Why Rules are Complex: Real-Valued Probabilistic Logic Programs are not Fully Expressive

David Buchman[*]
davidbuc@cs.ubc.ca
http://www.cs.ubc.ca/∼davidbuc

David Poole[*]
poole@cs.ubc.ca
http://www.cs.ubc.ca/∼poole

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

Abstract

This paper explores what can and cannot be represented by probabilistic logic programs (PLPs). Propositional PLPs can represent any distribution because they can be acyclic. For relational domains with fixed populations, the probabilistic parameters can be derived as the solutions to polynomial equations. Unfortunately, sometimes they only have complex-valued solutions. Thus PLPs, even with arbitrarily real-valued parameters, cannot represent all distributions. Moreover, they cannot approximate all distributions. Allowing the parameters to be complex numbers, we present a natural truly-cyclic canonical representation that with probability 1 can represent all distributions for (propositional or) relational domains with fixed populations, and, unlike standard representations, has no redundant parameters.

1 INTRODUCTION

Logic programs have the advantage of an intuitive declarative and procedural interpretation (Kowalski, 2014). On the other hand, probabilistic modeling is an established formalism for representing and processing quantified uncertainty. Much attention has been given to formalisms combining logic programming with probabilistic modeling. One such combination is the family of probabilistic logic programs, including ICL (Poole, 1997, 2000), Prism (Sato & Kameya, 1997, 2008), ProbLog (De Raedt et al., 2007), and CP-logic (Vennekens et al., 2009), which separate uncertainty from logic, by encapsulating uncertainty into special variables representing independent probabilistic noise.

Recently, Buchman & Poole (2016, 2017) demonstrated that probabilistic logic programs (PLPs) are incapable of representing some relational distributions, even with a population size of just two. They showed that introducing negations ($\neg$'s) to the language does not help, but allowing negative numeric values as noise probabilities increases expressiveness and solve the problem (in which case $\neg$'s are not needed). They did not, however, examined whether negative probabilities are sufficient for representing arbitrary relational distributions using PLPs.

In this paper, we pick up from where Buchman & Poole (2017) left off. Table 1 describes some of our results. We show that for the relational setting, even for fixed populations, some distributions cannot be represented even using arbitrary real-valued probabilities ($\mathrm{P} \in (-\infty, +\infty)$). We prove that, in general, with probability 1 an arbitrary relational distribution with fixed populations can be represented using complex numbers as rule probabilities; therefore using complex probabilistic values is **essential**.

Furthermore, we expose a natural truly-cyclic canonical form for PLPs, the **Broadcast Canonical Form (BCF)**. BCF is very simple, and it allows to simplify the syntax of the language. We show BCF perfectly strips the redundancy from PLP rules, while maintaining expressiveness in both the propositional and the general (relational) setting (with fixed populations) (using complex probabilities, which are required anyway). We also provide solver algorithms, demonstrating how a BCF PLP representation can be computed, for modeling a given distribution.

In addition, we provide an algorithm for directly computing probabilities of interpretations, which is efficient in domains where few variables are $true$ (e.g., applications modeling rare events, system failures, medical conditions, etc.).

Negative probabilities have been used for optimizing inference or computation (Dıez & Galán, 2003; Kisynski & Poole, 2009; Van den Broeck et al., 2013). Meert & Vennekens (2014) suggested a new language feature, using negation in the head of rules in CP-logic, thus facilitat-

Table 1: Properties of languages

		$\mathbf{range(P)}$	**BCF** (Def. 1)		**PLP**		**PLP with ¬'s**
General (Rel.) case:		$P \in [0,1]$	no		no		no
(fixed popu.)	(Almost-)Fully Expressive?:	$P \in (-\infty,+\infty)$	no (Thm 2)		no (Thm 2)		
		$P \in \mathbb{C}$	yes (Thm 5)		yes (Thm 5)		yes (Thm 5)
Prop. case:	**#params:** (Sec. 6)		2^n-1	$<$	$n2^{n-1}$	$\ll$	$n3^{n-1}$
		$P \in [0,1]$	no		no		yes
	(Almost-)Fully Expressive?:	$P \in (-\infty,1]$	no		yes		yes
		$P \in \mathbb{C}$	yes (Thm 4)		yes		yes

ing expressiveness by supporting a cancellation effect of existing rules. Complex probabilities (Cox, 1955; Parisi, 1983; Hamber & Ren, 1985; Salcedo, 1997; Zak, 1998; Hofmann, 2012; Salcedo, 2016) have found use in quantum physics, but much less in the probabilistic modeling of general, non-quantum data.

2 PROGRAMS

We use uppercase letters for (possibly parametrized) Boolean **variables**, and lowercase letters for the corresponding atoms. V represents a set of (unparametrized) Boolean variables. An **interpretation** I is an assignment to all variables in V, and it is represented as the set $I \subseteq V$ of variables that are assigned *true* (sometimes we use the set of atoms instead).

A (probabilistic) **rule** has the form $p : head \leftarrow body$, where p is a numeric value commonly interpreted as a probability, $head$ is a (non-negated) atom, and $body$ is a conjunction of atoms, some of which may be negated.

Due to Buchman & Poole (2017)'s result that negative probabilities are needed for expressiveness and that ¬'s do not increase expressiveness, we **do not use ¬'s**.

A **(propositional) probabilistic logic program (PLP)** R is a set of rules. A **deterministic program realization (DPR)** for a (propositional or ground) PLP R is a deterministic program, whose rules are a subset of R's. We use the unique stable-model semantics as the semantics for DPRs. A PLP R defines a distribution over its DPRs, where each rule $p : head \leftarrow body$ independently appears with probability p or does not appear with probability $1-p$. Considering that each DPR represents its stable-model, a PLP represents a distribution over interpretations, i.e., a joint distribution over the variables.

The **relational setting** introduces **populations**, which are sets of **individuals**, and allows variables to be parametrized by **logical variables** (written in uppercase), which refer to individuals. Rules may thus also become parametrized. Once populations are fixed, a PLP can be ground. The ground PLP can be treated as a propositional PLP.

Relational PLPs may use constants, which refer to specific individuals. Once there is a constant for every individual, the PLP can contain rules that refer to any specific ground variable, and the PLP's expressiveness becomes identical to a propositional PLP's. The interesting question is what happens when there are individuals not specifiable using constants. The PLP's expressiveness is reduced; however, the requirement for being "(almost-)fully expressive" (defined later) is also reduced, since the PLP is only required to be able to represent distributions that are symmetric w.r.t. unspecifiable individuals. Therefore, we use no constants in this paper.

Let R be a propositional PLP over variables V, and $V_1, V_2 \subseteq V$. $R_{V_1 \leftarrow V_2} \subseteq R$ is the PLP over $V_1 \cup V_2$ that contains all rules in R whose head is in V_1 and whose bodies only contain variables from V_2; and $R_I \overset{\text{def}}{=} R_{I \leftarrow I}$.

In this paper we follow the approach of Buchman & Poole (2015) for simplifying syntax and semantics, and implicitly assume that different logical variables correspond to different individuals, thus a rule $a(X) \leftarrow b(X) \wedge c(Y)$ actually stands for $a(X) \leftarrow b(X) \wedge c(Y) \wedge (X \neq Y)$. There is no loss of generality, as every rule can be converted to multiple rules in which all logical variables correspond to different individuals, such that the overall semantics of the PLP does not change; this can be shown in a manner analogous to Buchman & Poole (2015)'s proof for Markov logic networks.

2.1 USING COMPLEX PROBABILITIES

Buchman & Poole (2017) suggested using negative numerical values as rule probabilities. They defined the semantics, which may either be a proper or improper distribution over interpretations; either way, however, the sum of the interpretations' probabilities is 1. They also argued against the perceived meaninglessness of using negative probabilities, by treating them as meaningless mathematical parameters of a probabilistic model; as long as the mathematical definition of the PLP's semantics is a proper distribution, there is no problem with the negativeness of some rule "probabilities". In this paper, we consider using complex numerical values as rule probabilities. The arguments of Buchman & Poole (2017) about rule "probabilities" not having to be meaningful as probabilities, apply here as well. In addition, their analysis, showing that a PLP's distribution sums to 1 even if some rule probabilities are negative, holds also for complex rule probabilities.

2.2 ALMOST-FULL EXPRESSIVENESS

The space of all (proper) distributions over n binary variables is the standard D-dimensional simplex $S_{\langle D\rangle}$, where $D = 2^n - 1$. Let $S_{\mathcal{L}}^{+}$ be the set of all (proper) distributions that a language $\mathcal{L}$ can represent, and $S_{\mathcal{L}}^{-} = S_{\langle D\rangle} \setminus S_{\mathcal{L}}^{+}$. If $S_{\mathcal{L}}^{-} = \emptyset$, then $\mathcal{L}$ has **full expressiveness**. Sometimes, $S_{\mathcal{L}}^{-}$ represents one or more (nonlinear) **"$\neq$" constraints** on the distribution's probabilities that are "$(D-1)$-dimensional", and $S_{\mathcal{L}}^{+}$ has measure 1. We then say $\mathcal{L}$ has **almost-full expressiveness**. Note that, in this case, if $\mathrm{P} \in S_{\mathcal{L}}^{-}$, then $\mathcal{L}$ can still represent P, allowing for infinitesimal error. (If this were not the case, then one could draw a D-dimensional sphere around P, such that this sphere, with its positive D-dimensional volume, would be contained in $S_{\mathcal{L}}^{-}$.)

When $\mathcal{L}$ is not almost-fully expressive, we say $\mathcal{L}$ is **partially-expressive**. This may happen when, e.g., $S_{\mathcal{L}}^{-}$ is defined by a **"$<$" constraint** on the distribution's probabilities, leading $S_{\mathcal{L}}^{-}$ to have "positive D-dimensional volume". This leads to distributions P that cannot even be approximated (to arbitrary accuracy).

3 THE PROPOSITIONAL CASE

Given a propositional PLP R over variables V, the probability of an interpretation $I \subseteq V$ is a complicated function of R's parameters (i.e., rule probabilities). In general, $\mathrm{P}^R(I)$ can be decomposed into a product of two expressions: one sets variables in I to $true$, and one makes sure variables in $V \setminus I$ remain $false$. The following theorem makes this intuitive argument mathematically precise. The "expression that sets variables in I to $true$" is not well-defined, so we replace it with the equivalent "the probability of all variables being $true$, in another ('projected') PLP R_I that only contains the variables I".

Theorem 1. *Let R be a propositional (or grounded) PLP over variables V, and $I \subseteq V$ be an interpretation. Then:*

$$\mathrm{P}^R(I) = \mathrm{P}^{R_I}(I) \prod_{(p:\ head \leftarrow body) \in R_{(V\setminus I)\leftarrow I}} (1-p) \quad (1)$$

$$\mathrm{P}^{R_I}(I) = 1 - \sum_{I' \subsetneq I} \mathrm{P}^{R_I}(I') \quad (2)$$

Proof. Equation (2) holds because for all PLPs, the total probability of all interpretations is always 1 (even with complex probabilities).

The equation for $\mathrm{P}^R(I)$ has two components.

1. First, we make sure the variables in $V \setminus I$ remain $false$ even if variables in I are $true$. For this, we must make sure the rules in $R_{(V\setminus I)\leftarrow I}$ do not appear.

2. Second, we must make sure the variables in I are set to $true$. This only depends on the appearance of rules that completely ignore variables in $V \setminus I$. This probability is identical to $\mathrm{P}^{R_I}(I)$, which is the probability of all variables in I being $true$, when the variables in $V \setminus I$ simply do not exist. □

Theorem 1 defines a recursive formulation of interpretation probabilities, by Equation (1) reducing the computation to a computation in a PLP without the variables that are $false$, and Equation (2) reducing the computation further, to interpretations where fewer variables are $true$, for which (1) may again be applied.

Algorithm 1 is a recursive, bottom-up (dynamic programming) algorithm for computing the probability of any interpretation, based on Theorem 1. (Intermediate results can be cached, for efficiency.) Note that the recursion is over both interpretations and PLPs.

Algorithm 1 Compute probability of an interpretation.

```
Input:  $V$: Set of ground variables,
        $R$: ground PLP over $V$, $I \subseteq V$: interpretation
Output: computeP($V$,$R$,$I$) $= \mathrm{P}^R(I)$
if $V = \emptyset$ then
    return $1$
else if $I = V$ then
    return $1 - \sum_{I' \subsetneq I}$ computeP($V$,$R$,$I'$)
else
    return computeP($I$,$R_I$,$I$)
        $\cdot \prod_{(p:\ \text{head}\leftarrow\text{body}) \in R_{(V\setminus I)\leftarrow I}} (1-p)$
end if
```

While this algorithm takes exponential time in $|I|$, it can be useful for small I's. In addition, its explicit recursive formulation of $\mathrm{P}^R(I)$ makes it useful for theoretical analysis. We use it to derive theoretical results.

4 RELATIONAL PLPS OVER $\{A(X)\}$

We now consider the simplest setting for a relational PLP with fixed populations: a PLP having a single variable $A(X)$. This setting was analyzed by Buchman & Poole (2017) for $n = 2$ individuals; we generalize to $n > 2$.

Every rule in this setting has the following structure (recall that all logical variables may be assumed to be different (Section 2)): $\quad a(X) \leftarrow a(Y) \wedge a(Z) \wedge \ldots$

Thus, for a population of size n, a PLP can be described parametrically, as a set of n rules (some of which may have the probability 0):

$$
\begin{aligned}
p_0 &: \ a(X) \\
p_1 &: \ a(X) \leftarrow a(Y_1) \\
p_2 &: \ a(X) \leftarrow a(Y_1) \wedge a(Y_2) \\
&\ \ \vdots \\
p_{n-1} &: \ a(X) \leftarrow a(Y_1) \wedge a(Y_2) \wedge \ldots \wedge a(Y_{n-1})
\end{aligned}
$$

We use $\mathrm{P}^R_{n_t,n_f}$ for the probability a PLP R assigns to any interpretation where n_t variables are $true$ and n_f are $false$, for $n = n_t + n_f$. For example, for $n = 3$, $\mathrm{P}^R(\{a(\text{alice}), a(\text{bob})\}) = \mathrm{P}^R(a(\text{alice}) \wedge \neg a(\text{eve}) \wedge a(\text{bob})) = \mathrm{P}^R_{2,1}$.

The following proposition follows directly from Theorem 1, and it generalizes the derivation used for proving Proposition 1 in Buchman & Poole (2017).

Proposition 1. *Given a PLP R involving only $A(X)$, $\mathrm{P}^R_{n_t,n_f}$ for all interpretations for all population sizes n can be computed using the recursive formulae:*

$$
\begin{aligned}
\mathrm{P}^R_{0,0} &= 1 \\
\mathrm{P}^R_{n,0} &= 1 - \textstyle\sum_{m=0}^{n-1} \binom{n}{m} \mathrm{P}^R_{m,n-m} && (3) \\
\mathrm{P}^R_{n_t,n_f} &= \mathrm{P}^R_{n_t,0} \textstyle\prod_{i=0}^{n_t} (1-p_i)^{\left(n_f \cdot \binom{n_t}{i}\right)} && (4)
\end{aligned}
$$

4.1 A BACKTRACKING SOLVER

Let $\mathrm{P}^g > 0$ (for P^{goal}) be an arbitrary positive distribution over $\{A(X)\}$. Algorithm 2 is a recursive backtracking solver for finding a PLP R that represents P^g, if one exists (it is written in rolled-out form). A practical implementation would cache values, to reduce its complexity.

The idea behind the algorithm is as follows. R represents P^g, i.e., $\mathrm{P}^R = \mathrm{P}^g$, if and only if the following n equations hold: $\mathrm{P}^g_{0,n} = \mathrm{P}^R_{0,n}$, $\mathrm{P}^g_{1,n-1} = \mathrm{P}^R_{1,n-1}$, $\ldots$, $\mathrm{P}^g_{n-1,1} = \mathrm{P}^R_{n-1,1}$ (the additional equation $\mathrm{P}^g_{n,0} = \mathrm{P}^R_{n,0}$ is redundant). One can write these equations and use Proposition 1 to express each $\mathrm{P}^R_{n_t,n_f}$ using the parameters, $\{p_0, \ldots, p_{n-1}\}$. Ordering the equations by increasing n_t's, each equation contains exactly one parameter that does not appear in the previous equations. One can then solve each equation in turn. Each equation is polynomial, and may have multiple solutions. If some equation has no solutions, one backtracks. Therefore, the algorithm returns a correct PLP representation if one exists, and "impossible" if one does not exist. There are two reasons why a solution may not exist:

1. An equation's solutions may fall outside the allowed range for rule probabilities. This problem is eliminated by allowing complex rule probabilities.

2. If $\mathrm{P}^R_{n_t,0} = 0$ for some $n_t < n$, then there is no solution. This, however, only creates lower-dimensional manifolds that are (potentially) not representable.

Since $\mathrm{P}^R_{n_t,0}$ depends on values of previous p_i's computed, backtracking and changing their values may make it nonzero. Therefore, it might be the case that complex solutions always exist, and Proposition 2 can be strengthened to prove the language is fully expressive. Whether this is the case is an open problem.

Also note, that since $\mathrm{P}^g > 0$, the p_i's found are not 1, thus $\forall i, (1-p_i) \neq 0$, and the PLPs found have no deterministic rules.

4.2 REAL-VALUED PLPS: NOT FULLY EXPR.

Buchman & Poole (2017) have shown that all positive distributions with $n = 2$ can be represented, if one allows negative probabilities. Whether negative probabilities are enough to represent all positive distributions in general (for fixed n's) was left as an open problem. Unfortunately, the answer is negative:

Example 1. *For $n = 3$, the following positive distribution cannot be represented using PLPs, even allowing negative rule probabilities:*
$\mathrm{P}^g_{0,3} = \frac{1}{3^3} = \frac{1}{27}$, $\mathrm{P}^g_{1,2} = \frac{2^3}{3^3} = \frac{8}{27}$, $\mathrm{P}^g_{2,1} = \frac{1}{54}$, $\mathrm{P}^g_{3,0} = \frac{1}{54}$
Simulating Algorithm 2, (3) *and* (4) *give:*

$$
\begin{aligned}
\tfrac{1}{27} &= \mathrm{P}^g_{0,3} = \mathrm{P}^R_{0,3} = \mathrm{P}^R_{0,0} \textstyle\prod_{i=0}^{0} (1-p_i)^{\left(3\binom{0}{i}\right)} = (1-p_0)^3 && (5) \\
\tfrac{8}{27} &= \mathrm{P}^g_{1,2} = \mathrm{P}^R_{1,2} = \mathrm{P}^R_{1,0} \cdot \textstyle\prod_{i=0}^{1} (1-p_i)^{\left(2\cdot\binom{1}{i}\right)} && (6) \\
&= p_0 \cdot (1-p_0)^2 (1-p_1)^2 \\
\mathrm{P}^R_{2,0} &= 1 - \textstyle\sum_{m=0}^{1} \binom{2}{m} \mathrm{P}^R_{m,2-m} && (7) \\
&= 1 - \binom{2}{0} \mathrm{P}^R_{0,2} - \binom{2}{1} \mathrm{P}^R_{1,1} \\
&= 1 - (1-p_0)^2 - 2p_0(1-p_0)(1-p_1) \\
\tfrac{1}{54} &= \mathrm{P}^g_{2,1} = \mathrm{P}^R_{2,1} = \mathrm{P}^R_{2,0} \textstyle\prod_{i=0}^{2} (1-p_i)^{\left(1\cdot\binom{2}{i}\right)} && (8)
\end{aligned}
$$

Solving (5) *gives $p_0 = \frac{2}{3}$. Substituting into* (6) *gives $(1-p_1)^2 = 4$, so $p_1 \in \{-1, 3\}$. We required $p_1 < 1$, therefore $p_1 = -1$. Substituting into* (7) *gives $\mathrm{P}^R_{2,0} = 0$. Substituting $\mathrm{P}^R_{2,0} = 0$ into* (8) *gives $\mathrm{P}^R_{2,1} = 0 \neq \frac{1}{54}$ irrespective of p_2, thus representing P^g is not possible.*

Example 2. *If we also allow rule probabilities above* 1*, the distribution in Example 1 can be represented.*

Consider Example 1. This time, we can choose $p_1 = 3$. Substituting into (7) *gives $\mathrm{P}^R_{2,0} = \frac{16}{9}$. (The PLP thus represents an improper distribution for $n = 2$; however, we are trying to model P^g, which is only defined for $n = 3$. We treat "$\mathrm{P}^R_{2,0}$" as nothing more than a name given to a meaningless expression used in the formula of $\mathrm{P}^R_{2,1}$.) Substituting into* (8) *gives $p_2 = \frac{127}{128}$. These $\{p_0, p_1, p_2\}$ give $\{\mathrm{P}^g_{0,3} = \mathrm{P}^R_{0,3}, \mathrm{P}^g_{1,2} = \mathrm{P}^R_{1,2}, \mathrm{P}^g_{2,1} = \mathrm{P}^R_{2,1}\}$, thus $\mathrm{P}^g = \mathrm{P}^R$.*

Algorithm 2 `find_PLP` $(n, \mathrm{P}^g, range)$: Find a PLP representing a given positive relational dist. P^g over $\{A(X)\}$.

```
Inputs:   $n$:        Population size                    $\mathrm{P}^g$:   Positive relational distribution over $\{A(X)\}$
          $range$: Range allowed for rule probabilities
Output:   Either a PLP $R$ for $\mathrm{P}^g$, represented using $(p_0, p_1, \ldots, p_{n-1})$   /   or "impossible"

Write the $n$ equations (symbolically):   $\left( \mathrm{P}^g_{0,n} = \mathrm{P}^R_{0,n}, \quad \ldots, \quad \mathrm{P}^g_{n-1,1} = \mathrm{P}^R_{n-1,1} \right)$
loop over the equations in increasing order of $n_t$:
  Substitute the expression for $\mathrm{P}^R_{n_t,n_f}$ given by (4), into the current equation
  // Current equation is now:   $\mathrm{P}^g_{n_t,n_f} = \mathrm{P}^R_{n_t,0} \prod_{i=0}^{n_t} (1-p_i)^{\left(n_f \cdot \binom{n_t}{i}\right)}$
  Evaluate $\mathrm{P}^R_{n_t,0}$ numerically using $(p_0, \ldots, p_{n_t-1})$ according to Proposition 1
  if $\mathrm{P}^R_{n_t,0} = 0$ then backtrack          // Current equation cannot be solved: $\mathrm{P}^g_{n_t,n_f} = 0 \cdot (\prod \ldots)$
  Substitute into the equation:   $n_t$,   $n_f$,   $\mathrm{P}^g_{n_t,n_f}$,   $\mathrm{P}^R_{n_t,0}$,   $p_0, \ldots, p_{n_t-1}$
  // Equation is now:   positive real number = nonzero number $\cdot (1-p_{n_t})^{\text{positive integer}}$
  $solutions_{n_t} \leftarrow$ the solutions for $p_{n_t}$ that fall inside $range$      // Out of $n_f$ (possibly complex) solutions
  Try one of $solutions_{n_t}$, and advance to the next equation. If the next equation backtracks,
          try another solution. If all solutions in $solutions_{n_t}$ backtracked, then backtrack.
          If all equations were solved then return $(p_0, \ldots, p_{n-1})$.
end loop
return "impossible"
```

In general, p_0 and p_1 can be computed as functions of $\mathrm{P}^g_{0,3}$ and $\mathrm{P}^g_{1,2}$; $\mathrm{P}^R_{2,0}$ is then computed as a function of these, and we then get the constraint $\mathrm{P}^R_{2,0} \neq 0$. (If $\mathrm{P}^R_{2,0} = 0$, then $\mathrm{P}^g > 0$ cannot be represented.) $\mathrm{P}^R_{2,0} \neq 0$ is therefore a complicated non-linear constraint on the values of $\{\mathrm{P}^g_{0,3}, \mathrm{P}^g_{1,2}\}$. This constraint is unexpected and surprising. Given a distribution, it is not immediately obvious if this constraint is violated.

Fortunately, however, since the constraint is $\mathrm{P}^R_{2,0} \neq 0$ (and not, say, $\mathrm{P}^R_{2,0} > 0$), the probability that a "random" distribution P^g violates this constraint is 0, i.e., this constraint has measure 0. It represents a "singularity" of the language, but the language is still useful, since one can use an alternative $\mathrm{P}^{g'}$ that is infinitesimally close to a P^g that violates the constraint.

Unfortunately, there are also constraints with positive "D-dimensional volume". This is a limitation of the language itself, not of any specific program. Adding an arbitrary number of new formulae, or allowing probabilities above 1, does not solve the problem:

Theorem 2. *The language of PLPs with generalized real-valued rule probabilities* ($\mathrm{P} \in (-\infty, +\infty)$) *is only partially-expressive, even with fixed population sizes.*

Proof. We prove by giving a negative example, with a single variable $A(X)$ and $n = 4$. First, we show real-valued PLPs cannot represent any distribution P^g with:

$$\mathrm{P}^g_{0,4} = \tfrac{1}{3^4} = \tfrac{1}{81}, \quad \mathrm{P}^g_{1,3} = \tfrac{2^4+1}{3^4} = \tfrac{17}{81}, \quad \mathrm{P}^g_{2,2} > 0$$

Simulating Algorithm 2, (3) and (4) give:

$$\tfrac{1}{81} = \mathrm{P}^g_{0,4} = \mathrm{P}^R_{0,4} = \mathrm{P}^R_{0,0} \prod_{i=0}^{0} (1-p_i)^{\left(4\binom{0}{i}\right)} = (1-p_0)^4 \tag{9}$$

$$\tfrac{17}{81} = \mathrm{P}^g_{1,3} = \mathrm{P}^R_{1,3} = \mathrm{P}^R_{1,0} \cdot \prod_{i=0}^{1} (1-p_i)^{\left(3 \cdot \binom{1}{i}\right)} \tag{10}$$
$$= p_0 \cdot (1-p_0)^3 (1-p_1)^3$$

$$\mathrm{P}^R_{2,0} = \ldots \quad \text{(repeating (7))} \tag{11}$$
$$= 1 - (1-p_0)^2 - 2p_0(1-p_0)(1-p_1)$$

$$0 < \mathrm{P}^g_{2,2} = \mathrm{P}^R_{2,2} = \mathrm{P}^R_{2,0} \prod_{i=0}^{2} (1-p_i)^{\left(2 \cdot \binom{2}{i}\right)} \tag{12}$$
$$= \mathrm{P}^R_{2,0} \left((1-p_0)(1-p_1)^2 (1-p_2) \right)^2$$

Solving (9) gives $p_0 \in \{\frac{2}{3}, \frac{4}{3}\}$. We then substitute into (10) to find p_1, then into (11) to compute $\mathrm{P}^R_{2,0}$, and then into (12) to find p_2. Taking $p_0 = \frac{2}{3}$ gives $p_1 = -1.0408$ and $\mathrm{P}^R_{2,0} = -0.0181$. Taking $p_0 = \frac{4}{3}$ gives $p_1 = 2.6198$ and $\mathrm{P}^R_{2,0} = -0.5509$. Either way, we get $\mathrm{P}^R_{2,0} < 0$, and since (12) is "$0 < \mathrm{P}^g_{2,2} = \mathrm{P}^R_{2,0} \cdot (\ldots)^2$", P^g cannot be represented using real numbers.

Since $\mathrm{P}^R_{2,0}$ was computed as a continuous function of p_0 and p_1, which themselves were computed as continuous functions of $\mathrm{P}^g_{0,4}$ and $\mathrm{P}^g_{1,3}$, an infinitesimal perturbation of P^g will not be sufficient to make $\mathrm{P}^R_{2,0} > 0$ and the distribution representable. Thus, there is a D-dimensional region of distributions that are not representable. □

4.3 COMPLEX PLPS: ALMOST-FULLY EXPR.

We suggest extending the PLP language to support **complex** rule probabilities.

Proposition 2. *1. The language of PLPs with **complex** rule probabilities is almost-fully expressive w.r.t. positive relational distributions over $\{A(X)\}$ with a fixed n.*
2. The number of PLP representations for every such distribution is finite.
3. There is at most one such PLP representation where all parameters are real and < 1.

Proof. **1.** We prove by showing that Algorithm 2 finds a PLP representing (almost) any given P^g. Using complex probabilities, the only possibility of Algorithm 2 to fail to find a solution for p_{n_t}, is due to $\mathrm{P}^R_{n_t,0} = 0$. The algorithm thus finds a PLP representing P^g if and only if the following n constraints hold: $\mathrm{P}^R_{0,0} \neq 0$, $\mathrm{P}^R_{1,0} \neq 0, \ldots,$ $\mathrm{P}^R_{n-1,0} \neq 0$. We now show these are lower-dimensional constraints (with measure 0), thus the language is almost-fully expressive.

$\mathrm{P}^R_{n_t,0}$ is the sum of the probabilities of the DPRs (for population size n_t) in whose minimal stable model all variables are $true$. The probability of each DPR D is the polynomial $(\prod_{\text{rules} \in D} p_i) \cdot (\prod_{\text{rules} \notin D}(1 - p_i))$. Each $\mathrm{P}^R_{n_t,0}$ is therefore a polynomial of $p_0, \ldots, p_{n_t-1}$. Equating a polynomial to 0 gives a lower-dimensional manifold – unless the polynomial happens to be 0 everywhere. This, however, is not the case here, because if we set all rule probabilities to be, say, 0.5, the probability that all variables become $true$ will be positive.

2. Every equation the algorithm solves has a finite number of solutions, thus there is a finite number of PLP representations for P^g.

3. Every parameter is found by solving an equation of the form: positive real number = nonzero number $\cdot(1 - p_{n_t})^{\text{positive integer}}$, i.e., $q_{n_t}^{\text{positive integer}} =$ number, where $q_{n_t} = 1 - p_{n_t}$. q_{n_t} cannot have two positive solutions, thus p_{n_t} cannot have two real solutions below 1. □

5 BROADCAST CANON. FORM (BCF)

PLPs over $\{A(X)\}$ have a special property: every rule has a different body. In general, as we will show, the number of different possible rule **bodies** is identical to the number of degrees of freedom (DoF) the distribution P^g has. PLPs over $\{A(X)\}$ have the exact number of possible rules as the needed DoF, but they cannot represent all distributions, unless they allow their parameters (rule probabilities) to be complex. PLPs over other sets of parametrized variables are overparametrized, i.e., they contain more possible rules (thus more parameters) than P^g's DoF. Overparametrization is inefficient. While practical PLPs do not simultaneously use all possible rules, it is possible that the rules they use contain redundancy, which may be unintuitive and difficult to identify. In the context of learning undirected graphical models with binary variables from data, parametrizations with excessive redundant parameters learn graphical models with excessive parameters (even after pruning un-useful parameters), that are slower to process, without improving learning accuracy (Buchman et al., 2012).

Definition 1. *A program in the **Broadcast Canonical Form (BCF) Language** (or a **BCF PLP**) is a set of negation-free non-deterministic rules with the form: p : $body$. Its semantics is that of the corresponding PLP, in which each p : $body$ is replaced with the rules p : $head \leftarrow body$ for **all possible** heads. (If $p = 0$, then these rules simultaneously do not appear.)*

Bodies that are equivalent, e.g., $a(X) \wedge b(X)$ and $b(Y) \wedge a(Y)$, are considered the same body. "p : $body$" represents a set S_{body} of rules p : $head \leftarrow body$ in the corresponding PLP. In each of its DPRs, some of the rules in S_{body} may appear, and some might not appear.

BCF is a simplified language, and allows to write PLPs in succinct form. A BCF PLP is a PLP with tied weights. We show BCF is a canonical form for PLPs: it is almost-fully expressive, yet without excessive parameters.

We designate BCF parameters by subscripting them with the set of the variables in their bodies; e.g., for $V = \{A, B, C\}$, the BCF $\{p_{\{C\}} : c,\ p_{\{B,C\}} : b \wedge c\}$ corresponds to the PLP $\{p_{\{C\}} : a \leftarrow c,\ p_{\{C\}} : b \leftarrow c,\ p_{\{B,C\}} : a \leftarrow b \wedge c\}$.

Since PLPs over $\{A(X)\}$ can only have a single rule for every possible body, our results for PLPs over $\{A(X)\}$ actually describe results for BCFs over $\{A(X)\}$. Section 4's p_0 is represented by BCF's $p_\emptyset$, p_1 is represented by $p_{\{A(Y_1)\}}$, p_2 is represented by $p_{\{A(Y_1),A(Y_2)\}}$, etc.

BCF is very simple, yet highly unintuitive. For example, all probabilistic facts have the same probability (!). The unintuitiveness is because p : $head \leftarrow body$ is usually interpreted as a probability that is related to the influence that $body = true$ has on $head$. To make sense of BCF, take p : $body$ to mean a "broadcast message" that $body = true$ sends to all other variables, that increases (or decreases, for negative p's) the probabilities of all other variables. If $body = true$ should influence some variables but not others, this is achieved using the combined influence of other rules. The notion of "probabilistic strengths" Buchman & Poole (2017) makes this influence combination additive and intuitive.

6 BCF AND OTHER CANONICAL FORMS FOR PROPOSITIONAL PLPS

As noted in Section 5, PLPs for the simplest relational setting (Section 4) are actually BCF PLPs. We now develop BCF for the simplest (i.e., propositional) multi-variable case. The development and results are similar to those of Section 4.

In the propositional setting, there are n variables, $V = \{V_1, \dots, V_n\}$, and 2^n possible joint assignments (i.e., interpretations). A ("proper") joint distribution is a non-negative real function summing to 1, over all interpretations, thus it has $2^n - 1$ DoF. An "improper" joint distribution is a function summing to 1, which may contain negative, or even complex, numbers.

PLPs have one DoF for each possible rule (or two DoF, for complex rule probabilities). $\neg$-free (real-valued) PLPs thus have $n2^{n-1}$ DoF (see Table 1), because each rule specifies which variable appears in its head, and the subset of the $n-1$ remaining variables that appears in its body. $\neg$-free PLPs thus have excessive DoF, yet are only partially expressive. Allowing negative probabilities does not add DoF, but it makes $\neg$-free PLPs fully expressive. (It may also allow the representation of improper distributions.) The PLP, however, is still over-parametrized.

(Real-valued) PLPs with $\neg$'s have $n3^{n-1}$ DoF, because each variable (other than the head variable) may either appear in the body, not appear, or appear negated. PLPs with $\neg$'s are fully expressive, but extremely over-parametrized. (They may also be logically inconsistent.)

For PLPs with $\neg$'s, there is a systematic way (a "canonical form") to represent any given distribution: Order the variables arbitrarily $V_1, \dots, V_n$, and code each $\mathrm{P}(v_i \mid v_1, \dots, v_{i-1})$ using one rule for each joint assignment to $(v_1, \dots, v_{i-1})$, with v_i as its head. For $\neg$-free PLPs with negative probabilities there is a similar systematic way, where each $\mathrm{P}(v_i \mid v_1, \dots, v_{i-1})$ is coded using one rule for each conjunction of subsets of $\{v_1, \dots, v_{i-1}\}$, with v_i as its head. For both languages, there are a large but finite ($n!$) distinct PLPs that can be considered canonical representations for the given distribution. This finiteness is meaningful: In general, a single redundant real-valued parameter may give ∞ ways for representing any distribution, thus the finiteness means there are no redundant parameters. For both languages, each of the canonical PLP representations only uses a (different) subset of $2^n - 1$ rules out of the $n2^{n-1}$ or $n3^{n-1}$ available rules.

We embrace complex rule probabilities, since they are needed in the general, relational case. We now ask: Allowing complex probabilities, which new systematic ways are there, for representing any given distribution?

We suggest a new systematic way, BCF, to parametrize $\neg$-free propositional PLPs using $2^n - 1$ complex rule probabilities that, for non-extreme distributions, is almost-fully expressive. For some (but not all) distributions, BCF provides a unique canonical BCF PLP that only involves real probabilities in $(-\infty, 1)$. In general, BCF provides any given distribution with a finite number of (complex-valued) canonical BCF PLP representations. ($2^n - 1$ complex rule probabilities have $2 \cdot (2^n - 1)$ DoF, but they are not overparametrized. This is because PLPs with complex rule probabilities are fully expressive w.r.t. **complex** (improper) distributions, thus adding $2^n - 1$ DoF to the specification of the distribution.) BCF, however, simplifies the language, and requires no arbitrary ordering. BCF's lack of ordering means that its canonical PLPs are truly cyclic, instead of the Bayes-net semantics of the other two canonical forms. BCF's semantics is intriguing, and not fully understood.

Propositional BCF PLPs have $2^n - 1$ possible parameters: $\{p_s : s \subsetneq V\}$. p_V is excluded (it has no effect). We sometimes subscript by the set of atoms, e.g., $p_{\{a\}}$ instead of $p_{\{A\}}$.

Theorem 3 is the adaptation of Theorem 1 to BCF. It may be expressed as an algorithm, similar to Algorithm 1.

Theorem 3. *Let R be a propositional BCF PLP over variables V, and $I \subseteq V$ be an interpretation. Then:*

$$\mathrm{P}^R(I) = \mathrm{P}^{R_I}(I) \prod_{s \subseteq I} (1 - p_s)^{|V \setminus I|} \qquad (13)$$

$$\mathrm{P}^{R_I}(I) = 1 - \sum_{I' \subsetneq I} \mathrm{P}^{R_I}(I')$$

6.1 A BCF BACKTRACKING SOLVER

Algorithm 3 is the adaptation of Algorithm 2 to BCF in the propositional case. Let P^g be a positive distribution over V, and let R be a BCF PLP over V. R represents P^g, i.e., $\mathrm{P}^R = \mathrm{P}^g$, if and only if the following $2^n - 1$ equations hold: $\{\mathrm{P}^g(s) = \mathrm{P}^R(s) : s \subsetneq V\}$. (The additional equation $\mathrm{P}^g(V) = \mathrm{P}^R(V)$ is redundant.)

Example 3. *For two variables, $V = \{A, B\}$, the $2^n - 1$ equations $\mathrm{P}^g(V) = \mathrm{P}^R(V)$ are:*

$$\mathrm{P}^g(\emptyset) = \mathrm{P}^{R_\emptyset}(\emptyset) \prod_{s \subseteq \emptyset} (1 - p_s)^2 = (1 - p_\emptyset)^2$$
$$\mathrm{P}^g(a) = \mathrm{P}^{R_{\{A\}}}(a) \prod_{s \subseteq \{A\}} (1 - p_s)^1 = p_\emptyset (1 - p_\emptyset)(1 - p_{\{a\}})$$
$$\mathrm{P}^g(b) = \mathrm{P}^{R_{\{B\}}}(b) \prod_{s \subseteq \{B\}} (1 - p_s)^1 = p_\emptyset (1 - p_\emptyset)(1 - p_{\{b\}})$$

The algorithm solves one equation after another, and gives the general solution:

$$p_\emptyset = 1 \pm \sqrt{\mathrm{P}^g(\emptyset)}, \ \ p_{\{a\}} = 1 - \frac{\mathrm{P}^g(\{a\})}{p_\emptyset(1 - p_\emptyset)}, \ \ p_{\{b\}} = 1 - \frac{\mathrm{P}^g(\{b\})}{p_\emptyset(1 - p_\emptyset)}$$

There is always a solution in real probabilities below 1, and another solution in real probabilities above 1.

Example 4. *Consider running a solver like Algorithm 3, to find a general (non-BCF) PLP for $V = \{A, B\}$. Let*

Algorithm 3 `find_prop_BCF_PLP(`V`,`P^g`)`: Find a BCF PLP representing a given pos. **propositional** dist. P^g.

```
Inputs:   $V$:   The set of variables                    $\mathrm{P}^g$:   Positive distribution over $V$
Output:   BCF PLP for $\mathrm{P}^g$, represented using $\{p_s : s \subsetneq V\}$   /   or "impossible"
Write the $2^n - 1$ equations (symbolically):   $\{\mathrm{P}^g(I) = \mathrm{P}^R(I) \;:\; I \subsetneq V\}$.
loop over the equations in (any) increasing partial-order of $I$:
    Substitute the expression for $\mathrm{P}^R(I)$ given by (13)
    // Current equation is now:   $\mathrm{P}^g(I) = \mathrm{P}^{R_I}(I)\prod_{s\subseteq I}(1-p_s)^{|V\setminus I|}$
    Evaluate $\mathrm{P}^{R_I}(I)$ numerically using $\{p_s : s \subsetneq I\}$ according to Theorem 3
    if $\mathrm{P}^{R_I}(I) = 0$ then backtrack
    Substitute into the equation:   $n_t$,   $n_f$,   $\mathrm{P}^g(I)$,   $\mathrm{P}^{R_I}(I)$,   $\{p_s : s \subsetneq I\}$
    // Equation is now:   positive real number = nonzero number $\cdot(1-p_I)^{\text{positive integer}}$
    $solutions_I \leftarrow$ the solutions for $p_I$        // $|V \setminus I| \geq 1$ (possibly complex) solutions
    Try one of $solutions_I$, and advance to the next equation. If the next equation backtracks,
            try another solution. If all solutions in $solutions_I$ backtracked, then backtrack.
            If all equations were solved then return $\{p_s : s \subsetneq V\}$.
end loop
return "impossible"
```

the rule probabilities be $\{p_a, p_b, p_{a\leftarrow b}, p_{b\leftarrow a}\}$*. Its first equation would be:*

$$\mathrm{P}^g(\emptyset) = \ldots = (1-p_a)(1-p_b)$$

In general, since PLPs are over-parametrized, equations may simultaneously introduce multiple new parameters. PLPs using only $A(X)$ *are a special case: they are not over-parametrized, so this does not happen. BCF is a technique for resolving the over-parametrization, by effectively forcing simultaneous new parameters to be equal, thus transforming equations to have a single new variable, thus getting rid of the PLP's excessive DoF.*

BCF has tied weights, thus fewer potential parameters; however, it is still almost-fully expressive:

Theorem 4. ***1.*** *The language of BCF PLPs with complex rule probabilities is almost-fully expressive for positive propositional distributions.*
2. *The number of BCF representations for every such distribution is finite.*
3. *There is at most one such BCF representation where all parameters are real and* < 1*.*

Proof. The proof is very similar to Proposition 2's, only it relies on Algorithm 3 instead of Algorithm 2, and the constraints are $\{\mathrm{P}^{R_I}(I) \neq 0 \;:\; I \subsetneq V\}$ instead of $\{\mathrm{P}^R_{n_t,0} \neq 0 : n_t \in \{0,\ldots,n-1\}\}$. □

7 GENERAL RELATIONAL BCF PLPS

We now generalize Sections 4 and 6 to general relational PLPs, given fixed populations.

Theorem 5. ***1.*** *The language of BCF PLPs with complex rule probabilities is almost-fully expressive for positive relational distributions with fixed population sizes.*
2. *The number of BCF representations for every such distribution is finite.*
3. *There is at most one such BCF representation where all parameters are real and* < 1*.*

Proof. The proof follows from the correctness of Algorithm 4; we only outline the differences with the special cases presented in Sections 4 and 6.

The first step is to ground the PLP, according to the given population sizes. Similarly to Algorithm 3, we write all equations $\mathrm{P}^g(I) = \mathrm{P}^R(I)$ for the ground PLP, and order them in increasing partial-order of I (and omit the last one). Here, however, the ground PLP is exchangeable w.r.t. swapping individuals. Swapping two individuals causes multiple pairs of ground variables to be swapped. Therefore, some interpretations are equivalent. P^g is equal for equivalent interpretations, and the PLP has no rule that can distinguish among them; therefore, they give identical equations, and we only keep the first occurrence of each equation. (This was implicitly done by Algorithm 2, which wrote n equations, although there are 2^n interpretations for $V = \{A(X_1), \ldots, A(X_n)\}$.)

The number of equations that remain after removing duplicate equations is the number of P^g's DoF. Pick an arbitrary I; we need to show that $\mathrm{P}^g(I) = \mathrm{P}^R(I)$ has exactly one new parameter that did not occur in previous equations. This parameter is p_I, which represents the probability of all rules with the form $head \leftarrow \bigwedge_{v\in I} v$. The equations are in increasing partial-order of I, so $\bigwedge_{v\in I} v$ was false for their corresponding interpretations, and p_I did not appear in them. It is also not the case that there was a second new parameter that appears in

Algorithm 4 `find_BCF_PLP` (V, P^g): Find a BCF PLP representing a given **general** pos. (rel.) distribution P^g.

```
Inputs:  $V$:  The set of **ground** variables        $\mathrm{P}^g$:  Positive distribution over $V$
Output:  BCF PLP for $\mathrm{P}^g$  /  or "impossible"

Write the $2^{|V|}-1$ equations (symbolically):   $\{\mathrm{P}^g(I) = \mathrm{P}^R(I) \;:\; I \subsetneq V\}$.
Remove recurring equations (keep only the first of each)
**loop** over the equations in (any) increasing partial-order of $I$:
    Substitute the expression for $\mathrm{P}^R(I)$ given by (13)
    Evaluate $\mathrm{P}^{R_I}(I)$ numerically using $\{p_s : s \subsetneq I\}$ according to Theorem 3
    **if** $\mathrm{P}^{R_I}(I) = 0$ **then backtrack**
    Substitute into the equation:   $n_t$,  $n_f$,  $\mathrm{P}^g(I)$,  $\mathrm{P}^{R_I}(I)$,  $\{p_s : s \subsetneq I\}$
    $solutions_I \leftarrow$ the solutions for $p_I$      // $|V \setminus I| \geq 1$ (possibly complex) solutions
    Try one of $solutions_I$, and advance to the next equation. If the next equation backtracks,
            try another solution. If all solutions in $solutions_I$ backtracked, then **backtrack**.
            If all equations were solved then **return** $\{p_s : s \subsetneq V\}$.
**end loop**
**return** "impossible"
```

$\mathrm{P}^g(I) = \mathrm{P}^R(I)$ but not before. Every parameter is associated with a body, which is a conjunction. If a parameter contains a ground variable not in I, it will not have an effect on $\mathrm{P}^R(I)$, and thus not appear. If it is a proper subset $I' \subsetneq I$, then the parameter already appeared before, in the equation for I'. □

The proof of Theorem 5 also shows that the number of possible parameters a BCF PLP has, i.e., the number of different rule bodies a PLP may have, is identical to the number of DoF a relational distribution has.

8 CONCLUSIONS

It is perhaps alarming for one to investigate much effort into uncovering why a learning algorithm fails to learn a probabilistic model from data, even given very large amounts of data and a large number of very flexible rules, only to discover that the fault is not the algorithm's; rather, the underlying language is simply incapable of representing the distribution of the data, even approximately.

Before using a language for an application domain, it is therefore advisable to become aware of any limitations the language may have.

A concerning result (Buchman & Poole, 2016, 2017) has recently emerged, demonstrating cases where PLPs are incapable of representing some relational distributions, in which $\neg$'s do not help, but using negative rule probabilities allows the distributions to be represented.

We follow in their footsteps, and investigate general relational settings (with fixed populations). We discovered that, even for a population of size four, PLPs are lacking in their expressiveness, and negative probabilities (or probabilities above 1) do not solve the problem; however, allowing complex probabilities makes PLPs "complete", allowing PLPs to represent any distribution (with probability 1). This is due to a mathematical structure underlying PLPs, in which rule probabilities are determined as solutions of polynomial equations.

We have also provided an algorithm for directly computing the probability of an interpretation. The algorithm is efficient when few variables are $true$. Extending this algorithm, and investigating its usefulness in various applications, is beyond the scope of this paper.

After realizing that complex probabilities are unavoidable, we investigate the question: what is a natural canonical model for PLPs? We expose BCF – a simplified language for representing PLPs, that strips the redundancy among PLP rules by tying their weights (thus avoiding redundant parameters), while maintaining full expressiveness. We also provide solver algorithms, showing how arbitrary relational distributions can be converted into BCF PLPs.

This work raises a few intriguing questions, that suggest that further deep insights may be around the corner. These include questions such as: What is the meaning of complex probabilities? What insights or intuitions are there, for the general interaction of rules with complex probabilities? And: what are further insights and intuitions for BCF? Is there a complimentary (dual) canonical form to BCF, which is more intuitive?

References

Buchman, David and Poole, David. Representing aggregators in relational probabilistic models. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

Buchman, David and Poole, David. Negation without negation in probabilistic logic programming. In *KR*, pp. 529–532, 2016.

Buchman, David and Poole, David. Negative probabilities in probabilistic logic programs. *International Journal of Approximate Reasoning*, 83:43–59, 2017.

Buchman, David, Schmidt, Mark W., Mohamed, Shakir, Poole, David, and de Freitas, Nando. On sparse, spectral and other parameterizations of binary probabilistic models. *Journal of Machine Learning Research - Proceedings Track*, 22:173–181, 2012.

Cox, David R. A use of complex probabilities in the theory of stochastic processes. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 51, pp. 313–319. Cambridge University Press, 1955.

De Raedt, L., Kimmig, A., and Toivonen, H. ProbLog: A probabilistic Prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pp. 2462–2467, 2007.

Dıez, Francisco J and Galán, Severino F. An efficient factorization for the noisy max. *International Journal of Intelligent Systems*, 18(2):165–177, 2003.

Hamber, Herbert W and Ren, Hai-cang. Complex probabilities and the Langevin equation. *Physics Letters B*, 159(4-6):330–334, 1985.

Hofmann, Holger F. Complex joint probabilities as expressions of reversible transformations in quantum mechanics. *New Journal of Physics*, 14(4):043031, 2012.

Kisynski, Jacek and Poole, David. Lifted aggregation in directed first-order probabilistic models. In *Proc. Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 1922–1929, Pasadena, California, 2009.

Kowalski, Robert. *Logic for Problem Solving, Revisited*. BoD–Books on Demand, 2014.

Meert, Wannes and Vennekens, Joost. Inhibited effects in cp-logic. In *European Workshop on Probabilistic Graphical Models*, pp. 350–365. Springer, 2014.

Parisi, Giorgio. On complex probabilities. *Physics Letters B*, 131(4-6):393–395, 1983.

Poole, D. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94:7–56, 1997. special issue on economic principles of multi-agent systems.

Poole, D. Abducing through negation as failure: stable models in the Independent Choice Logic. *Journal of Logic Programming*, 44(1–3):5–35, 2000.

Salcedo, LL. Representation of complex probabilities. *Journal of Mathematical Physics*, 38(3):1710–1722, 1997.

Salcedo, LL. Gibbs sampling of complex-valued distributions. *Physical Review D*, 94(7):074503, 2016.

Sato, T. and Kameya, Y. PRISM: A symbolic-statistical modeling language. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pp. 1330–1335, 1997.

Sato, T. and Kameya, Y. New advances in logic-based probabilistic modeling by PRISM. In De Raedt, L., Frasconi, P., Kersting, K., and Muggleton, S. (eds.), *Probabilistic Inductive Logic Programming*, volume LNCS 4911, pp. 118–155. Springer, 2008.

Van den Broeck, Guy, Meert, Wannes, and Darwiche, Adnan. Skolemization for weighted first-order model counting. *arXiv preprint arXiv:1312.5378*, 2013.

Vennekens, Joost, Denecker, Marc, and Bruynooghe, Maurice. CP-logic: A language of causal probabilistic events and its relation to logic programming. *TPLP*, 9 (3):245–308, 2009.

Zak, Michail. Incompatible stochastic processes and complex probabilities. *Physics Letters A*, 238(1):1–7, 1998.