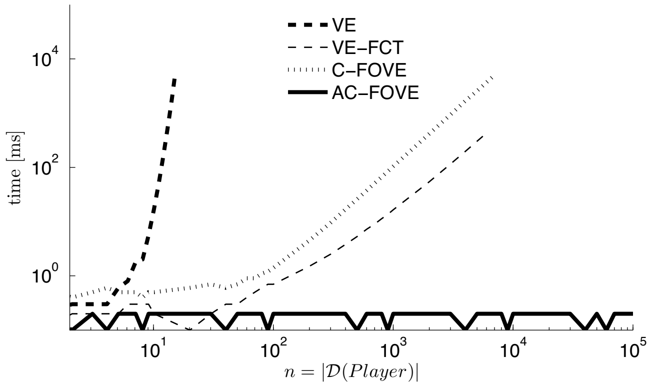

Figure 4: Performance on the model with noisy-MAX aggregation.

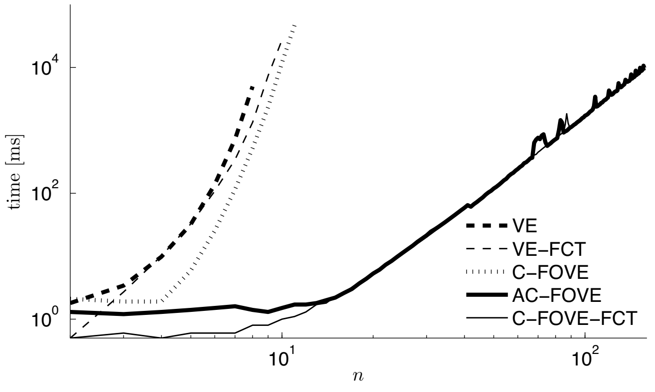

Figure 5: Performance on the smoking-friendship model.

For the third experiment we used an ICL theory [Poole, 2008] from Carbonetto *et al.* [2009] that explains how people alter their smoking habits within their social network. Parameters of the model were learned from data of smoking and drug habits among teenagers attending a school in Scotland [Pearson and Michell, 2000] using methods described by Carbonetto *et al.* [2009]. Given the population size n, the equivalent propositional graphical model has $3n^2 + n$ nodes and $12n^2 - 9n$ arcs. We varied n from 2 to 140 and for each value, we computed a marginal probability of a single individual being a smoker. Figure 5 shows the results of the experiment. VE, VE-FCT and C-FOVE algorithms failed to solve instances with a population size greater than 8, 10, and 11, respectively. AC-FOVE was able to handle efficiently much larger instances and it ran out of memory for a population size of 159. The AC-FOVE algorithm performed equally to the C-FOVE-FCT algorithm except for small populations. It is important to remember that the C-FOVE-FCT algorithm, unlike AC-FOVE, can only be applied to MAX and MIN-based aggregation.

6 Conclusions and Future Work

In this paper we demonstrated the use of aggregation parfactors to represent aggregation in directed first-order probabilistic models, and how aggregation parfactors can be incorporated into the C-FOVE algorithm. Theoretical analysis and empirical tests showed that in some cases, lifted inference with aggregation parfactors leads to significant gains in efficiency.

While presented the algorithm can handle a wide range of aggregation cases, there still exist models that can't be handled efficiently by lifted inference, for example a "lattice" structure presented by Poole [2008]. These models pose an interesting challenge for future research.

To date, all empirical evaluations of lifted inference, including the evaluation in this paper, have been performed using simple first-order probabilistic models. Now we have inference procedures that allow for more comprehensive evaluation of the practical potential of lifted inference.

Acknowledgments

Authors wish to thank Peter Carbonetto, Michael Chiang, Mark Crowley, Craig Wilson, James Wright, and IJCAI reviewers for valuable comments. This work was supported by NSERC grant to David Poole.

References

[Breese, 1992] Jack S. Breese. Construction of belief and decision networks. *Comput Intell*, 8(4):624–647, 1992.

[Buntine, 1994] Wray L. Buntine. Operations for learning with graphical models. *J Artif Intell Res*, 2:159–225, 1994.

[Carbonetto *et al.*, 2009] Peter Carbonetto, Jacek Kisyński, Michael Chiang, and David Poole. Learning a contingently acyclic, probabilistic relational model of a social network. TR-2009-08, Univ of British Columbia, Dept of Comp Sci, 2009.

[Chavira *et al.*, 2006] Mark Chavira, Adnan Darwiche, and Manfred Jaeger. Compiling relational Bayesian networks for exact inference. *Int J Approx Reason*, 42(1–2):4–20, 2006.

[De Raedt *et al.*, 2008] Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen H. Muggleton, editors. *Probabilistic Inductive Logic Programming*. Springer, 2008.

[de Salvo Braz *et al.*, 2007] Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. Lifted first-order probabilistic inference. In Lise Getoor and Ben Taskar, ed., *Introduction to Statistical Relational Learning*, 433–450. MIT Press, 2007.

[Díez and Galán, 2003] Francisco J. Díez and Severino F. Galán. Efficient computation for the noisy MAX. *Int J Intell Syst*, 18(2):165–177, 2003.

[Díez, 1993] Francisco J Díez. Parameter adjustment in Bayes networks. The generalized noisy OR-gate. In *Proc. 9th UAI*, 99–105, 1993.

[Getoor and Taskar, 2007] Lise Getoor and Ben Taskar, ed., *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[Gupta *et al.*, 2007] Rahul Gupta, Ajit A. Diwan, and Sunita Sarawagi. Efficient inference with cardinality-based clique potentials. In *Proc. 24th ICML*, 329–336, 2007.

[Horsch and Poole, 1990] Michael Horsch and David Poole. A dynamic approach to probabilistic inference using Bayesian networks. In *Proc. 6th UAI*, 155–161, 1990.

[Jaeger, 2002] Manfred Jaeger. Relational Bayesian networks: a survey. *Electron Art Comput Inform Sc*, 6, 2002.

[Koller and Pfeffer, 1997] Daphne Koller and Avi Pfeffer. Object-oriented Bayesian networks. In *Proc. 13th UAI*, 302–313, 1997.

[Milch *et al.*, 2008] Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted probabilistic inference with counting formulas. In *Proc. 23rd AAAI*, 1062–1068, 2008.

[Pearl, 1986] Judea Pearl. Fusion, propagation and structuring in belief networks. *Artif Intell*, 29(3):241–288, 1986.

[Pearson and Michell, 2000] Michael Pearson and Lynn Michell. Smoke Rings: social network analysis of friendship groups, smoking and drug-taking. *Drugs: education, prevention and policy*, 7:21–37, 2000.

[Pfeffer and Koller, 2000] Avi Pfeffer and Daphne Koller. Semantics and inference for recursive probability models. In *Proc. 17th AAAI*, 538–544, 2000.

[Piṅgala, 200 BC] Piṅgala. *Chandaḥ-sûtra*. 200 BC.

[Poole, 2003] David Poole. First-order probabilistic inference. In *Proc. 18th IJCAI*, 985–991, 2003.

[Poole, 2008] David Poole. The Independent Choice Logic and beyond. In Luc De Raedt *et al.*, ed., *Probabilistic Inductive Logic Programming*, 222–243. Springer, 2008.

[Savicky and Vomlel, 2007] Peter Savicky and Jiří Vomlel. Exploiting tensor rank-one decomposition in probabilistic inference. *Kybernetika*, 43(5):747–764, 2007.

[Zhang and Poole, 1994] Nevin L. Zhang and David Poole. A simple approach to Bayesian network computations. In *Proc. 10th AI*, 171–178, 1994.

[Zhang and Poole, 1996] Nevin L. Zhang and David Poole. Exploiting causal independence in Bayesian network inference. *J Artif Intell Res*, 5:301–328, 1996.