Generalising the Interaction Rules in Probabilistic Logic

Arjen Hommersom and Peter J. F. Lucas
Institute for Computing and Information Sciences
Radboud University Nijmegen
Nijmegen, The Netherlands
{arjenh,peterl}@cs.ru.nl

Abstract

The last two decades has seen the emergence of many different probabilistic logics that use logical languages to specify, and sometimes reason, with probability distributions. Probabilistic logics that support reasoning with probability distributions, such as ProbLog, use an implicit definition of an interaction rule to combine probabilistic evidence about atoms. In this paper, we show that this interaction rule is an example of a more general class of interactions that can be described by non-monotonic logics. We furthermore show that such local interactions about the probability of an atom can be described by convolution. The resulting extended probabilistic logic supports non-monotonic reasoning with probabilistic information.

1 Introduction

The last two decades has seen the emergence of many different probabilistic logics, such as Markov logic [Richardson and Domingos, 2006], probabilistic Horn clause logic [Poole, 1993], and ProbLog [Kimmig *et al.*, 2010], that use logical languages to specify, and sometimes reason, with probability distributions. Probabilistic logics are based on first-order logic, and they share the advantage of first-order logic in comparison to propositional logic in that they can be looked upon as more expressive knowledge-representation languages with considerable modelling power, which for probabilistic logics is extended to areas where uncertainty is encountered.

However, despite their generality, all of these languages are based on different semantic principles. Some of the principles, while taken for granted, should actually be considered as part of the design choices. When different choices would have been made, a different probabilistic logic would have emerged because of the close interaction between the logical aspects and probabilistic aspects of a probabilistic logic.

In the research described in the present paper, we introduce the theoretical foundation of a whole class of probabilistic logics, starting from a very general and unrestrictive probabilistic logic, the above-mentioned ProbLog, and show that it is possible to obtain a new, but related, probabilistic logic by changing the way probabilistic evidence is combined. We do this by redesigning the probabilistic and logical basis of the probabilistic logic hand in hand such that the resulting logic supports this double, logical and probabilistic, view. The resulting class of probabilistic logics has the advantage that it can be used to gear a logic to the problem at hand.

In the next section, we first review the basics of ProbLog, which is followed by the development of the main methods used in generalising probabilistic Boolean interaction, and, finally, default logic is briefly discussed. We will use probabilistic Boolean interaction as a sound and generic, algebraic way to combine uncertain evidence, whereas default logic will be used as our language to implement the interaction operators, again reflecting this double perspective on the probabilistic logic. The new probabilistic logical framework is described in Section 3 and compared to other approaches in Section 4. The achievements of this research are reflected upon in Section 5.

2 Preliminaries

In this section, probabilistic logic and a method for combining probabilistic evidence is described. We also briefly review concepts from default logic.

2.1 Probabilistic Logic: Syntax and Semantics

There are many different proposals in the scientific literature for probabilistic logics. In this paper we focus on the class of logics where probabilistic reasoning and logical reasoning go hand in hand. ProbLog is a typical example of such a language [Kimmig *et al.*, 2010]. ProbLog was specifically designed as a basic language to which other probabilistic logical languages can be compiled [De Raedt *et al.*, 2008]; thus, the results from this paper apply to a whole range of languages.

It is assumed that the reader is familiar with logic programming terminology, where a program consists of *Horn clauses* of the form

$$B \leftarrow A_1, \ldots, A_n.$$

where B, A_i are *atoms* of the form $p(t_1, \ldots, t_m)$, with t_k *terms*. If $n = 0$, then the Horn clause is called a *fact*, if B is $\bot$ (false), then the clause is called a *query*; otherwise, the clause is called a *rule*.

The ProbLog language extends logic programming by allowing that probabilities are attached to facts, then called *labelled facts*. Let F denote the set of labelled facts, then each element of F has the form

$$p :: f.$$

also abbreviated to p_f, where $p_f = P(f) \in [0,1]$ has the meaning of a probability, and f conforms to the syntax of an atom. The meaning of an atom in terms of probability theory is that of a set of random variables. For example,

$$0.4 :: \mathsf{parent(X)}.$$

specifies the collection of random variables $\mathsf{parent(X)}$, for each ground instance of $\mathsf{parent(X)}$, obtained by applying substitutions Θ to $\mathsf{parent(X)}$. For example, $\mathsf{parent(X)}\theta = \mathsf{parent(john)}$ with $\theta = \{\mathsf{john/X}\}$, where $\theta \in \Theta$. The resulting ground atoms are called *logical facts*.

In addition to labelled facts, a ProbLog program consists of rules, constituting the background knowledge B of the program. Now, let $\mathcal{T} = F \cup B$ be a ProbLog program and let Θ_f be the set of all possible substitutions associated with the logical fact f. Then, $L_\mathcal{T}$ is defined as the *subset-maximal* set of logical facts that can be added to B applying the set of substitutions Θ_f to the fact f, for each $f \in F$. The ProbLog program then defines a joint probability distribution on the logical facts $L_\mathcal{T}$. Let $L \subseteq L_\mathcal{T}$, then:

$$P_\mathcal{T}(L) = \prod_{f\theta \in L} p_f \prod_{f'\theta' \in L_\mathcal{T} \setminus L} (1 - p_{f'}) \,.$$

Let q now be any query to the ProbLog program, then it holds that

$$P_\mathcal{T}(q) = \sum_{L \subseteq L_\mathcal{T}} P(q \mid L) P_\mathcal{T}(L), \tag{1}$$

where

$$P(q \mid L) = \begin{cases} 1 & \text{if } \exists\theta : B \cup L \models q\theta \\ 0 & \text{otherwise} \end{cases}$$

and '$\models$' indicates logical entailment. Each L with $P(q \mid L) = 1$ is called an *explanation* of q.

The semantics of ProbLog is called the *distribution semantics*; it has been borrowed from PRISM [Sato, 1995]. Basically, in the distribution semantics all facts are assumed to be mutually independent. However, this does not imply that it is impossible to encode probabilistic dependences: all dependences are defined at the logical level. This allows defining any joint probability distribution. The distribution semantics also has particular consequences for obtaining probabilistic interactions between facts, as illustrated by the following example.

Example 1. *Consider the following (trivial) ProbLog program $\mathcal{T}$ that represents some causal, medical knowledge:*

0.7 :: flu.
0.2 :: pneumonia.

fever $\leftarrow$ flu.
fever $\leftarrow$ pneumonia.

In this program flu *and* pneumonia *are two independent random variables according to the distribution semantics. We now wish to compute the probability* $P_\mathcal{T}(\mathsf{fever})$. *Note that* fever *can be explained from either* flu, pneumonia, *or both* flu *and* pneumonia. *Thus, according to Equation (1) we get:*

$$\begin{aligned} P_\mathcal{T}(\mathsf{fever}) &= P(\mathsf{flu})P(\mathsf{pneumonia}) \\ &\quad + P(\mathsf{flu})(1 - P(\mathsf{pneumonia})) \\ &\quad + (1 - P(\mathsf{flu}))P(\mathsf{pneumonia}) \\ &= 0.7 \cdot 0.2 + 0.3 \cdot 0.2 + 0.7 \cdot 0.8 \\ &= 0.14 + 0.06 + 0.56 \\ &= 0.76 \end{aligned}$$

As is well known, this probabilistic result is identical to what would have been obtained by one of the most popular ways to model the interaction of (conditionally) independent events: the so-called *noisy-OR* [Pearl, 1988]. With the noisy-OR one models an uncertain, disjunctive interaction between events.

As the noisy-OR is based on logical disjunction, just one of the 16 binary Boolean operators, one can imagine that there might be other, equally sound, ways to combine probabilistic evidence. However, in order to be able to combine such evidence, one needs an algebraic method to combine such information. A general, algebraic way to combine probabilistic information is available from basic probability theory although it is rarely used to model Boolean interaction. In the following, we briefly review the necessary basics from probability theory, with convolution as a special case, and then investigate how ideas from probabilistic logic, Boolean interaction and default logic can be merged to obtain a more expressive probabilistic logic, which we call *probabilistic interaction logic*, or ProbIL for short.

2.2 Probabilistic Boolean Interaction

In the following, a probability mass function of a random variable X is referred to by f_X; P denotes the associated probability distribution. A classical result from probability theory that is useful when studying sums of variables is the following well-known theorem (cf. [Grimmett and Stirzaker, 2001]).

Theorem 1. *Let f be a joint probability mass function of the random variables X and Y, such that $X + Y = z$. Then it holds that $P(X + Y = z) = f_{X+Y}(z) = \sum_x f(x, z - x)$.*

Proof. See [Grimmett and Stirzaker, 2001]. □

If X and Y are independent, then, in addition, the following corollary holds.

Corollary 1. Let X and Y be two independent random variables, then it holds that

$$\begin{aligned} P(X + Y = z) &= f_{X+Y}(z) \\ &= \sum_x P(X = x)P(Y = z - x) \\ &= \sum_x f_X(x) f_Y(z - x) \end{aligned} \tag{2}$$

The probability mass function f_{X+Y} is in that case called the *convolution* of f_X and f_Y, and it is commonly denoted as $f_{X+Y} = f_X * f_Y$. The convolution theorem is very useful, as sums of independent random variables occur very frequently

in probability theory and statistics. It can also be applied recursively, i.e.,

$$f_{X_1+\cdots+X_n} = f_{X_1} * \cdots * f_{X_n}$$

as follows from the recursive application of Equation (2).

Theorem 1 does not only hold for the addition of two random variables, but also for Boolean functions of random variables. However, in contrast to the field of real numbers where a value of a random variable Y is uniquely determined by a real number x and z through $Y = z - x$, in Boolean algebra values of Boolean variables only *constrain* the values of other Boolean variables. These constraints may yield a set of values, rather than a single value, which is still compatible with the theorem. In the following, we use the notation $b(i, J) = q$ for such constraints, where the Boolean values i and q constrain J to particular values. For example, for $(i \vee J) = q$, where i, q stand for $I = 1$ (I has the value 'true') and $Q = 1$ (Q has the value 'true'), it holds that $J \in \{0, 1\}$. Thus, $f(i, (i \vee J) = q)$ is an abbreviation for $f(i, j) + f(i, \neg j)$. The theorem can then be re-expressed as follows.

Theorem 2. *Let f be a joint probability mass function of the random variables I and J such that $b(I, J) = q$, with b a Boolean function. Then, it holds that $P(b(I, J) = q) = f_{b(I,J)}(q) = \sum_i f(i, b(i, J) = q)$.*

Proof. The (I, J) space defined by $b(I, J) = q$ can be decomposed as follows: $\bigcup_i \{I = i\} \cap \{J = j \mid b(i, j) = q\}$, where the expression $b(i, j) = q$ should be interpreted as a logical constraint on the Boolean values of the variable J. Since the individual sets $\{I = i\} \cap \{J = j \mid b(i, j) = q\}$ are mutually exclusive, the result follows. □

The following corollary for convolution is obtained if I and J are independent.

Corollary 2. Let f be a joint probability mass function of independent random, Boolean variables I and J and let b be a Boolean function defined on I and J, then it holds that

$$P(b(I, J) = q) = \sum_i f_I(i) P(b(i, J) = q) \; .$$

Theorem 2 now appears to be the key insight to generalise ProbLog.

Example 2. *Reconsider Example 1, and the Boolean relation $L \vee P = F$, where L stands for 'flu,' P for 'pneumonia', and F for 'fever'. We use the same probability distributions as in Example 1: $P(l) = 0.7$ and $P(p) = 0.2$. By applying Theorem 2 the following results:*

$$\begin{aligned} &P(L \vee P = f) \\ &= \sum_l f_l(l) P(l \vee P = f) \\ &= f_L(l) \left(f_P(p) + f_P(\neg p)\right) + f_L(\neg l) f_P(p) \\ &= f_L(l) f_P(p) + f_L(l) f_P(\neg p) + f_L(\neg l) f_P(p) \\ &= 0.7 \cdot 0.2 + 0.7 \cdot 0.8 + 0.3 \cdot 0.2 = 0.76 \end{aligned}$$

where the term $f_P(p) + f_P(\neg p)$ results from the logical constraint that $l \vee P = f$, i.e., $P \in \{0, 1\}$.

Thus, the example demonstrates that the noisy-OR can be described quite naturally by convolution. See [Lucas and Hommersom, 2010] for details.

The correspondence between the two approaches is as follows. The Boolean function b corresponds to the deterministic probability distribution $P(q \mid L)$ of Equation (1). As $P(q \mid L)$ is defined in terms of logical entailment of $q\theta$ from $B \cup L$, the question addressed in this paper is whether there are ways to replace logical entailment by a reasoning method that incorporates probabilistic Boolean interaction as a method to express interaction between heads of rules. Since Boolean interaction can be looked upon as an inference rule, a natural way to extend ProbLog is by replacing standard logic by default logic.

2.3 Default Logic

In default logic [Reiter, 1980] one adds special inference rules, called defaults, to ordinary first-order predicate logic. The *defaults* have the following form:

$$\frac{\textit{prerequisite} : \textit{justifications}}{\textit{consequent}}$$

where 'prerequisite' is a condition that, if it is true, then 'consequent' can be derived, however, *only* when the resulting theory together with the assumptions described by the 'justifications' is consistent. The resulting theory is denoted as $\mathcal{T} = (D, W)$, where in this paper W stands for a set of facts and rules in Horn-clause logic as in ProbLog, and D are defaults. Inference in default logic is done by computing the extensions of the theory, using a fixed point operator. We will write $(D, W) \mathrel{|\!\sim} \varphi$ if $\varphi \in E$, where E is a default extension of (D, W).

The next section will develop the probabilistic interaction logic, ProbIL, by using defaults to represent Boolean interaction, which, when combined with probabilities yields probabilistic Boolean interaction as developed in Section 2.2.

3 Probabilistic Interaction Logic

3.1 General Idea

As Example 1 illustrates, probabilistic languages as ProbLog implicitly combine probabilistic evidence using logical disjunction, which corresponds to the noisy-OR operator. This choice gives rise to a particular probabilistic behaviour, that may not always be justified. The resulting probability when using the noisy-OR is always larger than its components, i.e., the probabilistic behaviour is monotonically increasing: $pq + p(1 - q) + (1 - p)q \geq p, q$. Thus, it is not possible to model that particular events, when taken together, cancel each other out. In contrast, non-monotonic logics such as default logic can be used to implement such reasoning.

The general idea of Probabilistic Interaction Logic (ProbIL) is to replace the background theory by a default logic theory. This logic has the same benefits as default logic, namely that we can specify problem-dependent default behaviour without sacrificing ordinary logical deduction. In the context of ProbIL, defaults model valid interactions dependent of the generic problem-solving method that needs to be expressed or of the actual problem at hand.

Formally, the probability $P(q \mid L)$ is adapted as follows. We assume we have a background theory consisting of a set of defaults D and a set of standard logical rules B. Then we have:

$$P(q \mid L) = \begin{cases} 1 & \text{if } \exists\theta : (D, B \cup L) \mid\sim q\theta \\ 0 & \text{otherwise} \end{cases}$$

i.e., the monotonic logical entailment $\models$ has been replaced by the default logic entailment $\mid\sim$.

Example 3. *The popular example from non-monotonic logics "birds typically fly" is modelled by the default set* $D = \left\{ \frac{\mathsf{Bird}(x) : \mathsf{Flies}(x)}{\mathsf{Flies}(x)} \right\}$ *and we add the standard logic rule expressing that Penguins do not fly, i.e.,* $B = \{\mathsf{Penguin}(x) \rightarrow \neg\mathsf{Flies}(x)\}$. *Further, we might know that in a pet store 30% of the animals are birds, i.e.,* $0.3 :: \mathsf{Bird}(x)$. *To compute whether Tweety (T) flies, we compute,* $P(\mathsf{Flies}(\mathsf{T})) = P(\mathsf{Flies}(\mathsf{T}) \mid \mathsf{Bird}(\mathsf{T}))P(\mathsf{Bird}(\mathsf{T})) = 0.3$.

This has the same representational benefit as default logic in comparison to standard logic, i.e., the example could also be formalised by the ProbLog rule:

$$\mathsf{Bird}(x) \wedge \neg\mathsf{Penguin}(x) \rightarrow \mathsf{Flies}(x)$$

However, if, as in the example, we do not know whether Tweety flies and we do not have a distribution for Tweety being a penguin, then $P(\mathsf{Flies}(\mathsf{T})) = 0$. This illustrates the interplay between non-monotonic and probabilistic reasoning, which is not available in many existing probabilistic logics.

In the following we use this mechanism to study types of interactions that can be modelled and ways to represent the resulting probability mass explicitly by using probabilistic Boolean interaction, and by a convolution operator in case of independence.

3.2 Boolean Interactions

Consider the following ProbLog clauses: $\{a \leftarrow c, a \leftarrow e\}$. In order to model the interaction between the probabilistic evidence contributing to the probability of a, we need to specify a combination function for these clauses, i.e., we wish to interpret these clauses as $a \leftarrow b(c, e)$, with b a Boolean function. As we have seen, in the case of ProbLog, $b(c, e)$ is always equal to $c \vee e$. Default logic, as an expressive logical formalism, is used here to reason about these Boolean interactions. The following proposition expresses that all possible interpretations of Boolean interactions can indeed be modelled in default logic.

Proposition 1. *For all Boolean functions* b *and a set of atoms* $H, B_1, \ldots, B_n$ *there is a default logic theory* D*, which does not contain* b*, such that:*

$$\exists\theta\colon (D, B \cup L) \mid\sim q\theta$$

iff

$$\exists\theta\colon B \cup L \cup \{H \leftarrow b(B_1, \ldots, B_n)\} \models q\theta$$

To illustrate this, we list a number of common binary Boolean functions in Figure 1 with their associated default logic theory. Hence, for a given set of interactions, we may replace the relevant rules by the corresponding theory where the interactions are modelled by default rules.

Name	*Default rules* D	*Choices for* L
false	$\varnothing$	$\varnothing$
true	$\left\{\frac{\top}{H}\right\}$	$\wp(\{B_1, B_2\})$
AND	$\left\{\frac{B_1, B_2}{H}\right\}$	$\{\{B_1, B_2\}\}$
OR	$\left\{\frac{B_1}{H}, \frac{B_2}{H}\right\}$	$\wp(\{B_1, B_2\}) \setminus \{\varnothing\}$
EQ	$\left\{\frac{B_1, B_2}{H}, \frac{: \neg B_1, \neg B_2}{H}\right\}$	$\{\varnothing, \{B_1, B_2\}\}$
XOR	$\left\{\frac{B_1 : \neg B_2}{H}, \frac{B_2 : \neg B_1}{H}\right\}$	$\{\{B_1\}, \{B_2\}\}$

Figure 1: Representation of the binary commutative and associative Boolean functions using default logic for rule R: $H \leftarrow b(B_1, B_2)$ such that $(D, L) \mid\sim H$ iff $\{R\} \cup L \models b(B_1, B_2)$. In this, $\wp(L)$ denotes the powerset of L. Other, less common Boolean operators can be defined similarly.

Of course, the converse of Proposition 1 also holds: extensions of a default theory can be characterised using a Boolean function. In the following, we assume that we have a Boolean function to model the interaction.

Example 4. *People having headache often use over-the-counter pain medication to obtain pain relief (*r*). However, scientific evidence indicates that headache can actually be triggered by the overuse of common pain killers. Suppose we have two pain killers* k_1 *and* k_2 *which in combination are ineffective against headache. Also assume that pain killer* k_1 *thins the blood (*t*). Finally assume we have two probabilistic facts* d_1 *and* d_2 *that model the probabilistic decision of whether or not to take* k_1 *and* k_2*. We can formalise this using the following ProbLog theory:*

$$\begin{array}{ll} \{\, p_1 :: d_1 & p_2 :: d_2 \\ \quad r \leftarrow k_1 & k_1 \leftarrow d_1 \\ \quad r \leftarrow k_2 & k_2 \leftarrow d_2 \\ & t \leftarrow k_1 \,\} \end{array}$$

As k_1 *and* k_2 *undermine each other effect on* r*, we would like to include an XOR interaction function for* r*. Thus, the probabilistic rules should be interpreted as the default theory:*

$$D \;=\; \left\{\frac{k_1 : \neg k_2}{r}, \frac{k_2 : \neg k_1}{r}\right\}$$

together with a purely logical theory:

$$B = \{t \leftarrow k_1, k_1 \leftarrow d_1, k_2 \leftarrow d_2\}.$$

We are interested in hypotheses $L \subseteq \{d_1, d_2\}$ *such that:*

$$(D, B \cup L) \mid\sim Q$$

with $Q \subseteq \{r, t\}$*. For* r*, the only explanations are* $\{d_1\}$ *and* $\{d_2\}$*, but not* $\{d_1, d_2\}$*. For* t*, on the other hand, the explanations are* $\{d_1\}$ *and* $\{d_1, d_2\}$*. Hence, the only common explanation for* r *and* t *is* $\{d_1\}$.

3.3 Probabilistic Boolean Interaction

Default logic provides a reasoning mechanism for Boolean interactions. In this section, we explore the complementary probabilistic perspective, by generalising results presented in Section 2.2. As mentioned, Theorem 2 can be used as a basis for probabilistic interaction logic.

Theorem 3. *Let Q be an atom that appears in the theory $\mathcal{T}$ by a single rule:*

$$Q \leftarrow b(I, J).$$

For the probability mass function f_Q of Q holds:

$$f_Q(q) = \sum_i f(i, b(i, J) = q) \ .$$

Proof.

$$\begin{aligned} f_Q(q) &= \sum_{L \subseteq L_\mathcal{T}} P(q \mid L) P_\mathcal{T}(L) \\ &= \sum_{L \subseteq L_\mathcal{T}} P(b(I, J) = q \mid L) P_\mathcal{T}(L) \\ &= \sum_{L \subseteq L_\mathcal{T}} \sum_i P(i, b(i, J) = q \mid L) P_\mathcal{T}(L) \\ &= \sum_i f(i, b(i, J) = q) \end{aligned}$$

□

Example 5. *Reconsider Example 4. As the explanations of r are $\{d_1\}$ and $\{d_2\}$, we have:*

$$P(r) = p_1(1 - p_2) + (1 - p_1)p_2 = p_1 + p_2 - 2p_1p_2$$

Using the standard semantics of ProbLog, note that $f(k_1, \neg k_2) = P(k_1 \wedge \neg k_2) = p_1(1 - p_2)$ and $f(\neg k_1, k_2) = (1 - p_1)p_2$. By applying Theorem 3, we obtain

$$\begin{aligned} P(r) &= f(r) = P(\textit{XOR}(k_1, k_2) = r) \\ &= f(k_1, \neg k_2) + f(\neg k_1, k_2) \\ &= p_1(1 - p_2) + (1 - p_1)p_2 \end{aligned}$$

and the results correspond. However, if we had used the standard (noisy-OR) semantics of ProbLog, we would have obtained $P^*(r) = P(\textit{OR}(k_1, k_2) = r) = p_1p_2 + (1 - p_1)p_2 + p_1(1 - p_2)$, which is larger than or equal to $P(r)$.

An assumption that is often made in probabilistic logics (e.g., [Poole, 1993]), is that explanations are mutually exclusive, i.e., $P(L_i \vee L_j) = P(L_i) + P(L_j)$, with L_i and L_j explanations. Without this assumption, the probability of the disjunction is computed by finding an equivalent disjunction for $L_i \vee L_j$ where the disjuncts are mutually exclusive (called the *disjoint-sum* problem). Similarly, if we assume that bodies of a head, say B_i and B_j, are independent of each other, no disjoint-sum problem has to be solved as then it holds that $P(B_i \vee B_j) = P(B_i) + P(B_j) - P(B_i)P(B_j)$. This can be seen as a somewhat 'weaker' requirement, as this does not restrict the structure of logical theories, but only the probability distribution that is generated. In the case of independent bodies for a query, convolution can be exploited.

Corollary 3. Let Q be an atom that appears in theory $\mathcal{T}$ as a single rule:

$$Q \leftarrow b(I, J)$$

where I and J independent. Then the probability mass function f_Q of Q:

$$f_Q(q) = \sum_i f_I(i) P(b(i, J) = q) \ .$$

Example 6. *Reconsider again Example 4. It is not difficult to see that $f(k_1) = p_1$ and $f(k_2) = p_2$. So, we may use the convolution as follows:*

$$\begin{aligned} P(r) &= \sum_{k_1} f(k_1) P(\textit{XOR}(k_1, k_2) = r) \\ &= f(k_1)f(\neg k_2) + f(\neg k_1)f(k_2) \\ &= p_1(1 - p_2) + (1 - p_1)p_2 \end{aligned}$$

which yields again the same result as before. However, in this case the computation of $f(k_1)$ is independent of $f(k_2)$.

3.4 Representation

In case the independence assumption does not hold, a disjoint-sum problem has to be solved, e.g., using binary decision diagrams (BDDs) [Kimmig *et al.*, 2010]. However, it cannot be determined beforehand when this problem has to be solved, so a representation of such independence can avoid the overhead of BDDs and improve the inference in practice.

Syntactically, we introduce a labelling r_i of rules, and use this to denote a Boolean interaction between rules with the same head. For example:

$$\{r_1 : a \leftarrow c, \ \ r_2 : a \leftarrow e, \ \ b(r_1, r_2)\}$$

In this specification $b(r_1, r_2)$ expresses that

$$f(a) = f(c) \textcircled{b} f(e)$$

where $\textcircled{b}$ denotes convolution using Boolean operator b. This operator is useful for algebraic manipulation of models with complex interactions between formulae, as the algebraic properties of the Boolean operator b carry over to the $\textcircled{b}$ operator. Moreover, it directly simplifies inference as it prevents the overhead of solving the disjoint-sum problem for a. Finally, the representation is generic for both discrete models, continuous models, and mixtures of these [Lucas and Hommersom, 2010], so the results generalise to probabilistic logical models with continuous distributions, such as proposed in [Gutmann *et al.*, 2010].

4 Related Work

It is well-known that default logic can be embedded in a logic programming language with negation as failure [Kakas, 1994]. A probabilistic variant of such a logic programming language with such negations is ICL [Poole, 1997]. Another approach is the language of P-log [Baral *et al.*, 2009], which extends the Answer Set Programming (ASP) framework with probabilistic facts. Both approach are closely related to ProbIL in the sense that they both are probabilistic non-monotonic logics. The goal of this paper, however, is

complementary as we are not focused on non-monotonic theories per se, but rather, theories with different types of interactions between bodies. In fact, it holds for both ICL and P-log that the interactions between bodies follow a noisy-OR interaction. In contrast, this paper has showed that interactions can be faithfully represented in a non-monotonic logic and can sometimes be decomposed using a type of convolution operator. ProbIL is used as a basic language to give insight to the former goal as it very intuitively can represent causes that cancel each other out. From a more practical point of view, other systems could be used to actually represent and compute these probabilities.

5 Conclusions

In this paper, we proposed a new mechanism for modelling interactions in probabilistic logics. As suggested in [Poole, 1990], abduction can be seen as formalism for explaining observations, whereas default logic is used to make predictions. While languages such as ProbLog use a non-monotonic approach (abduction) for explaining a possible query, predictions are completely monotonic. We have used default reasoning in this context of prediction as a method for modelling interactions between probabilistic facts.

Default reasoning is difficult in general; in its full generality, default abduction is intractable [Eiter *et al.*, 1997]. Nonetheless, algorithms for solving this problem for propositional default theories have been developed that use efficient quantified Boolean formula solvers [Tompits, 2003]. Moreover, we have shown that for many classes of problems, the probabilistic interactions can be decomposed and effectively represented by a convolution operator. This may significantly improve the computational complexity of reasoning in a way similar to the use of causal independence models in Bayesian networks [Zhang and Poole, 1996]. Although we have not presented a practical implementation of ProbIL in this paper — ProbIL is rather acting as a general language for representing interactions — it could be implemented using some of the existing approaches in probabilistic logics based on logic programming. Actually developing such implementations is future research.

Important in this work is that we have incorporated different methods of reasoning into a flexible probabilistic logic, while still maintaining the overall design aim of providing logical and probabilistic reasoning hand in hand. We believe that this can be of help for modelling and reasoning about a wide range of actual problems.

Acknowledgments

Arjen Hommersom was supported by VENI Grant 639.021.918 from The Netherlands Organization of Scientific Research. We thank the reviewers for their constructive comments which have significantly improved this paper.

References

[Baral *et al.*, 2009] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9:57–144, 2009.

[De Raedt *et al.*, 2008] L. De Raedt *et al.* Towards digesting the alphabet-soup of statistical relational learning. In *NIPS 2008 Workshop on Probabilistic Programming*, 2008.

[Eiter *et al.*, 1997] T. Eiter, G. Gottlob, and Leone N. Semantics and complexity of abduction from default theories. *Artificial Intelligence*, 90(1-2):177–223, 1997.

[Grimmett and Stirzaker, 2001] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, Oxford, 2001.

[Gutmann *et al.*, 2010] B. Gutmann, M. Jaeger, and L. De Raedt. Extending problog with continuous distributions. In *Proc ILP2010*, 2010.

[Kakas, 1994] A. Kakas. Default reasoning via negation as failure. In G. Lakemeyer and B. Nebel, editors, *Foundations of Knowledge Representation and Reasoning*, volume 810 of *LNCS*, pages 160–178. Springer Berlin / Heidelberg, 1994.

[Kimmig *et al.*, 2010] A. Kimmig, B. Demoen, L. De Raedt, V. Santos Costa, and R. Rocha. On the implementation of the probabilistic logic programming language ProbLog. *Theory and Practice of Logic Programming*, 2010.

[Lucas and Hommersom, 2010] P.J.F. Lucas and A.J. Hommersom. Modelling the interactions between discrete and continuous causal factors in Bayesian networks. In *Proc PGM-2010*, pages 185–192, 2010.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Inteligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Poole, 1990] D. Poole. A methodology for using a default and abductive reasoning system. *International Journal of Intelligent Systems*, 5(5):521–548, 1990.

[Poole, 1993] D. Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.

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

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

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

[Sato, 1995] T. Sato. A statistical learning method for logic programs with distribution semantics. In L. Sterling, editor, *Int Conf Logic Programming*, pages 715–729. MIT Press, 1995.

[Tompits, 2003] H. Tompits. Expressing default abduction problems as quantified Boolean formulas. *AI Commun.*, 16:89–105, June 2003.

[Zhang and Poole, 1996] N.L. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *JAIR*, 5:301–328, 1996.