Exploiting the Full Power of Pearl's Causality in Probabilistic Logic Programming

Kilian Rückschloß[1], Felix Weitkämper[1]

[1]Ludwig-Maximilians-Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Programmier- und Modellierungssprachen
Oettingenstraße 67
D-80538 München

Abstract

We introduce new semantics for acyclic probabilistic logic programs in terms of Pearl's functional causal models. Further, we show that our semantics is consistent with the classical distribution semantics and CP-logic. This enables us to establish all query types of functional causal models, namely probability calculus, predicting the effect of external interventions and counterfactual reasoning, within probabilistic logic programming. Finally, we briefly discuss the problems regarding knowledge representation and the structure learning task which result from the different semantics and query types.

Keywords

Counterfactual Reasoning, Functional Causal Models, Causal Bayesian Networks, Causal Structure Discovery, Distribution Semantics, CP-Logic, Probabilistic Logic Programming

1. Introduction

Intuitively, an acyclic probabilistic logic program under Problog semantics defines a distribution by solving a system of equations in terms of mutually independent predefined Boolean random variables. This intuition however is currently not reflected in the official distribution semantics of probabilistic logic programming. [1, §2.1] To illustrate this issue we introduce a version of the sprinkler example from [2, §1.4]:

Consider a road, which passes along a field with a sprinkler in it. The sprinkler is switched on, written $sprinkler$, by a weather sensor with probability $\pi_2 := 0.7$ in spring or summer, denoted by szn_spr_sum. Moreover, it rains, denoted by $rain$, with probability $\pi_3 := 0.1$ in spring or summer and with probability $\pi_4 := 0.6$ in fall or winter. If it rains or the sprinkler is on, the pavement of the road gets wet, denoted by wet. And in the case where the pavement is wet we observe that the road is slippery, denoted by $slippery$.

To model the situation above we generate mutually independent Boolean random variables $u1$-$u4$ with $\pi(u1) = 0.5$ and with $\pi(ui) = \pi_i$ for all $2 \leq i \leq 4$. The described

PLP 2022: The 9th Workshop on Probabilistic Logic Programming, August 1st, 2022, Haifa, Israel
kilian.rueckschloss@lmu.de (K. Rückschloß); felix.weitkaemper@lmu.de (F. Weitkämper)
https://www.pms.ifi.lmu.de/mitarbeiter/derzeitige/kilian-rueckschloss/ (K. Rückschloß);
https://www.pms.ifi.lmu.de/mitarbeiter/derzeitige/felix-weitkaemper/index.html (F. Weitkämper)
0000-0002-7891-6030 (K. Rückschloß); 0000-0002-3895-8279 (F. Weitkämper)

mechanism is then represented by the following system of equations:

$$
\begin{aligned}
szn\_spr\_sum &:= u1 \\
sprinkler &:= szn\_spr\_sum \wedge u2 \\
rain &:= (szn\_spr\_sum \wedge u3) \vee (\neg szn\_spr\_sum \wedge u4) \\
wet &:= (rain \vee sprinkler) \\
slippery &:= wet
\end{aligned} \tag{1}
$$

Further, with the usual reading of (probabilistic) logic programs one would encode (1) into the following LPAD-program **P**:

```
u1:0.5. u2:0.7. u3:0.1. u4:0.6.
szn_spr_sum :- u1.
sprinkler :- szn_spr_sum, u2.
rain :- szn_spr_sum, u3.
rain :- \+szn_spr_sum, u4.
wet :- rain.
wet :- sprinkler.
slippery :- wet.
```

However, neither the distribution semantics [1, §2.1] nor CP-logic [3] link the program **P** to the system of equations (1). Note for instance that these semantics do not introduce any random variables.

Similar to [4] the current paper uses the framework of Pearls functional causal models [2, §1.4] to obtain a new semantics for probabilistic logic programs. As a consequence of our new semantics we obtain an easy correspondence between probabilistic logic programs and causal Bayesian networks. Moreover, we are able to answer counterfactual queries with the theory discussed in [2, 1.4.4]:

Let us for instance assume we observe that the sprinkler is on and that the road is slippery. We would like to answer the query: "What is the probability of the road being slippery if the sprinkler were off?". In order to do so we need to update the distribution of the error terms ui in our program **P** according to the evidence $sprinkler$ and $slippery$ in a first step. Note that our observation may introduce new dependencies! Hence, we encode the joint distribution of the error terms $u1 - u4$ explicitly. To this aim we introduce new meta error terms $v1, v11, v01, ..., v1111, ..., v0001$, where $v01...1$ encodes the event that un is true if we observe that $u1$ is false, $u2$ is true and so on, i.e. we find

```
un :- \+u1,u2,...,v01...1.
```

Further, we update the probabilities of the meta error terms according to our observations. In particular, we recognize that it is spring or summer because the sprinkler is observed to be on, i.e. we replace $v1 : 0.5$ by the fact $v1$. In a second step we erase the clause with $sprinkler$ in the head to assure that the sprinkler is off. Finally, we query the modified program $\mathbf{P}'$ for $\pi(slippery)$ to obtain 0.1 for the desired probability.

We are not aware of any other semantics for probabilistic logic programming, which is able to answer counterfactual queries using Pearl's theory of causality as we just did.

However, there also exists a notion of counterfactual query in CP-logic [5]. In a future paper, we want to prove that these two approaches to counterfactual reasoning are equivalent, which means that CP-logic indeed consistently generalizes Pearl's theory of causality. As the theory of causal structure discovery is written in the language of causal Bayesian networks and causal models our results can also be seen as a first step towards structure learning of probabilistic logic programs under their causal semantics.

Our presentation begins with recalling Pearl's functional causal models and the distribution semantics in Section 2. Further, in Section 3 we define our new semantics and show that it generalizes the distribution semantics. Moreover, we prove that our semantics yield a notion of intervention that is consistent with the one in CP-logic. In Section 4 we discuss how counterfactual reasoning can be realized within probabilistic logic programming. Finally, Section 6 closes this paper with a discussion of our results.

2. Preliminaries

Before we begin with the presentation of our results, we introduce Pearl's functional causal models, which are the target structures for our new semantics and we recall the classical distribution semantics for probabilistic logic programming.

2.1. Pearl's Functional Causal Models

Let us recall the definition of a functional causal model from [2, §1.4.1]:

Definition 1 (Causal Model)**.** *A **functional causal model** or **causal model** $\mathcal{M}$ on a set of variables **V** is a system of equations, which consists of one equation of the form*

$$X := f_X(\mathrm{pa}(X), \mathrm{error}(X)) \tag{2}$$

*for each variable $X \in \mathbf{V}$. Here the **parents** $\mathrm{pa}(X) \subseteq \mathbf{V}$ of X form a subset of the set of variables **V**, the **error term** $\mathrm{error}(X)$ of X is a vector of random variables and f_X is a function defining X in terms of the parents $pa(X)$ and the error term $\mathrm{error}(X)$ of X.*

Example 1. *The system of equations (1) from Section 1 forms a causal model on the set of variables $\mathbf{V} := \{szn\_spr\_sum, rain, sprinkler, wet, slippery\}$ if we define $\mathrm{error}(szn\_spr\_sum) := u1$, $\mathrm{error}(sprinkler) := u2$, $\mathrm{error}(rain) := \begin{pmatrix} u3 \\ u4 \end{pmatrix}$.*

Next, we note that causal models do not only support queries about conditional and unconditional probabilities. They come along with two other more general query types, namely determining the effect of interventions and counterfactuals. To proceed we fix a causal model $\mathcal{M}$ on a set of variables $\mathbf{V}$.

Assume we are given a subset of variables $\mathbf{X} := \{X_1, ..., X_k\} \subseteq \mathbf{V}$ together with vector of possible values $\mathbf{x} := (x_1, ..., x_k)$ for the variables in $\mathbf{X}$. In order to model the effect of setting the variables in $\mathbf{X}$ to the values specified by $\mathbf{x}$ we simply replace the equations for X_i in $\mathcal{M}$ by $X_i := x_i$ for all $1 \leq i \leq k$ and calculate the desired probability. The resulting probability distribution is then denoted by $\pi(\_ | \,\mathrm{do}(\mathbf{X} = \mathbf{x}))$.[2, §1.4.3]

Example 2. *To predict the effect of switching the sprinkler on in the causal model (1) we simply replace the equation for $sprinkler$ by $sprinkler := True$.*

Finally, let $\mathbf{E} \subseteq \mathbf{V}$ and let $\mathbf{X} \subseteq \mathbf{V}$ be two subsets of our set of variables $\mathbf{V}$. Now suppose we observe the evidence that $\mathbf{E} = \mathbf{e}$ and we ask ourselves what would have been happened if we had set $\mathbf{X} = \mathbf{x}$. Note that in general $\mathbf{X} = \mathbf{x}$ and $\mathbf{E} = \mathbf{e}$ contradict each other. To answer this query we proceed in three steps: In the **abduction** step we adjust the distribution of our error terms by replacing the distribution $\pi(\mathrm{error}(V))$ with the conditional distribution $\pi(\mathrm{error}(V)|\mathbf{E} = \mathbf{e})$ for all variables $V \in \mathbf{V}$. Next, in the **action** step we intervene in the resulting model according to $\mathbf{X} = \mathbf{x}$. Finally, we are able to compute the desired probabilities from the modified model in the **prediction** step. [2, 1.4.4] For an illustration of the treatment of counterfactuals we refer to the introduction.

2.2. Causal Models and Bayesian Networks

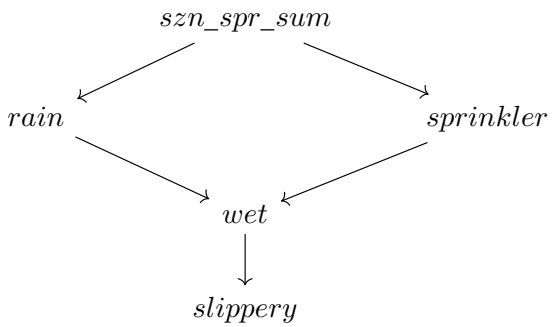
Next, we compare causal models with the widespread formalism of Bayesian networks. Recall, that a Bayesian network consists of a directed acyclic graph G on the involved random variables $\mathbf{V}$ and of the conditional probability distributions $\pi(X|\,\mathrm{pa}(X))$ for every $X \in \mathbf{V}$, where $\mathrm{pa}(X)$ denotes the set of parents of X in G. Further, a given distribution on $\mathbf{V}$ can be stored in a Bayesian network on the graph G if the **Markov condition** is satisfied, i.e. if each $X \in \mathbf{V}$ is independent of its predecessors conditioned on its parents $\mathrm{pa}(X)$. This motivates the following definition:

Definition 2 (Acyclic & Markovian Causal Models)**.** *For each causal model $\mathcal{M}$ on $\mathbf{V}$ we define the **causal diagram** $\mathrm{graph}(\mathcal{M})$ to be the directed graph on $\mathbf{V}$, which is given by drawing an arrow $X \to Y$ if and only if $X \in \mathrm{pa}(Y)$. Further, we call $\mathcal{M}$ an **acyclic** causal model if its causal diagram $\mathrm{graph}(\mathcal{M})$ is a directed acyclic graph. Finally, an acyclic causal model is said to be **Markovian** if the error terms $\mathrm{error}(X)$ are mutually independent for all $X \in \mathbf{V}$.*

The next result from [2, §1.4.2] links Markovian causal models to the theory of Bayesian networks:

Theorem 1. *Each Markovian causal model induces a distribution, which can be represented by Bayesian network on the associated causal digram.*

Example 3. *For the causal diagram of the causal model in Example 1 we obtain the following acyclic graph:*



(3)

As the error terms are mutually independent by construction we have a Markovian causal model, i.e. the resulting distribution can be represented by Bayesian network on the graph above.

Theorem 1 yields a way to determine effects of interventions from a Bayesian network structure. Assume we are given a Markovian causal model $\mathcal{M}$. We know that $\mathcal{M}$ gives rise to a Bayesian network $\mathcal{B}$ on the causal diagram $\mathrm{graph}(\mathcal{M})$. As it turns out to calculate the effect of setting $\mathbf{X} := \{X_1, ..., X_k\}$ to $\mathbf{x} := (x_1, ..., x_k)$ one can modify $\mathcal{B}$ by changing the distributions $\pi(X_i|\,\mathrm{pa}(X_i))$ to

$$\pi(X_i|\,\mathrm{pa}(X_i)) = \begin{cases} 1, & X_i = x_i \\ 0, & otherwise \end{cases} \text{ for all } 1 \leq i \leq k.$$

The resulting structure is called a **causal Bayesian network**, i.e. a causal Bayesian network is a Bayesian network that additionally supports queries about the effect of external interventions. [2, §1.3]

To summarize we are left with the following hierarchy: Probability distributions support the calculation of (conditional) probabilities. Causal Bayesian networks are more general because they are also able to predict the effect of interventions. Finally, causal models are the most general structure as they additionally provide the possibility of counterfactual reasoning. We also would like to highlight that each of these structures comes along with a corresponding learning problem if we want to derive a certain model from observational data.

2.3. Probabilistic Logic Programming under the Distribution Semantics

We proceed by recalling the classical semantics, the distribution semantics, of probabilistic logic programming. As the semantics of non-ground probabilistic logic programs is usually defined by grounding [6], we will focus on propositional probabilistic logic programs. Further, note that this paper uses LPAD-notation.

2.3.1. Syntax of Probabilistic Logic Programs

Before we discuss the semantics of a probabilistic logic program, we need to introduce it's syntax. Here, we construct a (probabilistic) logic program from a propositional alphabet $\mathfrak{P}$:

Definition 3 (propositional alphabet)**.** *A **propositional alphabet** $\mathfrak{P}$ is a finite set of **propositions** together with a subset $\mathfrak{E}(\mathfrak{P}) \subseteq \mathfrak{P}$ of **external propositions**. Further, we define $\mathfrak{I}(\mathfrak{P}) := \mathfrak{P} \setminus \mathfrak{E}(\mathfrak{P})$ to be the set of **internal propositions**.*

Example 4. *To build the program **P** in Section 1 we need the alphabet $\mathfrak{P}$ with internal propositions $\mathfrak{I}(\mathfrak{P}) := \{szn\_spr\_sum, sprinkler, rain, wet, slippery\}$ and with the external propositions $\mathfrak{E}(\mathfrak{P}) := \{u1, u2, u3, u4\}$.*

From propositional alphabets we built literals, clauses and random facts. Random facts are used to specify the probabilities in our model. To proceed let us fix a propositional alphabet $\mathfrak{P}$.

Definition 4 (Literals and Clauses)**.**

*i) A **literal** l is an expression p or $\neg p$ for a proposition $p \in \mathfrak{P}$. We call l a **positive** literal if it is of the form p and a **negative** literal if it is of the form $\neg p$.*

*ii) A **clause** LC is an expression of the form $h \leftarrow b_1, ..., b_n$, where $\mathrm{head}(LC) := h \in \mathfrak{I}(\mathfrak{P})$ is an internal proposition and where $\mathrm{body}(LC) := \{b_1, ..., b_n\}$ is a finite set of literals.*

*iii) A **random fact** RF is an expression of the form $u(RF) : \pi(RF)$, where $u(RF) \in \mathfrak{E}(\mathfrak{P})$ is an external proposition and where $\pi(RF) \in [0, 1]$ is a number called the **probability** of $u(RF)$.*

Example 5. *In Example 4 we have that szn_spr_sum is a positive literal, whereas $\neg szn\_spr\_sum$ is a negative literal. Further, $rain \leftarrow \neg szn\_spr\_sum, u4$ is a clause and $u4 : 0.6$ is a random fact.*

Next, we define logic programs and probabilistc logic programs as follows:

Definition 5 (Logic Programs and Probabilistic Logic Program)**.**

*i) A **logic program $\boldsymbol{P}$** is a finite set of clauses. We call such a program **$\boldsymbol{P}$ acyclic** if there exists a **level function** $\lambda : \mathfrak{P} \to \mathbb{N}$, which satisfies the following assertion for each clause $LC \in \boldsymbol{P}$: $\forall_{L \in \mathrm{body}(LC)} : \lambda(\mathrm{head}(LC)) > \lambda(L)$. (Here we use the convention $\lambda(\neg p) = \lambda(p)$ for every proposition $p \in \mathfrak{P}$.)*

*ii) A **probabilistic logic program $\boldsymbol{P}$** is given by a logic program $\mathrm{LP}(\boldsymbol{P})$ and set $\mathrm{Facts}(\boldsymbol{P})$, which consists of an unique random fact for every external proposition. We call $\mathrm{LP}(\boldsymbol{P})$ the **underlying logic program** of **$\boldsymbol{P}$**. Further, we call **$\boldsymbol{P}$ acyclic** if $\mathrm{LP}(\boldsymbol{P})$ is an acyclic logic program.*
*Finally, we define the **dependency graph** $\mathrm{graph}(\boldsymbol{P})$ of **$\boldsymbol{P}$** to be the directed graph on the set of internal propositions $\mathfrak{I}(\mathfrak{P})$, which is given by drawing an arrow $p \to q$ if and only if there exists a clause $LC \in \mathrm{LP}(\boldsymbol{P})$ with $\mathrm{head}(LC) = q$ and such that $\{p, \neg p\} \cap \mathrm{body}(LC) \neq \emptyset$.*

Notation 1. *To reflect the closed world assumption we will omit random facts of the form $u : 0$ in the set $\mathrm{Facts}(\boldsymbol{P})$.*

Example 6. *The program **$\boldsymbol{P}$** from the introduction is an acyclic probabilistic logic program. We obtain the corresponding underlying logic program $\mathrm{LP}(\boldsymbol{P})$ by erasing all random facts of the form $ui : \_$ from **$\boldsymbol{P}$**. Further, the dependency graph $\mathrm{graph}(\boldsymbol{P})$ of **$\boldsymbol{P}$** is given by (3).*

Finally, we define formulas as usual in propositional logic:

Definition 6. *Let $\mathfrak{Q} \subseteq \mathfrak{P}$ be a set of propositions. A **$\mathfrak{Q}$-formula** is inductively defined by:*

- *Each proposition $p \in \mathfrak{Q}$ is a $\mathfrak{Q}$-formula.*
- *For each formula ϕ we have that $\neg(\phi)$ is a $\mathfrak{Q}$-formula.*
- *For any two formulas ϕ and ψ we have that $(\phi \wedge \psi)$ is a $\mathfrak{Q}$-formula.*
- *For any two formulas ϕ and ψ we have that $(\phi \vee \psi)$ is a $\mathfrak{Q}$-formula.*
- *For any two formulas ϕ and ψ we have that $(\phi \leftrightarrow \psi)$ is a $\mathfrak{Q}$-formula.*

2.3.2. The Semantics of Propositions and Formulas

We will use Clark translation to define the semantics of acyclic logic programs, i.e. we define the semantics of logic programs via the semantics of propositions and formulas. To this aim we fix a set of propositions $\mathfrak{Q} \subseteq \mathfrak{P}$.

Definition 7. *A $\mathfrak{Q}$-**structure** is a function $\mathcal{M} : \mathfrak{Q} \to \{True, False\}, p \mapsto p^{\mathcal{M}}$.*

Next, we give a meaning to every $\mathfrak{Q}$-formula in the usual way:

Definition 8. *For a $\mathfrak{Q}$-structure $\mathcal{M}$ the **models relation** $\models$ is defined inductively over the structure of a $\mathfrak{Q}$-formula:*

- *For each proposition $p \in \mathfrak{Q}$ we write $\mathcal{M} \models p$ if and only if $p^{\mathcal{M}} = True$.*
- *For each formula ϕ we write $\mathcal{M} \models \neg(\phi)$ if and only if not $\mathcal{M} \models \phi$, i.e. if $\mathcal{M} \not\models \phi$.*
- *For any two formulas ϕ and ψ we write $\mathcal{M} \models (\phi \wedge \psi)$ if and only if $\mathcal{M} \models \phi$ and $\mathcal{M} \models \psi$.*
- *For any two formulas ϕ and ψ we write $\mathcal{M} \models (\phi \vee \psi)$ if and only if $\mathcal{M} \models \phi$ or $\mathcal{M} \models \psi$.*
- *For any two formulas ϕ and ψ we write $\mathcal{M} \models (\phi \leftrightarrow \psi)$ if and only if either $\mathcal{M} \models \phi$ and $\mathcal{M} \models \psi$ or $\mathcal{M} \not\models \phi$ and $\mathcal{M} \not\models \psi$.*

Now we can give a semantics to acyclic logic programs.

2.3.3. The Semantics of Acyclic Logic Programs

As already mentioned we define the semantics of an acyclic logic program by using Clark translation.

Definition 9 (Clark Translation)**.** *For each logic program $\boldsymbol{P}$ we define it's **Clark translation** by*

$$\boldsymbol{P}^{CL} := \left\{ \left(p \leftrightarrow \bigvee_{\substack{LC \in \boldsymbol{P} \\ \mathrm{head}(LC) = p}} \left(\bigwedge_{l \in \mathrm{body}(LC)} l \right) \right) : p \in \mathfrak{I}(\mathfrak{P}) \right\}$$

with the convention that an empty disjunction evaluates to $False$ and that the empty conjunction evaluates to $True$.

Example 7. *One key observation is that the Clark translation of the underlying logic program $\mathrm{LP}(\boldsymbol{P})$ of the probabilistic logic program $\boldsymbol{P}$ in Example 6 can be obtained from the system of equations (1) by replaceing the ":="-sign by the "$\leftrightarrow$"-sign.*

In the next proposition we formally justify, why it is permissible to exchange the ":="-sign by the "$\leftrightarrow$"-sign.

Proposition 1. *Let $\boldsymbol{P}$ be an acyclic logic program.*

i) For each $\mathfrak{E}(\mathfrak{P})$-structure $\mathcal{E}$ there exists an unique $\mathfrak{P}$-structure $\mathcal{M}(\mathcal{E}, \boldsymbol{P})$ such that $\mathcal{M}(\mathcal{E}, \boldsymbol{P})$ extends $\mathcal{E}$ and such that $\mathcal{M}(\mathcal{E}, \boldsymbol{P}) \models \boldsymbol{P}^{CL}$.

ii) We have that $\mathcal{M}(\mathcal{E}, \boldsymbol{P})$ is obtained from $\mathcal{E}$ by solving the system of equations:

$$\mathrm{ES}(\boldsymbol{P}) := \left\{ p := \bigvee_{\substack{LC \in \boldsymbol{P} \\ \mathrm{head}(LC)=p}} \left(\bigwedge_{l \in \mathrm{body}(LC)} l \right) : p \in \mathfrak{I}(\mathfrak{P}) \right\}$$

Proof. Let $\mathbf{P}$ be an acyclic logic program. First, we observe that i) trivially follows from ii) by the definition of $\models$. Hence, all we need to show is that $\mathrm{ES}(\mathbf{P})$ posses an unique solution for every $\mathfrak{E}(\mathfrak{P})$-structure $\mathcal{E}$.

Let $\mathcal{E}$ be an $\mathfrak{E}(\mathfrak{P})$-structure. As $\mathbf{P}$ is acyclic there exists a level function $\lambda : \mathfrak{P} \to \mathbb{N}$. Let us proceed by induction over the number n of the values of λ.

n=1 In this case for every clause $LC \in \mathbf{P}$ we have that $\mathrm{body}(LC) = \emptyset$. Hence, the only $\mathfrak{P}$-structure $\mathcal{M}$ extending $\mathcal{E}$ and satisfying $\mathrm{ES}(\mathbf{P})$ is given by

$$\mathcal{P} : \mathfrak{P} \to \{True, False\}$$
$$p \mapsto \begin{cases} e^{\mathcal{E}}, & e \in \mathfrak{E}(\mathfrak{P}) \\ True, & p \in \mathfrak{I}(\mathfrak{P}) \text{ and } p = \mathrm{head}(LC) \text{ for a } LC \in \mathbf{P} \\ False, & otherwise \end{cases}.$$

n>1 Denote by $\mathfrak{P}^{\max}$ the set of all propositions p with $\lambda(p) = \max_{p \in \mathfrak{P}} \lambda(p)$. Further, denote by $\mathbf{P}^{\max}$ the set of all clauses $LC \in \mathbf{P}$ with $\mathrm{head}(LC) \in \mathfrak{P}^{\max}$. Finally, set $\mathbf{P}' := \mathbf{P} \setminus \mathbf{P}^{\max}$ and set $\mathfrak{P}' := \mathfrak{P} \setminus \mathfrak{P}^{\max}$. Now by induction hypothesis there exists an unique $\mathfrak{P}'$-structure $\mathcal{M}' := \mathcal{M}(\mathcal{E}, \mathbf{P}')$ extending $\mathcal{E}$ and satisfying $\mathrm{ES}(\mathbf{P}')$. To conclude we observe that by acyclicity the only $\mathfrak{P}$-structure $\mathcal{M}$ extending $\mathcal{E}$ and satisfying $\mathrm{ES}(\mathbf{P})$ is given by

$$\mathcal{M} : \mathfrak{P} \to \{True, False\}$$
$$p \mapsto \begin{cases} p^{\mathcal{M}'}, & p \in \mathfrak{P}' \\ True, & p = \mathrm{head}(LC) \text{ and } \mathcal{M}' \models \mathrm{body}(LC) \text{ for a } LC \in \mathbf{P}^{\max} \\ False, & otherwise \end{cases}.$$

□

Remark 1. *We take the assignment $\mathcal{E} \mapsto \mathcal{M}(\mathcal{E}, \boldsymbol{P})$ as the semantics of an acyclic logic program $\boldsymbol{P}$. Note that this is consistent with the standard minimal Herbrand-model semantics. [7]*

Corollary 1. *Let $\boldsymbol{P}$ be an acyclic logic program. Each model of the Clarck translation $\mathcal{M} \models \boldsymbol{P}^{CL}$ is uniquely determined by it's restriction $\mathcal{M}|_{\mathfrak{E}(\mathfrak{P})}$ to the external propositions.*

Proof. This is a direct consequence of Proposition 1. □

2.3.4. The Distribution Semantics of Acyclic Probabilistic Logic Programs

Finally, we are in the position to give the definition of the distribution semantics for acyclic probabilistic logic programs:

Definition 10. *Let $\boldsymbol{P}$ be a probabilistic logic program. An **atomic choice** is a subset $\boldsymbol{C} \subseteq \mathrm{Facts}(\boldsymbol{P})$ of the random facts of $\boldsymbol{P}$. To each atomic choice $\boldsymbol{C}$ we associate the $\mathfrak{E}(\mathfrak{P})$-structure*

$$\begin{aligned}\mathcal{E}(\boldsymbol{C}) :&\mathfrak{E}(\mathfrak{P}) \to \{True, False\} \\ &e \mapsto \begin{cases} True, & \text{if there exists a random fact of the form } e : \_ \text{ in } \boldsymbol{C} \\ False, & otherwise \end{cases}\end{aligned}$$

and the probability

$$\pi(\boldsymbol{C}) := \prod_{RF \in \boldsymbol{C}} \pi(RF) \cdot \prod_{RF \notin \boldsymbol{C}} (1 - \pi(RF)).$$

*The **distribution semantics** of an acyclic probabilistic logic program $\boldsymbol{P}$ is the distribution $\pi_{\boldsymbol{P}}^{dist}$ on the $\mathfrak{P}$-structures $\mathcal{M}$, which is given by*

$$\pi_{\boldsymbol{P}}^{dist}(\mathcal{M}) := \begin{cases} \pi(\boldsymbol{C}), & \text{if } \mathcal{M} = \mathcal{M}(\mathcal{E}(\boldsymbol{C}), \mathrm{LP}(\boldsymbol{P})) \text{ for an atomic choice } \boldsymbol{C} \\ 0, & otherwise \end{cases}.$$

*Finally, for every $\mathfrak{P}$-formula ϕ we define the **probability** for ϕ to hold by*

$$\pi(\phi) := \sum_{\substack{\mathcal{M}\ \mathfrak{P}-structure \\ \mathcal{M} \models \phi}} \pi(\mathcal{M}). \tag{4}$$

Remark 2. *Remark 1 ensures that Definition 10 defines the distribution semantics as in [1, 2.1.2].*

3. The Functional Causal Models Semantics for Probabilistic Logic Programs

Since we are now finished with our preparations, we proceed to the functional causal models semantics we are heading for:

Definition 11. *Let $\boldsymbol{P}$ be an acyclic probabilistic logic program. We define the **functional causal models semantics** or **FCM semantics** of $\boldsymbol{P}$ to be the functional causal model on the internal propositions $\mathfrak{I}(\mathfrak{P})$, which is given by the system of equations*

$$\mathrm{FCM}(\boldsymbol{P}) := \left\{ p^{\mathrm{FCM}} := \bigvee_{\substack{LC \in \mathrm{LP}(\boldsymbol{P}) \\ \mathrm{head}(LC) = p}} \left(\bigwedge_{l \in \mathrm{body}(LC)} l^{\mathrm{FCM}} \right) : p \in \mathfrak{I}(\mathfrak{P}) \right\}$$

and mutually independent Boolean random variables $u(RF)^{\mathrm{FCM}}$ for every random fact $RF \in \mathrm{Facts}(\boldsymbol{P})$ that are distributed according to $\pi\left[u(RF)^{\mathrm{FCM}}\right] = \pi(RF)$. Here we again use the convention that an empty disjunction evaluates to $False$ and that an empty conjunction evaluates to $True$.

Remark 3. *By construction the dependency graph* $\mathrm{graph}(\boldsymbol{P})$ *of* $\boldsymbol{P}$ *yields the causal diagram* $\mathrm{graph}(\mathrm{FCM}(\boldsymbol{P}))$ *of the associated functional causal model.*

Remark 4. *The theory so far has been restricted to acyclic probabilistic logic programs, since the Clark translation is known not to be correct for general acyclic logic programs [7]. However, potentially cyclic programs with stratified negation can also be accommodated by* cycle-breaking. *This process replaces a propositional (or ground) logic program with an equivalent acyclic logic program. There are different algorithms available for this task, the current state-of-the-art being* treewidth-aware cycle-breaking *based on forward-unfolding of the logic program [8].*

Example 8. *As intended in the introduction the causal model (1) yields the FCM-semantics of the program* $\boldsymbol{P}$*.*

Next, we check that the FCM-semantics indeed yields a well-defined distribution.

Proposition 2. *Let* $\boldsymbol{P}$ *be an acyclic probabilistic logic program. The FCM-semantics assigns to each predicate* $p \in \mathfrak{P}$ *an unique random variable* p^{FCM}*.*

Proof. Similar to Proposition 1. □

Our next aim is to verify that the FCM-semantics yield a consistent generalization of the distributions semantics. For this purpose we introduce the following notation:

Notation 2. *Let* $\boldsymbol{P}$ *be an acyclic probabilistic logic program. For each* $\mathfrak{P}$*-structure* $\mathcal{M}$ *we write*

$$\pi^{\mathrm{FCM}}_{\boldsymbol{P}}(\mathcal{M}) := \pi\left[\left\{p^{\mathrm{FCM}} = \mathcal{M}(p) : p \in \mathfrak{P}\right\}\right].$$

Note that $\pi^{\mathrm{FCM}}_{\boldsymbol{P}}$ *encodes the joint distribution of the random variables* p^{FCM}*. Hence, the distribution of the random variables* p^{FCM} *is uniquely determined by* $\pi^{\mathrm{FCM}}_{\boldsymbol{P}}$*.*

With Notation 2 at hand we now proceed to the proof of the desired consistency:

Theorem 2. *For each acyclic probabilistic logic program* $\boldsymbol{P}$ *we have that* $\pi^{\mathrm{FCM}}_{\boldsymbol{P}} = \pi^{dist}_{\boldsymbol{P}}$*.*

Proof. We have to show that $\pi^{\mathrm{FCM}}_{\mathbf{P}}(\mathcal{M}) = \pi^{dist}_{\mathbf{P}}(\mathcal{M})$ for every $\mathfrak{P}$-structure $\mathcal{M}$. Denote by $\mathcal{E}$ the restriction of $\mathcal{M}$ to the external alphabet $\mathfrak{E}(\mathfrak{P})$ and by $\mathbf{C}$ the unique atomic choice with $\mathcal{E} = \mathcal{E}(\mathbf{C})$. Now observe that

$$\begin{aligned}
\pi^{\mathrm{FCM}}_{\mathbf{P}}(\mathcal{M}) &\overset{DEF}{=} \pi\left[\left\{p^{\mathrm{FCM}} = \mathcal{M}(p) \text{ for all } p \in \mathfrak{P}\right\}\right] \overset{Corollary\ 1}{=} \\
= \pi\left[\left\{e^{\mathrm{FCM}} = \mathcal{M}(e) : e \in \mathfrak{E}(\mathfrak{P})\right\}\right] &\overset{DEF}{=} \pi\left[\left\{e^{\mathrm{FCM}} = \mathcal{E}(e) : e \in \mathfrak{E}(\mathfrak{P})\right\}\right] \overset{DEF}{=} \\
= \prod_{RF \in \mathbf{C}} \pi(RF) \cdot \prod_{RF \notin \mathbf{C}} (1 - \pi(RF)) &\overset{DEF}{=} \pi(\mathbf{C}) = \pi^{dist}_{\mathbf{P}}(\mathcal{M}).
\end{aligned}$$

□

Corollary 2. *The FCM-sematics and the distribution sematics yield the same probability for every proposition in* $\mathfrak{P}$*.*

Proof. This follows by marginalization and equation (4). □

Further, recall from 2.1 that functional causal models do not only provide a probability distribution they also support two further query types, namely the prediction of the effects of interventions and counterfactual reasoning.

Predicting the Effect of External Interventions in FCM-semantics

Let $\mathbf{P}$ be a probabilistic logic program and let $\mathbf{X}_1, \mathbf{X}_2 \subseteq \mathfrak{I}(\mathfrak{P})$ be two subsets of internal propositions. Assume we would like to calculate the effect of setting the variables in $\mathbf{X}_1^{\mathrm{FCM}}$ to $True$ and the variables in $\mathbf{X}_2^{\mathrm{FCM}}$ to $False$. According to the discussion in 2.1 and Definition 11 the desired distribution can be calculated from a modified program $\mathbf{P}'$, which is obtained as follows:

1.) Replace every clause $LC \in \mathbf{P}$ with $\mathrm{head}(LC) \in \mathbf{X}_1$ by the clause $\mathrm{head}(LC) \leftarrow$.
2.) Erase every clause $LC \in \mathbf{P}$ with $\mathrm{head}(LC) \in \mathbf{X}_2$ form $\mathbf{P}$.

We observe that this is the same notion of intervention, which was introduced for CP-logic in [3, §7.5] and which is implemented in cplint [9]. As CP-logic is consistent with the distribution semantics Theorem 2 yields the following result:

Theorem 3. *The FCM-semantics consistently generalizes the notion of intervention in CP-logic.* □

Next, we want to use probabilistic logic programming as a language for causal Bayesian networks:

Definition 12. *We call an acyclic probabilistic logic program $\boldsymbol{P}$ **Markovian** if we obtain for every external proposition $u \in \mathfrak{E}(\mathfrak{P})$, which occurs in the body of a clause LC_1 that*

$$\forall_{LC_2 \in \mathrm{LP}(\boldsymbol{P})} : (\mathrm{head}(LC_1) \neq \mathrm{head}(LC_2) \Rightarrow \{u, \neg u\} \cap \mathrm{body}(LC_2) = \emptyset) .$$

Example 9. *The program $\boldsymbol{P}$ in the introduction yields a Markovian probabilistic logic program.*

Theorem 4 (Probabilistic Logic Programs & Causal Bayesian Networks)**.** *A Markovian probabilistic logic program $\boldsymbol{P}$, equipped with the above notion of intervention, gives rise to a Boolean causal Bayesian network $\mathcal{CB}(\boldsymbol{P})$ on the dependency graph $\mathrm{graph}(\boldsymbol{P})$ of $\boldsymbol{P}$.*

Proof. This follows directly from Definition 11, Remark 3 and Theorem 1. □

Definition 13 (Bayesian networks semantics)**.** *In the situation of Theorem 4 we call $\mathcal{CB}(\_)$ the **causal Bayesian network semantics** for Markovian probabilistic logic programs.*

Remark 5. *As the causal Bayesian network semantics allows for the computation of the effect of interventions it is a proper generalization of the distribution semantics.*

Next, we see that the causal Bayesian network semantics yields a proper generalization of the distribution semantics for probabilistic logic programs.

Example 10. *Consider the Markovian probabilistic logic programs $\boldsymbol{P}_1$, respectively $\boldsymbol{P}_2$ below:*

```
a :- b. b :- u. u :0.5.
```

```
b :- a. a :- u. u :0.5.
```

It is easy to observe that $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ determine the same distribution. However, intervening according to $a^{\mathrm{FCM}} = True$ leaves b^{FCM} unchanged in $\boldsymbol{P}_1$, while b^{FCM} becomes true in $\boldsymbol{P}_2$. Hence, $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are equivalent under the distribution semantics, whereas they are not equivalent under the causal Bayesian network semantics.

Considering Example 10, we see that from the classical LPAD-syntax it is not clear, which knowledge a probabilistic logic program actually states! Hence, one should be careful when using probabilistic logic programs for causal inference. In particular this observation has tremendous consequences for the structure learning theory of such programs:

Assume we want to learn the probabilistic logic program $\mathbf{P}_1$ from data. If we only claim to learn $\mathbf{P}_1$ under the distribution semantics, it is equally good to end up with $\mathbf{P}_2$. However, if we are interested in the effect of interventions, we need to distinguish between $\mathbf{P}_1$ and $\mathbf{P}_2$.

In the theory of Bayesian networks the task of learning a causal Bayesian network is referred to as causal structure discovery. One of our future goals is to develop a causal structure discovery algorithm for non-ground probabilistic logic programs.

4. Counterfactual Reasoning in Probabilistic Logic Programming

Finally, we also want to establish counterfactual reasoning in probabilistic logic programming. Assume that we are given an acyclic probabilistic logic program $\mathbf{P}$. Additionally, let $\mathbf{E} \subseteq \mathfrak{I}(\mathfrak{P})$ and $\mathbf{X} \subseteq \mathfrak{I}(\mathfrak{P})$ be two subsets of internal propositions. Now suppose we observe the evidence $\mathbf{E}^{\mathrm{FCM}} = \mathbf{e}$ and we ask ourselves what would have been happened if we set $\mathbf{X}^{\mathrm{FCM}} = \mathbf{x}$. In particular it may be that $\mathbf{E}^{\mathrm{FCM}} = \mathbf{e}$ and $\mathbf{X}^{\mathrm{FCM}} = \mathbf{x}$ contradict each other!

Recalling the discussion in 2.1 and Definition 11 we proceed as follows: First, in the abduction step, we build a modified program $\mathbf{P}'$, where we need to adjust the distribution of the external predicates $u1^{\mathrm{FCM}}$-un^{FCM} according to our observations. Note that in general given new evidence the random variables $u1^{\mathrm{FCM}}$-un^{FCM} are no longer mutually independent! Hence, we encode the full joint distribution of the ui in our program. To this aim we introduce 2^n new **meta external predicates** va, which we enumerate with sequences $(aj) \in \{0,1\}^m$ with $m \leq n$ and $am = 1$. The sequence (aj) encodes the event that um^{FCM} is true once we observe uj if $aj = 1$ and $\neg uj$ if $aj = 0$. From Theorem 4 we see that the distribution on the ui can be stored in the following program $\mathbf{P}_0$:

```
v1 : p1. %u1:p1 random fact in P
v11 : p2. v11 : p2. %u2:p2 random fact in P
...
v11...1 : pn. ... v0...01 : pn. %un : pn random fact in P

u1 :- v1.
u2 :- u1,v11. u2 :- \+u1,v01.
...
un :- u1,u2,...,un-1,v11...1.
...
un :- \+u1,\+u2,...,\+un-1,v0...01.
```

Hence, the program $\mathbf{P}_0'$ that results by replacing the error terms of $\mathbf{P}$ with the program $\mathbf{P}_0$ yields the same distribution on the internal prdicates. Finally, we obtain the modified $\mathbf{P}'$ by replacing each random fact $va : \pi_a$ in the program $\mathbf{P}_0'$ with the random fact $va : \pi\,(va|\mathbf{e})$. We can then compute the desired probabilities by intervening according to $\mathbf{X}^{\mathrm{FCM}} = \mathbf{x}$. For a detailed example of the treatment of counterfactuals we refer the reader to the introduction. Note that there is also a notion of counterfactual reasoning in CP-logic [5]. We verify in a future paper that these to approaches to counterfactual reasoning in probabilistic logic programming lead to the same result.

Further, like the causal Bayesian network semantics generalizes the distribution semantics, we also find that the FCM-semantics properly generalizes the causal Bayesian network semantics.

Example 11. *Consider the probabilistic logic programs $\boldsymbol{P}_1$, respectively $\boldsymbol{P}_2$ below:*

```
u1:0.5. u2:0.5. u3:0.4.
treatment :- u1.
recovery :- u2.
recovery :- treatment, u3.
```

```
u1:0.5. u2:0.5. u3:0.7.
treatment :- u1.
recovery :- \+treatment, u2.
recovery :- treatment, u3.
```

According to Theorem 4 both programs give rise to a causal Bayesian network on the graph $treatment \rightarrow recovery$.

Observe that in $\boldsymbol{P}_1$, there is a baseline chance of 0.5 for recovering from an illness, and treatment gives an additional chance of 0.4 that a patient recovers from treatment that would not otherwise have recovered. Therefore, the probability of recovery under treatment is 0.7.

In $\boldsymbol{P}_2$, the recovery rates with and without treatment have been modelled separately, with a 0.7 recovery rate under treatment and a 0.5 rate without it.

Therefore the programs $\boldsymbol{P}_{1/2}$ yield the same probabilities for $\pi(treatment)$, $\pi(recovery|treatment)$ and $\pi(recovery|\neg treatment)$. Thus they share the same causal Bayesian network semantics and therefore the same distribution semantics.

Next, let us assume we observe that treatment is false and that recovery is true. Let Q be the query: "What is the probability of recovery if treatment was true?" In both programs $\boldsymbol{P}_{1/2}$ our observations imply that $u2$ is true. Hence, $\boldsymbol{P}_1$ answers the query Q with 1. For $\boldsymbol{P}_2$ we obtain that $\boldsymbol{P}_2'$ is equivalent to the following program:

```
v1:0.5.  v11 : 0.7. v01:0.7.
u1 :- v1. u2. u3 :- u1,v11. u3:-\+u1,v01.
treatment :- u1.
recovery :- \+treatment, u2.
recovery :- treatment, u3.
```

This means that $\boldsymbol{P}_2$ answers Q with a probability of 0.7.

Note that when modelling under the distribution semantics, one would often be tempted to prefer $\boldsymbol{P}_2$, since separating the possible situations makes it easier to see the probabilities

of recovery of each reference class at first glance. However, when using the resulting program for counterfactual reasoning, the FCM-semantics of $\boldsymbol{P}_2$ implies that recovery with and without treatment are completely independent. In particular, we obtain for every patient that recovered untreated a 0.3 probability that he would not have recovered if he had indeed received the treatment. This would be an extraordinary claim, and usually the interpretation encoded by $\boldsymbol{P}_1$, namely that treatment enhances any individual's chance of recovery, would be closer to the intended meaning.

5. Syntactic Sugar and Non-Ground Probabilistic Logic Programs

In probabilistic logic programming under Problog semantics, a special case of the distribution semantics [1, §2.1.2], one introduces the following syntactic sugar:

Definition 14 (Problog Program)**.** *A **Problog clause** RC is an expression of the form $h : \pi \leftarrow b_1, ..., b_n$ for a proposition $\mathrm{head}(RC) := h$, a real number $\pi(RC) := \pi \in [0, 1]$ and literals $\mathrm{body}(RC) := \{b_1, ..., b_n\}$. Further, a **Problog program** $\boldsymbol{P}$ is a finite set of Problog clauses.*

*For each Problog clause $RC \in \boldsymbol{P}$ we add a distinct external literal $u(RC)$ to our alphabet. In this case the **FCM-semantics** of $\boldsymbol{P}$ is given by the FCM-semantics of the program $\mathrm{PLP}(\boldsymbol{P})$, which is given by*

$$\mathrm{LP}(\mathrm{PLP}(\boldsymbol{P})) := \{\mathrm{head}(RC) \leftarrow \mathrm{body}(RC) \cup u(RC) : RC \in \boldsymbol{P}\}$$

$$\mathrm{Facts}(\mathrm{PLP}(\boldsymbol{P})) := \{u(RC) : \pi(RC) : RC \in \boldsymbol{P}\}.$$

Finally, we call $\boldsymbol{P}$ acyclic is $\mathrm{PLP}(\boldsymbol{P})$ is an acyclic probabilistic logic program.

Remark 6. *An acyclic Problog program is automatically Markovian, i.e. it posses a well-defined causal Bayesian network semantics.*

Example 12. *The causal model in Example 1 can be defined using the following Problog program:*

```
szn_spr_sum:0.5.
sprinkler:0.7 :- szn_spr_sum.
rain:0.1 :- szn_spr_sum.
rain:0.6 :- \+szn_spr_sum.
wet:1 :- rain.
wet:1 :- sprinkler.
slippery:1 :- wet.
```

Finally, non-ground probabilistic logic programs can be considered as Problog programs built over relational alphabets. Instantiating such a non-ground program with a given domain then yields a propositional program to which the results of this paper can be applied. Hence, the FCM-semantics for probabilistic logic programs and all results of this paper easily generalize to the non-ground case. In particular by Remark 6 all non-ground programs constructed in this way possess a well-defined causal Bayesian network semantics.

6. Conclusion

In this paper we present an interpretation of acyclic probabilistic logic programs in terms of Pearl's functional causal models. As a consequence we are able to establish the three query types of functional causal models, namely probability calculus, prediction of the effect of interventions and counterfactual reasoning in the framework of probabilistic logic programming.

Moreover, we see that each of these query types comes with its own semantics and that it is not clear from the syntax which knowledge a given probabilistic logic program actually states. Hence, one should be careful when using a probabilistic logic program for causal queries.

Finally, we highlight that each of those semantics yields a new structure learning task. This brings the subject of causal structure discovery to the realm of probabilistic logic programming. In the future we aim to develop causal structure discovery algorithms for learning non-ground probabilistic logic programs under causal Bayesian network and FCM-semantics.

Acknowledgments

The research leading to this publication was supported by LMUexcellent, funded by the Federal Ministry of Education and Research (BMBF) and the Free State of Bavaria under the Excellence Strategy of the Federal Government and the Länder.

The athosr would like to thank Kailin Sun for proofreading the initial submission.

References

[1] F. Riguzzi, Foundations of Probabilistic Logic Programming: Languages, Semantics, Inference and Learning, River Publishers, 2020.

[2] J. Pearl, Causality, 2 ed., Cambridge University Press, Cambridge, UK, 2000. doi:10.1017/CBO9780511803161.

[3] J. Vennekens, M. Denecker, M. Bruynooghe, CP-logic: A Language of Causal Probabilistic Events and Its Relation to Logic Programming, Theory Pract. Log. Program. 9 (2009) 245–308. doi:10.1017/S1471068409003767.

[4] A. Bochman, V. Lifschitz, Pearl's causality in a logical setting, in: B. Bonet, S. Koenig (Eds.), Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, AAAI Press, 2015, pp. 1446–1452. doi:10.1609/aaai.v29i1.9411.

[5] J. Vennekens, M. Bruynooghe, M. Denecker, Embracing events in causal modelling: Interventions and counterfactuals in cp-logic, in: T. Janhunen, I. Niemelä (Eds.), Logics in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 313–325.

[6] J. Vennekens, S. Verbaeten, Logic Programs with Annotated Disjunctions, Technical Report CW 368, Katholieke Universiteit Leuven, Department of Computer Science, 2003. URL: https://www.cs.kuleuven.be/publicaties/rapporten/cw/CW368.abs.html.

[7] F. Fages, Consistency of clark's completion and existence of stable models., Meth. of Logic in CS 1 (1994) 51–60.

[8] T. Eiter, M. Hecher, R. Kiesel, Treewidth-aware cycle breaking for algebraic answer set counting, in: M. Bienvenu, G. Lakemeyer, E. Erdem (Eds.), Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021, 2021, pp. 269–279. doi:10.24963/kr.2021/26.

[9] F. Riguzzi, G. Cota, E. Bellodi, R. Zese, Causal inference in cplint, International Journal of Approximate Reasoning 91 (2017) 216–232. doi:10.1016/j.ijar.2017.09.007.