International Journal of Approximate Reasoning 83 (2017) 43–59

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

# Negative probabilities in probabilistic logic programs ☆

David Buchman, David Poole

*Department of Computer Science, University of British Columbia, Vancouver, BC, Canada*

ARTICLE INFO

*Article history:*
Received 29 January 2016
Received in revised form 11 July 2016
Accepted 6 October 2016
Available online 13 October 2016

*Keywords:*
Negative probability
Cyclic logic programs
Representation
Negation
Canonical

ABSTRACT

We consider probabilistic logic programs (PLPs) for non-extreme distributions. We show that in the relational case with fixed populations, PLPs cannot represent many non-extreme distributions, even using negations. We introduce negative rule probabilities in PLPs, and show they make the language strictly more expressive. In addition, they render negations unnecessary: negations in PLPs can be translated to rules with negative probabilities, thus avoiding the problem of logical inconsistency. Furthermore, this translation keeps the PLP size compact (assuming the number of negations per rule is small). This translation algorithm also allows algorithms for exact inference that do not support negations to be applicable to PLPs with negations.

The noise probabilities for non-exclusive rules are difficult to interpret and unintuitive to manipulate. To alleviate this we define "probability-strengths", an alternative representation for probabilistic values, which results in an intuitive additive algebra for combining rules.

For acyclic propositional PLPs we prove what constraints on the strengths allow for proper distributions on the non-noise variables and allow for all non-extreme distributions to be represented. We show how arbitrary CPDs can be converted into this form in a canonical way.



## 1. Introduction

In recent years, there is rising interest in combining probabilistic reasoning and logic formalisms. Logic programs have the advantage of an intuitive declarative and procedural interpretation [10]. Probabilistic modeling, on the other hand, is a well-understood formalism for representing quantified uncertainty. ICL [13,14] is such a combined formalism. ICL allows to confine randomness to independent probabilistic "noise variables", arguing that it is better to invent new hypotheses to explain probabilistic dependencies rather than dealing with implicit dependencies implied by the language. A few more languages include Prism [16,17], ProbLog [3], and CP-logic [19].

Negations are a way to extend program expressiveness, but they introduce the problem of logical inconsistencies. In this paper we prove that even with negations, program expressiveness is limited. We introduce negative noise probabilities as an alternative to negations, and show that expressiveness is strictly enhanced relative to using negations, while avoiding the problem of logical inconsistencies. Introducing "negative probabilities" means that intermediate probabilistic values are allowed to be negative *as if* that was meaningful, as long as the program's semantics, the marginal distribution over the variables of interest (the non-noise variables) is nonnegative.

☆ This paper is part of the virtual special issue on the Probabilistic Logic Programming 2015, edited by Joost Vennekens and Fabrizio Riguzzi.

*E-mail addresses:* davidbuc@cs.ubc.ca (D. Buchman), poole@cs.ubc.ca (D. Poole).
*URLs:* http://www.cs.ubc.ca/~davidbuc (D. Buchman), http://www.cs.ubc.ca/~poole (D. Poole).

http://dx.doi.org/10.1016/j.ijar.2016.10.001
0888-613X/

Dıez and Galán [4] used negative probabilities to optimize computation for the noisy-or in probabilistic graphical models. Kisynski and Poole [8] examined extending this approach to noisy-or to first order logic. Jha and Suciu [7] examined using negative probabilities for answering queries in probabilistic databases. Van den Broeck et al. [18] use negative probabilities to allow for skolemization in the presence of existential quantifiers, thus allowing efficient model counting.

Negative probabilities allow the cancellation of evidence. It also allows the cancellation of rules from a program, by adding a rule whose "probabilistic strength" is negated. Meert and Vennekens [12] suggested a new language feature of negation in the head of rules for CP-logic, that supports a cancellation effect of existing rules.

## 2. Programs

### 2.1. Propositional programs

We use capital letters for random variables, and lower-case letters for them being *true*, e.g., $b$ means $B = true$. A **(probabilistic) rule** has the form $p:\ head \longleftarrow body$, where $p$ is a number in the range $[0, 1]$ representing a probability, *head* is a positive literal, and *body* is a conjunction of positive or negative literals (excluding *head* and $\neg head$). When $p = 1$, it can be omitted, and the rule is called a **deterministic rule**.

Sometimes, the following convention is adopted: Non-deterministic rules are limited to have the special form $p_i:\ n_i$ (called a **probabilistic fact**), where each such variable $n_i$ is called a **noise variable**, and $n_i$ is not allowed to appear as a head of another rule. Adopting this convention does not limit expressiveness, since any rule $p:\ head \longleftarrow body$ can be represented using $p:\ n$ and $head \longleftarrow n \wedge body$, where $n$ is a new (auxiliary) noise variable that is not used in any other rule. In this paper, we do not adopt this convention.

A **probabilistic logic program (PLP)** is a *multiset* of rules. For convenience, we sometimes treat programs as sets; however, unions may produce programs with recurring rules. A program with only deterministic rules is a **deterministic program**. Programs can be either **cyclic** or **acyclic**.

A **model** is an assignment to all the variables, and is represented as a set of positive literals. We use the stable-model semantics [6]. We take the semantics $M(D)$ of a deterministic program $D$ to be the model which is its unique stable model. If it does not have a unique stable model, we call the program **"(logically) inconsistent"**. Inconsistency only arises in cyclic programs, when a cycle of rules contains a negation [1].

A **deterministic program realization (DPR)** for a (propositional) program $R$ is a deterministic program derived from $R$ by having every rule $p_i:\ head_i \longleftarrow body_i$ either omitted or converted to the deterministic rule $head_i \longleftarrow body_i$. A probabilistic program $R$ represents a distribution over its $2^{|R|}$ DPRs, where, independently, each rule $p_i:\ head_i \longleftarrow body_i$ is converted to $head_i \longleftarrow body_i$ with probability $p_i$ or omitted with probability $1 - p_i$. The unique stable-model semantics provides a semantics of a unique model for each DPR. This provides the PLP with a **semantics** of a distribution over models, which represents a **joint distribution of the variables** $\mathrm{P}(\mathbf{V})$. The joint distribution is **extreme** if the probability of some possible model is 0.

We say a program is **invalid** if its semantics is ill-defined. This can happen in two cases:

1. Using negations, a cyclic program may have a positive probability of (logical) inconsistency; and
2. Using negative probabilities, $\mathrm{P}(\mathbf{V})$ may be "improper". This is defined in Section 2.3.

### 2.2. Relational programs

The relational setting is a generalization of the propositional setting. The relational setting introduces **populations**, which are sets of **individuals**. Variables may now be parametrized by **logical variables**, which refer to individuals from the populations. Since rules contain variables, they may also be parametrized by logical variables. A **relational (first-order) PLP** is a generalization of PLPs for relational settings, where rules may be parametrized. A **grounding** is the process of converting a relational PLP to a propositional PLP called a ground PLP. Grounding consists of specifying the individuals in each population, and creating (or "instantiating") "ground" (unparametrized) variables for each possible parametrization of a parametrized variable by individuals. Parametrized rules are similarly instantiated, giving ground (unparametrized) rules involving the ground variables.

Relational PLPs may also use **constants**, that refer to specific individuals in the population (determined by the grounding). We do not make use of constants in this paper.

The concepts of models, DPRs, and the semantics of a PLP as a joint distribution, are not defined directly for relational PLPs; they are only defined for each of its ground (propositional) PLPs.

Since the population may contain individuals not marked by constants, a given relational PLP may be defined once, but applied to different populations (unlimited in size), thus giving an unlimited number of different ground PLPs (with different numbers of ground variables).

Importantly, relational PLPs imply constraints on the ground PLPs: individuals which are not marked by constants, are indistinguishable.

### 2.3. Negative rule probabilities

In this paper we propose to use "negative probabilities" for PLP rules. However, since probabilities cannot be negative, this necessitates an explanation of what the semantics of such programs is.

The PLP semantics we defined can be thought of as a mathematical function *PLPSemantics*$(\cdot)$, that receives the description of a PLP $R$ (a set of rules with associated probabilities) as input, and outputs its semantics, which is a joint distribution of the variables, $\mathrm{P}(\mathbf{V})$:

$$\mathit{PLPSemantics}(R) = \mathrm{P}(\mathbf{V})$$

If we fix the rules but not their probabilities $p_1, p_2, \ldots, p_{|R|}$, we can describe the semantics as a mathematical function of the rule probabilities:

$$\mathit{PLPSemantics}(p_1, p_2, \ldots, p_{|R|}) = \mathrm{P}(\mathbf{V})$$

We usually think of the rule probabilities $p_1, \ldots, p_{|R|}$ as having a meaning (the probability of a rule appearing). This meaning usually aids our intuition, but here this meaning ends up getting in our way, because it suggests $p_1, \ldots, p_{|R|}$ must be nonnegative. *The crucial insight is that it is not essential to attach a meaning to* $p_1, \ldots, p_{|R|}$. We can view $p_1, \ldots, p_{|R|}$ as meaningless parameters of some arbitrary probabilistic model, whose semantics is defined by the function *PLPSemantics*$(p_1, \ldots, p_{|R|})$. If one views $p_1, \ldots, p_{|R|}$ as meaningless parameters, one can then use arbitrary (e.g., negative) values for these parameters.

We now analyze *PLPSemantics*$(\cdot)$ as a mathematical function that receives meaningless numerical inputs $p_1, \ldots, p_{|R|}$ and produces the output $\mathrm{P}(\mathbf{V})$, and see what happens when some of its inputs are negative. To assign a function the meaning of a "distribution", it must be nonnegative and to sum to 1. A PLP $R$ represents a function over its $2^{|R|}$ DPRs that we previously called a "distribution", and this leads to a second function $\mathrm{P}(\mathbf{V})$ that we call a "joint distribution". Assume some inputs (that we previously called "rule probabilities") are negative. The first function, which we previously took to represent the "probability" of DPRs, may now be negative for some DPRs, but we ignore this fact, as this first function is only used as an intermediate computation, and we need not attach a probabilistic meaning to this function. The second function, $\mathrm{P}(\mathbf{V})$, still sums to 1, but may or may not be nonnegative.

1. If $\mathrm{P}(\mathbf{V})$ contain negative probabilities, we cannot attach a probabilistic meaning to it. We then say $\mathrm{P}(\mathbf{V})$ and $R$ are **"improper"**, and thus $R$ is invalid – i.e., $R$ has ill-defined semantics.
2. If $\mathrm{P}(\mathbf{V})$ is nonnegative, then we can attach a probabilistic meaning to it, and the semantics of $R$ is well-defined. We thus ignore the meaninglessness of $p_1, \ldots, p_{|R|}$ and of the intermediate negative "probabilistic" values, since the program has a proper joint distribution semantics over the non-noise variables.

**Example 1.** Consider the probabilistic acyclic program:

$$\begin{aligned} R = \{\ & p_1 : a, && p_1 = 0.5 \\ & p_2 : b, && p_2 = 0.7 \\ & p_3 : b \longleftarrow a\ \} && p_3 = -\tfrac{4}{3} \end{aligned}$$

The program defines an improper "distribution" over the $2^3$ DPRs (the "first function"), since, for example, the "probability" of the DPR $\{a, b \longleftarrow a\}$ is negative: $\mathrm{P}(\{a, b \longleftarrow a\}) = 0.5 \cdot (1 - 0.7) \cdot (-\frac{4}{3}) = -0.2 < 0$. However, the DPRs $\{a, b, b \longleftarrow a\}$ and $\{a, b\}$ have the same unique stable model as $\{a, b \longleftarrow a\}$, and the sum of their probabilities is positive. If we thus only consider the joint distribution over the variables, $\mathrm{P}(A, B)$, we find it is proper, and can be described by $\mathrm{P}(A, B) = \mathrm{P}(A)\,\mathrm{P}(B \mid A)$, where $\mathrm{P}(a) = p_1 = 0.5$, $\mathrm{P}(b \mid \neg a) = p_2 = 0.7$ and $\mathrm{P}(b \mid a) = p_2(1 - p_3) + (1 - p_2)p_3 + p_2 p_3 = 0.3$.

Note that $\mathrm{P}(b \mid a) = p_2(1 - p_3) + (1 - p_2)p_3 + p_2 p_3$ exemplifies how $\mathrm{P}(b \mid a)$, which is a part of the semantics $\mathrm{P}(\mathbf{V})$, is a mathematical function of the numeric inputs $(p_1, p_2, p_3)$, which does not depend on the inputs having any meaning. Substituting the specific values we chose for $(p_1, p_2, p_3)$ happens to yield a proper joint distribution semantics $\mathrm{P}(\mathbf{V})$, despite $p_3 = -\frac{4}{3}$.

**Example 2.** It is possible to rewrite $R$ from Example 1, using auxiliary variables, so that the only non-deterministic rules are probabilistic facts:

$$\begin{aligned} R' = \{\ & p_1 :\ n_1, && a \longleftarrow n_1, && p_1 = 0.5 \\ & p_2 :\ n_2, && b \longleftarrow n_2, && p_2 = 0.7 \\ & p_3 :\ n_3, && b \longleftarrow n_3 \wedge a\ \} && p_3 = -\tfrac{4}{3} \end{aligned}$$

$R'$ is improper, since the joint distribution over its variables, $\mathrm{P}(N_1, N_2, N_3, A, B)$, is improper. This can easily be seen, by noting that $\mathrm{P}(n_3) = -\frac{4}{3} < 0$. However, if we ignore this fact, and marginalize out the noise variables $N_1$, $N_2$ and $N_3$ (which were added as auxiliary variables), we get the same proper distribution $\mathrm{P}(A, B, C)$ as in Example 1.

We call the set of PLPs with no negations and no negative probabilities **positive PLPs**.[1] We call the set of PLPs with no negations (but possibly with negative probabilities) **negative PLPs**. We call the set of PLPs with no negative probabilities but possibly with negations **PLPs-naf**. (We do not name the set of PLPs with both negations and negative probabilities.)

## 3. Motivating the need for a more expressive language

Consider the following relational cyclic positive PLP, loosely based on [15] and on the ProbLog tutorial[2]:

$0.3:\quad smokes(X)$

$0.1:\quad friends(X,Y)$

$0.9:\quad friends(X,Y) \longleftarrow friends(Y,X)$

$0.6:\quad susceptible(X)$

$0.2:\quad smokes(X) \longleftarrow susceptible(X) \wedge friends(X,Y) \wedge smokes(Y)$

$\quad\quad friends(chris, sam)$

According to this program, there is a baseline of 30% of people with no friends that smoke. Furthermore, 60% of people are "susceptible", and the chance of a susceptible person smoking increases with every smoking friend they have. If $X$ is susceptible, then every smoking friend has a probability of 20% of also causing $X$ to smoke. (The increase in P($smokes(X)$) is less than 20%, because even if $Y$ causes $X$ to smoke, $X$ may have picked up smoking for other reasons.)

Consider now wanting to model nonconformity instead of susceptibility. The probability a "nonconformist" person smokes increases with every *non-smoking* friend they have (they want to be different than their friends.) The straightforward way to change the program to model nonconformity instead of susceptibility gives the following PLP-naf program (changes are in bold):

$0.3:\quad smokes(X)$

$0.1:\quad friends(X,Y)$

$0.9:\quad friends(X,Y) \longleftarrow friends(Y,X)$

$0.6:\quad \boldsymbol{nonconformist(X)}$

$0.2:\quad smokes(X) \longleftarrow \boldsymbol{nonconformist(X)} \wedge friends(X,Y) \wedge \neg smokes(Y)$

$\quad\quad friends(chris, sam)$

Unfortunately, the program is not logically consistent, because the negation that was added appears inside a cycle:

$0.2:\quad smokes(chris) \longleftarrow nonconformist(chris) \wedge friends(chris, sam) \wedge \neg smokes(sam)$

$0.2:\quad smokes(sam) \longleftarrow nonconformist(sam) \wedge friends(sam, chris) \wedge \neg smokes(chris)$

This example demonstrates that:

1. The need of negations for expressiveness arises very naturally in real-world applications.
2. Unfortunately, negations may arise inside cycles, rendering the program logically inconsistent.
3. The example suggests (but not proves) that distributions of practical interest may not be representable by either positive PLPs nor valid PLPs-naf.
4. This raises the need for a more expressive language.

In Section 4 we prove that positive PLPs and valid PLPs-naf are indeed not fully expressive, and that negative PLPs can express distributions that positive PLPs and valid PLPs-naf cannot.

## 4. Relational PLPs for fixed populations

We now consider the simplest setting for a relational PLP with fixed populations: a single parametrized variable $A(X)$, no constants, and with a population of two: $\{x_1, x_2\}$. We mark $a(x_1)$, $a(x_2)$ as $a_1$, $a_2$. This simple setting suffices to show that:

[1] Technically, it should have been called "nonnegative PLPs"; however, rules with probability 0 have no effect.

[2] https://dtai.cs.kuleuven.be/problog/tutorial.html#tut_part1_smokers.

1. Positive PLPs are severely limited, and cannot represent all distributions.
2. Allowing negations does not increase expressiveness.
3. Allowing negative probabilities (with no negations) allows full expressiveness of non-extreme distributions.

This illustrates that relational programs without negative probabilities are limited in their expressiveness, and that negative probabilities may allow to represent distributions that are not otherwise expressible. Exploring the limits of the expressiveness of negative probabilities in the relational setting is left for future work.

### 4.1. Relational positive PLPs for fixed populations

The general form of a relational positive PLP with no constants, using only the variable $A(X)$ and with a population of two is:

$$\left\{ \; p_1 : \; a(X), \qquad p_2 : \; a(X) \longleftarrow a(Y) \; \right\} \tag{1}$$

which is grounded to the propositional program:

$$R_{\not\neg} = \left\{ \begin{array}{ll} p_1 : a_1, & p_2 : a_1 \longleftarrow a_2, \\ p_1 : a_2, & p_2 : a_2 \longleftarrow a_1 \end{array} \right\}$$

This program is a special case of Example 11 (in Section 8.1), with $p_3 = p_1$ and $p_4 = p_2$. Example 11 shows that if $p_1, p_2 \in [0, 1]$, then:

$$\mathrm{P}(a_1 \wedge a_2) \;\geq\; \mathrm{P}(a_1 \mid \neg a_2)\,\mathrm{P}(a_2 \mid \neg a_1) \tag{2}$$

Thus distributions that violate this constraint cannot be represented, e.g.:

$$\mathrm{P}(\neg a_1 \wedge \neg a_2) = 0.1 \qquad \mathrm{P}(\neg a_1 \wedge a_2) = 0.4$$
$$\mathrm{P}(\;\; a_1 \wedge \neg a_2) = 0.4 \qquad \mathrm{P}(\;\; a_1 \wedge a_2) = 0.1$$

### 4.2. Relational PLPs-naf for fixed populations

Allowing negations (but not negative probabilities), the general form of the relational PLP is:

$$\left\{ \begin{array}{l} p_1 : a(X), \\ p_2 : a(X) \longleftarrow a(Y), \\ p_3 : a(X) \longleftarrow \neg a(Y) \end{array} \right\}$$

which is grounded to the propositional program:

$$R_{\neg} = \left\{ \begin{array}{lll} p_1 : a_1, & p_2 : a_1 \longleftarrow a_2, & p_3 : a_1 \longleftarrow \neg a_2, \\ p_1 : a_2, & p_2 : a_2 \longleftarrow a_1, & p_3 : a_2 \longleftarrow \neg a_1 \end{array} \right\}$$

If $p_3 = 0$, then there are no negations. If $p_1 = 1$, then $\mathrm{P}(a_1 \wedge a_2) = 1$, and we need no negations to represent this distribution. If $p_1 < 1$, $p_3 > 0$ and $p_2 < 1$, then the following inconsistent DPR has a positive probability:

$$\left\{ \begin{array}{l} a_1 \longleftarrow \neg a_2, \\ a_2 \longleftarrow \neg a_1 \end{array} \right\}$$

And if $p_1 < 1$, $p_3 > 0$ and $p_2 = 1$, then the following inconsistent DPR has a positive probability:

$$\left\{ \begin{array}{ll} a_1 \longleftarrow a_2, & a_1 \longleftarrow \neg a_2, \\ a_2 \longleftarrow a_1, & a_2 \longleftarrow \neg a_1 \end{array} \right\}$$

Therefore, no matter what the parameters' values are, either the program's distribution can also be represented without negations, or the program has a positive probability of inconsistency and is thus invalid; thus for this setting, negations do not increase expressiveness.

The poor expressiveness is due to the PLP-naf being a grounding of a relational PLP-naf. A propositional PLP-naf with two variables can break the symmetry between the variables, and has six parameters instead of three. This additional flexibility allows it to represent any non-extreme distribution (Section 8.2).

### 4.3. Relational negative PLPs for fixed populations

We now allow $p_1 < 0$ and/or $p_2 < 0$ in $R_{\not\leftarrow}$. The program gives:

$$\begin{aligned} \mathrm{P}(\neg a_1 \wedge \neg a_2) &= (1-p_1)^2 \\ \mathrm{P}(a_1 \wedge \neg a_2),\ \ \mathrm{P}(\neg a_1 \wedge a_2) &= p_1(1-p_1)(1-p_2) \\ \mathrm{P}(a_1 \wedge a_2) &= p_1^2 + 2p_1(1-p_1)p_2 \end{aligned}$$

($\mathrm{P}(a_1 \wedge a_2)$ can be computed from the constraint that the probabilities sum to 1.)

Any non-extreme distribution (we actually only require $\mathrm{P}(\neg a_1 \wedge \neg a_2) > 0$) can thus be represented, by choosing

$$p_1 = 1 - \sqrt{\mathrm{P}(\neg a_1 \wedge \neg a_2)}, \qquad p_1 \in [0, 1) \tag{3}$$

for $p_1$ and then

$$\begin{aligned} p_2 &= 1 - \frac{\mathrm{P}(a_1 \wedge \neg a_2)}{p_1(1-p_1)} \\ &= 1 - \frac{\mathrm{P}(a_1 \wedge \neg a_2)}{\sqrt{\mathrm{P}(\neg a_1 \wedge \neg a_2)} - \mathrm{P}(\neg a_1 \wedge \neg a_2)} \end{aligned} \tag{4}$$

for $p_2$. When $p_1 = 0$, the value of $p_2$ has no effect, and we can choose $p_2 = 0$ to avoid a division by zero. The only distribution with $\mathrm{P}(\neg a_1 \wedge \neg a_2) = 0$ that can be represented is $\mathrm{P}(a_1 \wedge a_2) = 1$. It is represented by choosing $p_1 = 1$.

**Example 3.** Consider the distribution $\mathrm{P}'$:

$$\begin{aligned} \mathrm{P}'(\neg a_1 \wedge \neg a_2) &= \tfrac{1}{3} \\ \mathrm{P}'(a_1 \wedge \neg a_2),\ \ \mathrm{P}'(\neg a_1 \wedge a_2) &= \tfrac{1}{3} \\ \mathrm{P}'(a_1 \wedge a_2) &= 0 \end{aligned}$$

I.e., this distribution represents mutual exclusion, $\neg(a_1 \wedge a_2)$. This distribution violates (2), so it cannot be represented without negative probabilities. However, using (3) and (4), we can pick:

$$p_1 = 1 - \sqrt{\tfrac{1}{3}} = 0.4226$$

$$p_2 = 1 - \frac{\frac{1}{3}}{\sqrt{\frac{1}{3}} - \frac{1}{3}} = -0.3660$$

This gives a program with negative probabilities that represents $\mathrm{P}'$.

## 5. Relational PLPs with unbounded populations

We now analyze the setting, in which the populations' sizes are not fixed or bounded, and a PLP needs to be valid for all possible population sizes, i.e., be proper and have probability 0 of inconsistency. This is a more general setting than having fixed populations; however, this form of generalization makes this setting *more restrictive*, because a PLP must be valid for all possible groundings simultaneously, thus fewer PLPs can be used.

Using the less restrictive fixed populations setting, we already discovered the disappointing fact that both positive PLPs and PLPs-naf have limited expressiveness. We therefore now only consider negative PLPs, to see if their representative superiority carries over to the unbounded-populations setting. Unfortunately, this may not be the case. The following proposition shows that in our simple setting, having unbounded populations removes the additional expressiveness that negative probabilities brought.

**Proposition 1.** *If the program* $\{\ p_1 :\ a(X), \quad p_2 :\ a(X) \longleftarrow a(Y)\ \}$ *is proper for all possible populations, then it can also be expressed using nonnegative probabilities.*

The proof is in Appendix A.

Proposition 1 contrasts with Section 4.3, where we showed that programs with $p_2 < 0$ can express distributions that nonnegative programs cannot, when the population size is 2. Such programs, therefore, are not proper for some population size.

## 6. Probabilistic strengths

### 6.1. Summing evidence

Summing evidence refers to having multiple rules with a common head, in which case the probabilistic "evidence" must be summed up.

When all rules with a common head have disjoint bodies, the rule probabilities can be interpreted as conditional probabilities. When the bodies are not disjoint, their probabilistic influences must be combined.

**Example 4.** Let $R = \{\ p_1 :\ a,\ \ p_2 :\ h,\ \ p_3 :\ h \longleftarrow a\ \}$. Then $\mathrm{P}(h \mid a) = 1 - (1 - p_2)(1 - p_3)$.

### 6.2. Summing rules

We defined PLPs to be *multisets* of probabilistic rules. Some of our results, e.g., the definition of $R^T$ and Theorem 5 (and 6) are defined in terms of the unions of PLPs and rules, so they may give PLPs that have multiple rules with a common head *and* a common body. Summing rules refers to having multiple rules with a common head and a common body replaced with a single rule, without modifying the PLP's semantics.

When a program contains multiple rules with the same head and body, $p_1 :\ head \longleftarrow body$, $p_2 :\ head \longleftarrow body$, ..., they can be replaced with the single rule $\left(1 - \prod_i (1 - p_i)\right) :\ head \longleftarrow body$, without a change in semantics.

### 6.3. Strengths

The math of summing evidence and of summing rules is similar. However, the behavior of the rule probabilities is not very intuitive. We propose a different representation for probabilistic values, which we call "probabilistic strengths", that makes the math in both cases additive. This greatly simplifies our subsequent results. Furthermore, strengths also give intuition to the meaning of negative probabilities.

In probabilistic models, e.g., in graphical models, it is common to use the log-P representation instead of working directly with probabilities. For example, in Markov logic networks [15,5], formulae are annotated with "weights", which represent log-P values. The log-P representation sometimes simplifies the handling of such models, making combination of weighted formulae additive. $\sigma$ (strengths) is the PLP parallel to the log-P representation, making combination of probabilistic rules additive.

**Definition 1.** The **strength** $-\infty < \sigma \leq \infty$ of a probability $-\infty < p \leq 1$ is

$$\sigma \stackrel{\text{def}}{=} -\ln(1 - p).$$

Therefore, a strength $\sigma$ represents the probability:

$$p = 1 - e^{-\sigma}.$$

**Example 5.** $R$ from Example 4 can be represented as $R = \{\ \sigma_1 :\ a,\ \ \sigma_2 :\ h,\ \ \sigma_3 :\ h \longleftarrow a\ \}$. If we express $\mathrm{P}(h \mid a)$ as $\sigma(h \mid a)$, we get $\sigma(h \mid a) = -\ln(1 - \mathrm{P}(h \mid a)) = -\ln(1 - p_2)(1 - p_3) = -\ln\left(1 - (1 - e^{-\sigma_2})\right)\left(1 - (1 - e^{-\sigma_3})\right) = \sigma_2 + \sigma_3$, thus summing evidence becomes additive.

**Definition 2.** $\mathrm{sum}(R)$ is the program $R$ after rules with the same head and body are "summed up", i.e., replaced with a single rule whose strength is the sum of the strengths of the rules summed up.

**Proposition 2.** *Let $R$ be either a positive PLP, a negative PLP, a PLP-naf, or any PLP that does not simultaneously have negations in cycles and negative probabilities. Then* $\mathrm{sum}(R)$ *and $R$ have identical semantics.*

**Proof.** Let $R_2 = R' \cup \{\ \sigma_1 :\ h \longleftarrow body,\ \ \sigma_2 :\ h \longleftarrow body\ \}$ be an arbitrary program, and let $R_1 = R' \cup \{\ (\sigma_1 + \sigma_2) :\ h \longleftarrow body\ \}$. We now show that $R_1$ has the same semantics as $R_2$. Repeating the argument recursively proves the proposition.

The semantics of $R'$ is that of a distribution over $2^{|R'|}$ DPRs. The semantics of $R_2$ is a distribution over $2^{|R_2|} = 2^{|R'|+2}$ DPRs. For each DPR $D$ for $R'$ with probability $\mathrm{P}_{R'}(D)$, there are four corresponding DPRs for $R_2$, whose total probability sums to $\mathrm{P}_{R'}(D)$, and which are identical to $D$ with the addition of zero, one or two occurrences of the deterministic rule $h \longleftarrow body$. The probability of adding zero occurrences of $h \longleftarrow body$ is $(1 - p_1)(1 - p_2) = \left(1 - (1 - e^{-\sigma_1})\right)\left(1 - (1 - e^{-\sigma_2})\right) = e^{-\sigma_1} e^{-\sigma_2} = e^{-(\sigma_1 + \sigma_2)}$. With probability $1 - e^{-(\sigma_1 + \sigma_2)}$ we add either one or two occurrences of $h \longleftarrow body$. However, two occurrences of a deterministic rule are equivalent to a single occurrence, so the semantics of $R_2$ is identical to that of $R_1$, where the rule $h \longleftarrow body$ appears with probability $1 - e^{-(\sigma_1 + \sigma_2)}$. □

**Example 6.** Using the strengths notation, $\text{sum}(\{\ \frac{2}{3} :\ a,\ \ \frac{3}{4} :\ a\ \}) = \{\ 1\frac{5}{12} :\ a\ \}$.

When $R$ simultaneously contains negations inside cycles and negative probabilities, the semantics of $\text{sum}(R)$ and $R$ are still identical, except that it is also possible for $\text{sum}(R)$ to be strongly consistent while $R$ is not. This is defined and discussed in Section 9.1, and especially in Example 17.

Example 5 shows that the influence of rules can be combined with simple addition: Given that $body_1$ and $body_2$ are *true*, then $\sigma_1 :\ h \longleftarrow body_1$ and $\sigma_2 :\ h \longleftarrow body_2$ are equivalent to $\sigma_1 + \sigma_2 :\ h$. The same is true for multiple occurrences of the same rule: $\sigma_1 :\ r, \sigma_2 :\ r, \ldots$ can be replaced with $\left(\sum_i \sigma_i\right) :\ r$.

Unlike probabilities, "strengths" is interpretable: the strength of a rule represents the amount (or "weight") of information that it provides, which is to be added to the information provided by other rules.

The representation as strengths allows us to derive our main results in this paper much more easily, and our results become mathematically simpler and more intuitive.

A few intuitive properties of $\sigma$ are:

1. $\sigma$ is monotonically increasing with $p$.
2. $\sigma = 0$ corresponds to $p = 0$.
3. $\sigma > 0$ corresponds to $p > 0$.
4. $\sigma = \infty$ corresponds to $p = 1$ (deterministic rules.) This is intuitive, as a deterministic rule is equivalent to an infinite number of occurrences of a probabilistic rule.
5. $\sigma < 0$ corresponds to $p < 0$.
6. $p > 1$ cannot be represented using a strength $\sigma$.

### 6.4. Negative strengths

A probability $p < 0$ corresponds to a strength $\sigma < 0$. Using $\sigma$-notation, negative probabilities become intuitive: They allow to subtract the weights of evidence, or to subtract rule weights.

**Example 7.** Let $R$ contain a rule $\sigma :\ r$. Then adding the new rule $-\sigma :\ r$ to $R$ is equivalent to removing $\sigma :\ r$ from $R$, i.e., the new and old rules cancel out.

Without the representation as strengths, it is not immediately clear what negative probabilities mean. In particular, it is not trivial whether and how rules can cancel out.

**Example 8.** Let $R$ contain a rule $p :\ r$. Then adding the new rule $-\frac{p}{1-p} :\ r$ to $R$ is equivalent to removing $p :\ r$ from $R$, i.e., the new and old rules cancel out.

$\sigma = -\infty$, which corresponds to $\text{P} = -\infty$, is not allowed, because it makes the semantics ill defined (adding positive and negative infinities). Therefore, adding a new rule cannot cancel out a deterministic rule.

## 7. Representing CPDs inside acyclic propositional PLPs

This section refers only to propositional (non-relational) PLPs, even when this is not stated explicitly.

Acyclic propositional PLPs can be interpreted as Bayesian networks. The set of all rules that have a variable $H$ as their head can be seen as a specification of the conditional probability distribution (CPD) of $H$ given (a subset of) the variables that precede it in the ordering, called its "parents".

### 7.1. Acyclic propositional PLPs-naf can represent all CPDs

We represent an assignment to $A_1, \ldots, A_n$ using the set $\{a_i : A_i = \textit{true}\}$. Using negations, it is easy to represent any CPD $\text{P}(h \mid A_1, \ldots, A_n)$. The simplest approach is to create $2^n$ rules, one for each possible assignment $s$:

$$\text{P}(h \mid s) :\ h \longleftarrow \bigwedge_{a_i \in s} a_i \wedge \bigwedge_{a_j \notin s} \neg a_j$$

Using negations, however, CPD representation is not unique.

**Example 9.** The sets of rules $\{\ p :\ h \longleftarrow a,\ \ p :\ h \longleftarrow \neg a \wedge b\ \}$ and $\{\ p :\ h \longleftarrow b,\ \ p :\ h \longleftarrow \neg b \wedge a\ \}$ are equivalent, since both represent the disjunctive rule $p :\ h \longleftarrow a \vee b$.

**Algorithm 1** Convert CPD to rules.

Input: CPD $P(h \mid A_1, \ldots, A_n) < 1$, represented as $\sigma(h \mid s) < \infty$
Output: set of rules $R$ representing the CPD
$S \leftarrow$ set of all subsets of $\{a_1, \ldots, a_n\}$
$R \leftarrow \emptyset$
**while** $S \neq \emptyset$ **do**
  $s \leftarrow$ some minimal subset in $S$
    (i.e., $s \in S$, but no proper subsets of $s$ are in $S$)
  $\sigma_s \leftarrow \sigma(h \mid s) - \sum_{s' \subset s} \sigma_{s'}$
  $rule_s \leftarrow \left(\sigma_s : h \longleftarrow \bigwedge s\right)$
  $R \leftarrow R \cup \{rule_s\}$
  $S \leftarrow S \setminus s$
**end while**
**return** $R$

### 7.2. Acyclic propositional positive PLPs cannot represent all CPDs

Without negations (or negative probabilities), however, acyclic programs cannot represent some CPDs. For example, non-monotonic CPDs cannot be represented. A CPD is **monotonic** if changing some $A_i$ from *false* to *true* can only increase $P(h \mid A_1, \ldots, A_n)$.

**Example 10.** An positive acyclic PLP cannot represent the CPD $P(b \mid \neg a) = 0.7$, $P(b \mid a) = 0.3$.

The reason is that a rule $b \longleftarrow a \wedge (\ldots)$ increases $P(b \mid a)$ without changing $P(b \mid \neg a)$, and a rule $b \longleftarrow (\ldots)$ in which $a$ does not appear increases both $P(b \mid a)$ and $P(b \mid \neg a)$. Positive acyclic PLPs, therefore, cannot represent non-monotonic CPDs, where $P(b \mid a) < P(b \mid \neg a)$.

### 7.3. Representing CPDs without negations using negative PLPs

Negative rule probabilities in acyclic programs allow to express CPDs that cannot be otherwise expressed without negations. For example, the non-monotonic CPD $P(B \mid A)$ of Example 10 is represented in Example 1.

Given a set of positive literals $L$, we use $\bigwedge L$ for $\bigwedge_{l \in L} l$, and $\bigwedge \neg L$ for $\bigwedge_{l \in L} \neg l$.

The following theorem characterizes when using negative probabilities in negation-free rules defines a proper CPD.

**Theorem 1.** *Consider a set $R$ of negation-free probabilistic rules (possibly with negative probabilities), in which the head is $h$ and the bodies are conjunctions of subsets of $\{a_1, \ldots, a_n\}$. Let $\sigma_s$ be the probabilistic strength of the rule $h \longleftarrow \bigwedge s$, where $s \subseteq \{a_1, \ldots, a_n\}$. Then $R$ represents a "proper" CPD (i.e., $\forall s$, $P(h \mid s) \in [0, 1]$) if and only if:*

$$\forall s \subseteq \{a_1, \ldots, a_n\}, \ \sum_{s' \subseteq s} \sigma_{s'} \geq 0.$$

**Proof.** Given an assignment $s$, $P(h \mid s)$ is determined by the sum of the influences of all rules whose body is *true*. The body of $h \longleftarrow \bigwedge s'$ is *true* if and only if $s' \subseteq s$, so the combination of the rules' influences gives $\sum_{s' \subseteq s} \sigma_{s'}$. If $\sum_{s' \subseteq s} \sigma_{s'} < 0$ then $P(h \mid s) < 0$, but if $\sum_{s' \subseteq s} \sigma_{s'} \geq 0$, then $P(h \mid s) \in [0, 1]$. □

**Theorem 2.** *Using negative rule probabilities, any CPD $P(h \mid A_1, \ldots, A_n) < 1$ can be expressed without negations.*

**Proof.** $P(h \mid A_1, \ldots, A_n)$ can be specified using $2^n$ probabilities, each specifying $P(h \mid s)$ for another assignment $s$. We represent each $P(h \mid s) < 1$ as $\sigma(h \mid s) < \infty$. The theorem then follows from the correctness of Algorithm 1. □

**Proof of Correctness of Algorithm 1.** The algorithm creates $2^n$ rules, each corresponding to a subset of $\{a_1, \ldots, a_n\}$. Given a joint assignment represented by $s$, the combination of the rules' influences on $h$ is $\sum_{s' \subseteq s} \sigma_{s'}$, as shown in the proof for Theorem 1. Among these $2^{|s|}$ rules, the algorithm creates $rule_s$ last, and sets $\sigma_s$ in a way that assures that $\sum_{s' \subseteq s} \sigma_{s'} = \sigma(h \mid s)$, thus the CPD is correctly represented. □

#### 7.3.1. Sparsity

If the algorithm creates rules for which $\sigma_s = 0$, these rules have no effect and can be pruned. Therefore, when the CPD has structure, the algorithm may create significantly fewer than $2^n$ effective rules. For example, if $H$ never depends on some $A_i$, then all rules whose body contains $a_i$ end up having $\sigma = 0$ and can be pruned. In the case of noisy-or, there are only $n$ rules left.

### 7.4. Canonical CPD form

Unlike PLPs-naf, negative PLPs represent each CPD in a unique way.

**Corollary 1.** *Algorithm 1 provides a* canonical form *for representing CPDs without negations.*
*It is the only form a CPD can be represented without negations (without having multiple occurrences of the same rule).*

Note the clear mathematical similarity of this canonical form to the "canonical parametrization" for undirected discrete probabilistic graphical models [9,11,2].

## 8. General (cyclic) propositional PLPs

This section refers only to propositional (non-relational) PLPs, even when this is not stated explicitly.

### 8.1. Propositional positive PLPs cannot represent all distributions

In Section 7.2 we showed acyclic propositional positive PLPs cannot represent all CPDs. We now show general propositional positive PLPs cannot represent all distributions. Characterizing which distributions can be expressed is complicated. (The motivation for avoiding negations is that negations may create logical inconsistencies in cyclic PLPs.)

**Example 11.** Consider a general-form positive PLP with two variables:

$$R = \{\ p_1 : a, \qquad p_2 : a \longleftarrow b,$$
$$p_3 : b, \qquad p_4 : b \longleftarrow a\ \}$$
$$p_1, p_2, p_3, p_4 \in [0, 1]$$

Then, assuming $\mathrm{P}(\neg a \wedge \neg b) > 0$:

$$\frac{\mathrm{P}(\ \ a \wedge \neg b)}{\mathrm{P}(\neg a \wedge \neg b)} = \frac{p_1(1-p_3)(1-p_4)}{(1-p_1)(1-p_3)} \leq \frac{p_1}{1-p_1}$$
$$(1-p_1)\,\mathrm{P}(a \wedge \neg b) \leq p_1\,\mathrm{P}(\neg a \wedge \neg b)$$
$$p_1 \geq \frac{\mathrm{P}(a \wedge \neg b)}{\mathrm{P}(a \wedge \neg b) + \mathrm{P}(\neg a \wedge \neg b)}$$
$$= \mathrm{P}(a \mid \neg b)$$

And similarly:

$$p_3 \geq \mathrm{P}(b \mid \neg a).$$

Also,

$$\mathrm{P}(a \wedge b) = p_1 p_3 + p_1(1-p_3)p_4 + (1-p_1)p_2 p_3$$

thus if the rule probabilities $p_1$, $p_2$, $p_3$, $p_4$ are nonnegative, then:

$$\mathrm{P}(a \wedge b) \geq p_1 p_3$$

therefore:

$$\mathrm{P}(a \wedge b) \geq \mathrm{P}(a \mid \neg b)\,\mathrm{P}(b \mid \neg a)$$

This inequality rules out representing some distributions. For instance, for the distribution

$$\mathrm{P}(\neg a \wedge \neg b) = 0.1 \qquad \mathrm{P}(\neg a \wedge b) = 0.4$$
$$\mathrm{P}(\ \ a \wedge \neg b) = 0.4 \qquad \mathrm{P}(\ \ a \wedge b) = 0.1$$

the inequality implies $0.1 = \mathrm{P}(a \wedge b) \geq \mathrm{P}(a \mid \neg b)\,\mathrm{P}(b \mid \neg a) = 0.8 \cdot 0.8 = 0.64$, so it cannot be represented by a positive PLP.

*8.2. Propositional PLPs-naf and negative PLPs can represent all non-extreme distributions*

**Proposition 3.** *Propositional negative PLPs can represent any non-extreme joint distribution.*

**Proof.** Define an arbitrary ordering among the variables. The joint distribution can be written as $\mathrm{P}(V_1,\ldots,V_n)=\prod_{i=1}^{n}\mathrm{P}(V_i \mid V_1,\ldots,V_{i-1})$. According to Theorem 2, each CPD can be expressed using probabilistic rules without negations. Since this creates an acyclic program, the joint distribution is the product of the CPDs. □

By the same argument, PLPs-naf can also represent all non-extreme distributions.

*8.3. Unintuitive properties of cyclic programs*

The first part of this section might be surprising to readers less experienced with cyclic PLPs. The second part investigates the extension to negative probabilities, giving a surprising result. The overall goal of this section is to provide insight and intuition, which may help the reader understand the intricacies of proofs in Section 9.

*8.3.1. Complex rule interactions*

Given a program $R$, we use $\mathrm{P}_R(\cdots)$ for its joint distribution, and $R_h$ for the set of all rules whose head is $h$. $R_h$ can be seen as defining a CPD $\mathrm{P}_{R_h}(h \mid \text{inputs})$. For acyclic programs $R$, their joint distribution $\mathrm{P}_R(\cdots)$ reflects this CPD, i.e., $\mathrm{P}_R(h \mid \text{inputs})=\mathrm{P}_{R_h}(h \mid \text{inputs})$. For cyclic programs, this is not the case.

**Example 12.** Consider $R$ of Example 11 with:

$$p_1=p_3=\tfrac{1}{3} \qquad p_2=p_4=\tfrac{1}{4}$$

$R$ defines a CPD $\mathrm{P}_{R_a}(A \mid B)$ for which $\mathrm{P}_{R_a}(a \mid \neg b)=p_1=\frac{1}{3}$ and $\mathrm{P}_{R_a}(a \mid b)=1-(1-p_1)(1-p_2)=\frac{1}{2}$. Similarly, $R$ also defines a CPD $\mathrm{P}_{R_b}(B \mid A)$ for which $\mathrm{P}_{R_b}(b \mid \neg a)=\frac{1}{3}$ and $\mathrm{P}_{R_b}(b \mid a)=\frac{1}{2}$. There is a joint distribution that matches $\mathrm{P}_{R_a}(B \mid A)$ and $\mathrm{P}_{R_b}(B \mid A)$, and it is:

$$\mathrm{P}(\neg a \wedge \neg b)=0.4 \qquad \mathrm{P}(\neg a \wedge b)=0.2$$
$$\mathrm{P}(\ \ a \wedge \neg b)=0.2 \qquad \mathrm{P}(\ \ a \wedge b)=0.2$$

However, $R$ actually represents:

$$\mathrm{P}_R(\neg a \wedge \neg b)=\tfrac{4}{9}=0.444 \qquad \mathrm{P}_R(\neg a \wedge b)=\tfrac{1}{6}=0.167$$
$$\mathrm{P}_R(\ \ a \wedge \neg b)=\tfrac{1}{6}=0.167 \qquad \mathrm{P}_R(\ \ a \wedge b)=\tfrac{2}{9}=0.222$$

thus $\mathrm{P}_R(A \mid B)\neq\mathrm{P}_{R_a}(A \mid B)$ and $\mathrm{P}_R(B \mid A)\neq\mathrm{P}_{R_b}(B \mid A)$, so the cyclic program represents CPDs that are *different* than those described by its rules in isolation.

*8.3.2. Complex rule interactions with negative probabilities*

Furthermore, with negative rule probabilities, $\mathrm{P}_R(\mathbf{V})$ may even be improper, even when all CPDs $\mathrm{P}_{R_h}(h \mid \text{inputs})$ are proper and a proper joint distribution exists that matches these CPDs.

**Example 13.** Consider $R$ of Example 11 with:

$$p_1=p_3=0.7 \qquad p_2=p_4=-\tfrac{4}{3}$$

where $\mathrm{P}_{R_a}(a \mid \neg b)=0.7$, $\mathrm{P}_{R_a}(a \mid b)=0.3$, $\mathrm{P}_{R_b}(b \mid \neg a)=0.7$ and $\mathrm{P}_{R_b}(b \mid a)=0.3$. These CPDs are proper, and there is a proper joint distribution that matches these CPDs:

$$\mathrm{P}(\neg a \wedge \neg b)=0.15 \qquad \mathrm{P}(\neg a \wedge b)=0.35$$
$$\mathrm{P}(\ \ a \wedge \neg b)=0.35 \qquad \mathrm{P}(\ \ a \wedge b)=0.15$$

However, $R$ actually represents a different joint distribution, which is improper, since $\mathrm{P}_R(a,b)=p_1p_3+p_1(1-p_3)p_4+(1-p_1)p_3p_2=-\frac{1}{12}<0$.

## 9. Compact expressiveness of propositional negative PLPs

This section refers only to propositional (non-relational) PLPs, even when this is not stated explicitly.

This section is the most mathematically involved section in this paper. Its main goal is to define the translation of PLPs-naf to negative PLPs and to prove this translation preserves the semantics and does not blow up the PLP's size. This is achieved in Theorem 6, at the very end of this section. The rest of this section builds up gradually towards Theorem 6.

### 9.1. Strong consistency

When PLPs contain negations inside cycles, there is a non-zero probability of logical inconsistency. When PLPs simultaneously contain negations inside cycles and negative probabilities, there is also a possibility of having multiple inconsistent DPRs, some with positive probability and some with negative probability, while their total probability is 0. In other words, it is possible to have probability 0 of inconsistency, while having positive probability for some inconsistent DPRs.

**Definition 3.** A program $R$ is **strongly consistent** if any logically inconsistent DPR of $R$ has probability 0.

Thus P(inconsistency) $= 0$ is a necessary condition for strong consistency, but it is not sufficient for PLPs that simultaneously contain negations inside cycles and negative probabilities. A PLP-naf program has no negative probabilities, and is thus strongly consistent if and only if it is logically inconsistent with probability 0. A deterministic program is strongly consistent if and only if it is logically consistent.

**Example 14.** Consider the program $R$:

$$\begin{aligned} R = \{ \quad & a \longleftarrow b, \\ & \sigma : b \longleftarrow \neg a, \\ & -\sigma : b \longleftarrow \neg a \wedge \neg c \,\}, \qquad \sigma > 0 \end{aligned}$$

$\sigma$ represents the probability $1-e^{-\sigma}$, and $-\sigma$ represents the probability $1-e^{-(-\sigma)}$. The DPR $\{a \longleftarrow b\}$ is logically consistent, and its probability is $(1-(1-e^{-\sigma}))(1-(1-e^{-(-\sigma)})) = 1$. The other three DPRs are logically inconsistent, and their probabilities are $e^{\sigma}-1>0$, $e^{-\sigma}-1<0$ and $2-e^{\sigma}-e^{-\sigma}<0$, which sum to 0. The total probability of logical inconsistency is 0, but $R$ is not strongly consistent.

We mark $R_1 \equiv R_2$ if programs $R_1$ and $R_2$ are both strongly consistent and represent the same (proper or improper) joint distribution. Note that if $R_1 \equiv R_2$, then $R_1$ is proper if and only if $R_2$ is proper. Also note that $R_1 \equiv R_2$ does not imply $R \cup R_1 \equiv R \cup R_2$.

**Example 15.** $\{\ b \longleftarrow a\ \} \equiv \emptyset$, because $M(\{\ b \longleftarrow a\ \}) = \emptyset = M(\emptyset)$.
However, if we add the rule $a$ to both, we get non-equal programs: $\{\ b \longleftarrow a\ \} \cup \{\ a\ \} \not\equiv \emptyset \cup \{\ a\ \}$. This is because $M(\{\ b \longleftarrow a\ \} \cup \{\ a\ \}) = \{a,b\} \neq \{a\} = M(\emptyset \cup \{\ a\ \})$.

Note that a proper program may have an improper distribution over DPRs.

**Example 16.** $R = \{\ p:\ b \longleftarrow a\ \}$, $p = -2$ has an improper distribution over its two DPRs: P($\{\ b \longleftarrow a\ \}$) $= -2$, P($\emptyset$) $= 3$. However, $R$ is proper, since its distribution over models is nonnegative:

$$\begin{aligned} &\mathrm{P}(\neg a \wedge \neg b) = 1 \qquad &&\mathrm{P}(\neg a \wedge b) = 0 \\ &\mathrm{P}(\ \ a \wedge \neg b) = 0 \qquad &&\mathrm{P}(\ \ a \wedge b) = 0 \end{aligned}$$

**Example 17.** $R = \{\ a \longleftarrow b,\ \ \sigma:\ b \longleftarrow \neg a,\ \ -\sigma:\ b \longleftarrow \neg a\ \}$ is not strongly consistent. $R$ has three logically inconsistent DPRs; the sum of their probabilities, however, is 0.
Summing up $R$'s rules gives the equivalent sum$(R) = \{\ a \longleftarrow b\ \}$ which is strongly consistent and proper. Summing up rules, therefore, may remove inconsistent DPRs that cancel each other out in probability.

### 9.2. Consistency-maintaining orderings (CMOs)

The results in this section are needed for proving Theorem 6 in the next section.

**Definition 4.** A **consistency-maintaining ordering (CMO)** for a program $R$ is an ordering $(r_1, r_2, \ldots)$ of $R$'s rules, such that for all $i$, the **sub-program** $R_i \stackrel{\text{def}}{=} \{r_1, r_2, \ldots, r_i\}$ is strongly consistent.

**Example 18.** $(b,\ a \longleftarrow b,\ b \longleftarrow \neg a)$ is a CMO for $R = \{a \longleftarrow b,\ b \longleftarrow \neg a,\ b\}$, but $(a \longleftarrow b,\ b \longleftarrow \neg a,\ b)$ is not a CMO.

**Theorem 3.** *Every logically consistent propositional deterministic program has a CMO.*

**Algorithm 2** Find CMO.

Input: A logically consistent propositional deterministic program $R$
Output: A CMO for $R$
*ordering* $\leftarrow ()$
**while** $\exists r \in R \setminus$ *ordering*, such that:
**(1)** all positive literals in $r$'s body are heads of rules in *ordering*, and
**(2)** all negative literals in $r$'s body are not in $M(R)$ **do**
append( *ordering*, $r$ )
**end while**
append( *ordering*, remaining rules) *(in arbitrary order)*
**return** *ordering*

**Proof.** We show Algorithm 2 produces a CMO.

By induction, during the while loop, each rule $r_i$ added has a body which is evaluated to *true*, and its head is in $M(R)$, and adding $r_i$ does not make the bodies of previous rules *false*. Therefore, during the while loop, each sub-program $R_i$ is consistent, and $M(R_i)$ is the set of the literals in the heads of its rules.

After the while loop, each remaining rule $r_i$ being added has a body which is evaluated to *false*, and therefore does not change the program semantics, i.e., $R_i \equiv R_{i-1}$. □

For negation-free programs, the algorithm can be simplified, by dropping condition (2). Therefore the algorithm does not need to assume $R$ is consistent. The correctness of the algorithm thus also proves that all positive deterministic programs are consistent and have a CMO.

We extend Theorem 3 to probabilistic programs:

**Theorem 4.** *Every strongly consistent propositional program $R$ has a CMO.*

**Proof.** Let $D \subseteq R$ be the subset of deterministic rules in $R$. $R$ is strongly consistent, and $D$ is a non-zero-probability DPR of $R$, so $D$ is consistent. By Theorem 3, $D$ has a CMO. We define $R$'s ordering as $D$'s ordering, followed by all probabilistic rules at an arbitrary order.

Consider some $R_i$. If $R_i$ only contains deterministic rules, then it is strongly consistent because it is part of $D$'s CMO. Otherwise, each non-zero-probability DPR $D'$ of $R_i$ is also a DPR of $R$ (where all rules in $R \setminus R_i$ are not used.) $R$ is strongly consistent, so all such $D'$'s are consistent, therefore $R_i$ is strongly consistent. □

### 9.3. Translations

With negations, cyclic programs may become logically inconsistent, when there is a negation in a rule cycle. However, this still leaves a wide array of possible logically consistent programs, that are expressive enough to be useful. For example, consider programs composed of negation-free cyclic components, and an acyclic structure containing negations that connects the components.

Section 8.2 showed that negative PLPs can represent all (non-extreme) distributions. Negations are therefore not essential for representing non-extreme joint distributions. However, negative PLPs would not be an attractive alternative to PLPs-naf, if they required an exponential number of rules to represent compact PLPs-naf. To better motivate using negative probabilities instead of negations, we show this is not the case, by showing a translation from PLPs-naf to negative PLPs, that does not blow up exponentially.

**Definition 5.** An arbitrary propositional rule

$$r \quad = \quad \sigma : h \longleftarrow \bigwedge r^+ \wedge \bigwedge \neg r^-$$

is called **translatable** if $\sigma < \infty$ or $r^- = \emptyset$.

If $r$ is translatable, then the **translated rule** $r^T$ is a set of $2^{|r^-|}$ negation-free rules:

$$r^T \stackrel{\text{def}}{=} \left\{ (-1)^{|L|}\sigma : h \longleftarrow \bigwedge r^+ \wedge \bigwedge L \quad : \quad L \subseteq r^- \right\}$$

**Definition 6.** If all rules in a propositional program $R$ are translatable, then $R$ is called **translatable**, and the **translated program** $R^T$ is:

$$R^T \stackrel{\text{def}}{=} \text{sum}\left(\bigcup_{r \in R} r^T\right)$$

In other words, programs are translatable if deterministic rules have no negations.

**Theorem 5.** *Let $R$ be a set of negation-free propositional rules, and $r$ be a translatable rule $\sigma : h \longleftarrow \bigwedge r^+ \wedge \bigwedge \neg r^-$. If $R \cup \{r\}$ is strongly consistent, then $R \cup \{r\} \equiv R \cup r^T$.*

The proof is in Appendix A. It is complex because $R$ is cyclic, so adding rules to $R$ may create complex "interactions" with rules in $R$.

Given an existing program with negations in bodies of non-deterministic rules, we now show how one can construct an equivalent negation-free program, using negative probabilities.

**Theorem 6.** *If $R$ is a strongly consistent translatable propositional program, then $R^T$ is a strongly consistent negation-free (propositional) program and $R^T \equiv R$.*

*Furthermore, $|R^T| \leq \sum_i 2^{k_i} \leq 2^k |R|$, where $k_i$ is the number of negations in rule $r_i$, and $k = \max_i k_i$.*

**Proof.** $R$ is strongly consistent, so by Theorem 4 it has a CMO. $R$ can be thus formed by iteratively adding rules, while maintaining strong consistency throughout. For every rule $r_i$ with $k_i$ negations being added, since $r_i$ is translatable, Theorem 5 gives a set of $2^{k_i}$ negation-free rules, such that adding them instead of $r_i$ gives a strongly consistent program with the same semantics. Repeating this replacement for all rules, we get a negation-free program with the same semantics, with $|\text{sum}(\bigcup_{r \in R} r^T)| \leq \sum_i 2^{k_i} \leq 2^k |R|$ rules. □

## 10. Compact expressiveness of relational negative PLPs with fixed populations

Theorem 6, the main result of the previous section, can be extended to relational PLPs with fixed populations. In this section we give an outline of this extension.

The basic insight as to why Theorem 6 holds in the relational case is as follows. Given fixed populations, a relational PLP $R$ can be seen as a compact description of a ground (propositional) PLP $R_{ground}$, with specific symmetries among the rule probabilities, i.e., some rule probabilities are equal. For example, if $x_7$ and $x_8$ are individuals in the population that are not assigned constants, and the rules $a(x_7) \longleftarrow b$ and $a(x_8) \longleftarrow b$ appear in $R_{ground}$, then their probabilities are equal. This is because every rule in $R$ which involves $a(X)$, is grounded in $R_{ground}$ to rules involving $a(x_7)$ and $a(x_8)$, with equal probabilities. We can apply Theorem 6 to $R_{ground}$, and find a ground (propositional) translated PLP ${R_{ground}}^T$ such that ${R_{ground}}^T \equiv R_{ground}$. Since the translation operation does not break the symmetries (equalities among rule probabilities) implied by the relational PLP $R$, these also hold for ${R_{ground}}^T$; thus ${R_{ground}}^T$ can be described using a more compact, relational, PLP, using first-order rules.

Performing the translation on the ground model gives insight as to *why* Theorem 6 can be extended to the relational case, but it is not efficient. It is much more efficient to perform a **lifted translation**, i.e., to perform the translation directly on the relational model.

To translate a PLP liftedly, one must define how a first-order rule is translated. Let $R$ contain a single rule, the first-order rule $r = \sigma : h \longleftarrow \neg a(X)$. Let the population be $x_1, \ldots, x_n$. $r$ is "instantiated" $n$ times in $R_{ground}$, to $r_{ground,1}, \ldots, r_{ground,n}$. Each grounded rule $r_{ground,i} = \left(\sigma : h \longleftarrow \neg a(x_i)\right)$ is translated into

$$ {r_{ground,i}}^T = \left\{ \ \sigma : h, \ \ -\sigma : h \longleftarrow a(x_i) \ \right\} $$

The rule $\sigma : h$ is added $n$ times to ${R_{ground}}^T$, once for each ground rule being translated. Therefore,

$$ {R_{ground}}^T = \left\{ n\sigma : h \right\} \cup \left\{ -\sigma : h \longleftarrow a(x_i) \quad : \quad i \in \{1, \ldots, n\} \right\} $$

thus the lifted translation of the first-order rule $r$ is

$$ r^T = \left\{ \ n\sigma : h, \ \ -\sigma : h \longleftarrow a(x) \ \right\} $$

This lifted translation is very similar to the translation of propositional rules, only here the weight of the rule $h$ was also multiplied by $n$, the population's size. In general, the definition of rule translation can be extended to first-order rules, by multiplying by (powers of) population sizes.

Our results, generalized to the relational setting:

**Theorem 7.** *Given fixed populations, if $R$ is a strongly consistent translatable (relational) PLP, then $R^T$ is a strongly consistent negation-free program and $R^T \equiv R$.*

*Furthermore, $|R^T| \leq \sum_i 2^{k_i} \leq 2^k |R|$, where $k_i$ is the number of negations in (first-order) rule $r_i$, and $k = \max_i k_i$.*

**Corollary 2.** *Given fixed populations, if $R$ is a (relational) PLP-naf which is consistent with probability 1 and has no negations in deterministic rules (only in probabilistic ones), then $R$ can be converted to an equivalent negation-free (relational) program $R^T$, in a lifted manner.*

## 11. Conclusions

PLPs with nonnegative probabilities and without negations are lacking in the distributions they can represent. Allowing negations solves this problem in the propositional setting, but not in the relational setting. Unfortunately, a simple, intuitive real-world relational example suggests that the lack of PLP representational power comes up not only in esoteric cases, but also in real-world applications. Being aware of representational shortcomings of the language is crucial for users, to prevent wasted efforts trying to learn or represent distributions that the language cannot represent.

We introduced negative rule probabilities, and suggest using them in lieu of negations.

In the propositional setting, negative rule probabilities can represent any non-extreme distribution that can be represented using negations. Furthermore, a translation algorithm proves that the size of a program with negative probabilities is larger by at most a factor of $2^k$ than an equivalent program using negations, where $k$ is the maximal number of negations per rule, which is frequently a small number.

We analyzed the relational setting, when the populations are given (fixed) (the populations may be much larger than the set of constants available). The translation algorithm can be extended to first-order (relational) PLPs for this setting, and done in an efficient lifted manner (without grounding), while maintaining the PLP's semantics and limiting the growth in the PLP's size. Furthermore, in this setting, using negative probabilities is not only not inferior to using negations, but negative probabilities are also capable of representing distributions that cannot be represented otherwise, thus are more expressive languages.

In addition, for acyclic programs, we completely characterized the conditions for avoiding probabilistic improperness using negative probabilities, and showed how any non-extreme CPD can be expressed.

We also highlighted some of the unintuitive properties of general (cyclic) programs, and highlighted consistency-maintaining orderings.

Many of our results involve mathematical analysis that involves summing evidence or summing recurring rules. We presented *probabilistic strengths*, a new representation for probabilities in PLPs. Strengths provide deeper insights into PLPs, where rule algebra becomes additive; they are the PLP equivalent of the log-P notation of graphical models such as Markov logic networks. Strengths also make the concept of negative probabilities simpler and more intuitive.

As a result of the translation algorithm, *exact inference* in logically consistent probabilistic cyclic programs containing negations can be carried out by translating the program to a negation-free program using negative probabilities, and employing exact inference algorithms that do not support negations. In other words, our work allows to extend the scope of exact inference algorithms that do not support negations, to be applicable to programs with negations.

One major open problem is characterizing conditions on the rule probabilities of general (cyclic) PLPs for ensuring properness.

Another open problem is finding useful extensions to the language that can increase the expressiveness of PLPs in the relational case, when population sizes are *not bounded*.

Another open problem is how *approximate inference* can be efficiently carried out in cyclic probabilistic programs with negative probabilities. There is no guarantee that an approximate computation will remain a good approximation when some probabilities are negative.

## Acknowledgements

This work was supported by an NSERC operating grant to David Poole.

## Appendix A

### A.1. Proof of Proposition 1

**Proof.** Let the population size be $n$, let us mark the variables $\{a(X)\}$ by $a_1, \ldots, a_n$, and let $\mathrm{P}_{n_t,n_f}$ be the probability of any specific model where $n_t$ of the variables are *true* and $n_f$ are *false*, for $n = n_t + n_f$. (The probabilities of all such models are equal, due to symmetry.)

Let $n = m + 1$, and consider the model $\left(\bigwedge_{i=1}^{m} a_i\right) \wedge \neg a_{m+1}$. The probability of this model is $\mathrm{P}_{m,1}$. Let us analyze $\mathrm{P}_{m,1}$ as having two components:

1. First, some combination of rules must simultaneously appear, to set $a_1, \ldots, a_m$ to *true*. Since $a_{m+1}$ is *false*, rules involving $a_{m+1}$ are irrelevant to this analysis. Therefore, the probability of setting $a_1, \ldots, a_m$ to *true* is equal to $\mathrm{P}_{m,0}$, which is the probability of all $m$ variables being *true* in a different grounded PLP which only has $m$ ground variables.
2. Second, some rules must not appear, to ensure $a_{m+1}$ is not set to *true*. For this, the rules $a_{m+1}$ and $a_{m+1} \longleftarrow a_1, \ldots, a_{m+1} \longleftarrow a_m$ must not appear. The probability of these rules not appearing is $(1 - p_1)(1 - p_2)^m$.

We thus received

$$\mathrm{P}_{m,1} = (1 - p_1)(1 - p_2)^m \, \mathrm{P}_{m,0}$$

This can be generalized to:

$$A(n_t) \stackrel{\text{def}}{=} (1-p_1)(1-p_2)^{n_t} \quad > 0$$
$$\mathrm{P}_{n_t,n_f} = (1-p_1)^{n_f}(1-p_2)^{n_t n_f}\,\mathrm{P}_{n_t,0} \;=\; A(n_t)^{n_f}\,\mathrm{P}_{n_t,0} \tag{5}$$

We require the PLP to be proper, i.e., $\mathrm{P}_{n_t,n_f} \in [0,1]$ for all $n_t$ and $n_f$. Since $\mathrm{P}_{1,0} = p_1$, we get $p_1 \in [0,1]$. For the program not to be expressible using nonnegative probabilities, we must pick $p_2 < 0$. We must also pick $p_1 \in (0,1)$, because for $p_1 \in \{0,1\}$, $p_2$ has no effect and can be set to 0.

In general, for any $n$:

$$1 = \sum_{\text{model}} \mathrm{P}_n(\text{model}) = \sum_{n_t=0}^{n} \binom{n}{n_t} \mathrm{P}_{n_t,n-n_t} = \sum_{n_t=0}^{n} \binom{n}{n_t} A(n_t)^{n-n_t}\,\mathrm{P}_{n_t,0}$$

**Case 1: $\exists n_t'$, $\left(\mathrm{P}_{n_t',0} > 0 \text{ and } A(n_t') \geq 1\right)$.** If the distribution is proper for all $n$'s, then for any $n \geq n_t'$:

$$1 \geq \binom{n}{n_t'} A(n_t')^{n-n_t'}\,\mathrm{P}_{n_t',0} \geq \binom{n}{n_t'}\mathrm{P}_{n_t',0}$$

For large-enough $n$'s, the RHS becomes greater than 1; thus, the program is not proper for all $n$'s.

**Case 2: $\forall n_t$, $\left(\mathrm{P}_{n_t,0} = 0 \text{ or } A(n_t) < 1\right)$.** Since $(1-p_1) \in (0,1)$ and $(1-p_2) > 1$, an integer threshold $n_t^*$ exists such that:

$$\forall n_t < n_t^*, \quad A(n_t) < 1$$
$$\forall n_t \geq n_t^*, \quad A(n_t) \geq 1$$

Therefore $\mathrm{P}_{n_t,0} = 0$ for all $n_t \geq n_t^*$, so:

$$1 = \sum_{n_t=0}^{n_t^*-1} \binom{n}{n_t} A(n_t)^{n-n_t}\,\mathrm{P}_{n_t,0} \leq \sum_{n_t=0}^{n_t^*-1} \binom{n}{n_t} A(n_t)^{n-n_t} \leq \sum_{n_t=0}^{n_t^*-1} n^{n_t} A(n_t)^{n-n_t}$$

Taking the limit:

$$1 \leq \lim_{n\to\infty} \sum_{n_t=0}^{n_t^*-1} n^{n_t} A(n_t)^{n-n_t} = \sum_{n_t=0}^{n_t^*-1} \lim_{n\to\infty} n^{n_t} A(n_t)^{n-n_t} = 0$$

Therefore, this case cannot happen. □

### A.2. Proof of Theorem 5

**Proof.** First, consider the case where $R$ is deterministic. Since $R$ is deterministic and negation-free, it has a unique stable model $M(R)$.

If $\sigma = \infty$, then since $r$ is translatable, $r$ must be negation-free, so $r^T = \{r\}$.

If $h \in M(R)$, then adding $r$ to $R$ does not change $R$'s semantics. Adding $r^T$ to $R$ also does not change $R$'s semantics, thus $R \cup \{r\} \equiv R \cup r^T$.

Similarly, if $r^+ \not\subseteq M(R)$, or if $\sigma = 0$, then neither adding $r$ or $r^T$ to $R$ has any effect.

We can now assume that $r^+ \subseteq M(R)$, $h \notin M(R)$, $\sigma \neq 0$, and $\sigma < \infty$.

(Note that $r^T$ contains $2^{|r^-|}$ rules, and the semantics of $R \cup r^T$ is that of the (possibly improper) distribution over $2^{2^{|r^-|}}$ deterministic programs.)

$$\text{Define:} \quad r^{T\prime} \stackrel{\text{def}}{=} \left\{ (-1)^{|L|}\sigma : h \longleftarrow \bigwedge r^+ \wedge \bigwedge L \right.$$
$$\left. : \quad L \subseteq r^- \cap M(R) \right\} \quad \subseteq \quad r^T$$

Since $R \cup r^T$ and $R \cup r^{T\prime}$ are negation-free, they are strongly consistent (because all their DPRs are logically consistent.) We now show that $R \cup r^{T\prime} \equiv R \cup r^T$. If $r^- \cap M(R) = r^-$, this is trivial. Otherwise, let $l \in r^- \setminus M(R)$. For every $L \subseteq r^-$ that contains $l$, adding the corresponding rule $r_L$ to $R$ never changes the semantics of the program. This is because $r_L$'s only effect is to set $H$ to *true*; however, since $l$ is *false*, $r_L$'s body is *false*. The only way for $r_L$'s body to be *true* is when another rule $r' \in r^T$ that is also added to $R$ sets $H$ to *true*, and setting $H$ to *true* causes rules in $R$ to then set $l$ to true. However, in this scenario, $H$ is already *true* due to $r'$, so adding $r_L$ has no effect. (This is true even if $r_L$'s probability is negative.)

We still need to show $R \cup \{r\} \equiv R \cup r^{T\prime}$. For each of the rules being added by $r^{T\prime}$, the body is *true* in $M(R)$, and cannot change to *false* due to the addition of other rules in $r^{T\prime}$, because all rules in $R \cup r^{T\prime}$ are negation-free. Therefore, they have the same effect as the bodyless rule $h$, thus

$$R \cup r^{T\prime} \equiv R \cup \left\{ (-1)^{|L|}\sigma : h \ : \ L \subseteq r^- \cap M(R) \right\}$$

Since these rules are identical, we can sum them up:

$$R \cup r^{T\prime} \equiv R \cup \left\{ \sigma'' : h \right\}, \quad \sigma'' \stackrel{\text{def}}{=} \sum_{L \subseteq r^- \cap M(R)} (-1)^{|L|}\sigma$$

**Case 1:** If $r^- \cap M(R) \neq \emptyset$, then adding $r$ to $R$ has no effect. Let $l \in r^- \cap M(R)$:

$$\begin{aligned} \sigma'' &= \sum_{L \subseteq r^- \cap M(R) \setminus \{l\}} \left( (-1)^{|L|} + (-1)^{|L \cup \{l\}|} \right)\sigma \\ &= \sum_{L \subseteq r^- \cap M(R) \setminus \{l\}} \left( (-1)^{|L|} - (-1)^{|L|} \right)\sigma \ = \ 0 \end{aligned}$$

Thus adding $r^{T\prime}$ to $R$ has no effect either, so $R \cup r^{T\prime} \equiv R$. (The rules cancel each other's effect when added to $R$.)

**Case 2:** If $r^- \cap M(R) = \emptyset$, then adding $r$ to $R$ sets $H$ to *true* (with strength $\sigma$.) Then, due to the (negation-free) rules in $R$, some other variables may become *true* as well. Since $R \cup \{r\}$ is strongly consistent, none of the variables in $r^-$ may becomes *true*. Therefore:

$$R \cup \{r\} \equiv R \cup \{\sigma : h\}$$

Since $r^- \cap M(R) = \emptyset$,

$$\begin{aligned} R \cup r^{T\prime} &\equiv R \cup \left\{ \sum_{L \subseteq \emptyset} (-1)^{|L|}\sigma : h \right\} \\ &= R \cup \left\{ (-1)^{|\emptyset|}\sigma : h \right\} \\ &= R \cup \{\sigma : h\} \quad \equiv \quad R \cup \{r\} \end{aligned}$$

Finally, consider the case that $R$ is probabilistic. In this case, $R$'s semantics is a distribution over $2^{|R|}$ deterministic programs $D_i$. For any $D_i$ that has non-zero probability, $D_i \cup \{r\}$ is strongly consistent. $D_i$ is deterministic, therefore our proof so far implies that $D_i \cup \{r\} \equiv D_i \cup r^T$. Therefore, $R \cup r^T$ is also strongly consistent, and $R \cup \{r\} \equiv R \cup r^T$. □

## References


[1] K.R. Apt, M. Bezem, Acyclic programs, New Gener. Comput. 9 (3–4) (1991) 335–363.
[2] D. Buchman, M.W. Schmidt, S. Mohamed, D. Poole, N. de Freitas, On sparse, spectral and other parameterizations of binary probabilistic models, J. Mach. Learn. Res. 22 (2012) 173–181.
[3] L. De Raedt, A. Kimmig, H. Toivonen, ProbLog: a probabilistic Prolog and its application in link discovery, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI-2007, 2007, pp. 2462–2467.
[4] F.J. Dıez, S.F. Galán, An efficient factorization for the noisy max, Int. J. Intell. Syst. 18 (2) (2003) 165–177.
[5] P. Domingos, S. Kok, D. Lowd, H. Poon, M. Richardson, P. Singla, Markov logic, in: L.D. Raedt, P. Frasconi, K. Kersting, S. Muggleton (Eds.), Probabilistic Inductive Logic Programming, Springer, New York, 2008, pp. 92–117.
[6] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: ICLP/SLP, vol. 88, 1988, pp. 1070–1080.
[7] A. Jha, D. Suciu, Probabilistic databases with markoviews, Proc. VLDB Endow. 5 (11) (2012) 1160–1171.
[8] J. Kisynski, D. Poole, Lifted aggregation in directed first-order probabilistic models, in: Proc. Twenty-first International Joint Conference on Artificial Intelligence, IJCAI-09, Pasadena, California, 2009, pp. 1922–1929.
[9] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[10] Kowalski, R., Logic for Problem Solving, revisited. BoD–Books on Demand, 2014.
[11] S.L. Lauritzen, Graphical Models, Oxford University Press, USA, 1996.
[12] W. Meert, J. Vennekens, Inhibited effects in CP-logic, in: European Workshop on Probabilistic Graphical Models, Springer, 2014, pp. 350–365.
[13] D. Poole, The independent choice logic for modelling multiple agents under uncertainty, Artif. Intell. 94 (1997) 7–56, special issue on economic principles of multi-agent systems.
[14] D. Poole, Abducing through negation as failure: stable models in the Independent Choice Logic, J. Log. Program. 44 (1–3) (2000) 5–35, http://cs.ubc.ca/~poole/abstracts/abnaf.html.
[15] M. Richardson, P. Domingos, Markov logic networks, Mach. Learn. 62 (2006) 107–136.
[16] T. Sato, Y. Kameya, PRISM: a symbolic-statistical modeling language, in: Proceedings of the 15th International Joint Conference on Artificial Intelligence, IJCAI-97, 1997, pp. 1330–1335.
[17] T. Sato, Y. Kameya, New advances in logic-based probabilistic modeling by PRISM, in: L. De Raedt, P. Frasconi, K. Kersting, S. Muggleton (Eds.), Probabilistic Inductive Logic Programming, in: LNCS, vol. 4911, Springer, 2008, pp. 118–155, http://www.springerlink.com/content/1235t75977x62038/.
[18] G. Van den Broeck, W. Meert, A. Darwiche, Skolemization for weighted first-order model counting, arXiv:1312.5378, 2013.
[19] J. Vennekens, M. Denecker, M. Bruynooghe, CP-logic: a language of causal probabilistic events and its relation to logic programming, TPLP 9 (3) (2009) 245–308.