Reasoning about Independence in Open Universe Probabilistic Logic Programs

Kilian Rückschloß, Felix Weitkämper

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

Abstract

In the 1980's J. Pearl and T. Verma [1] developed a graphical criterion to reason about probabilistic independence in Bayesian networks, which is famously known as d-separation. As J. Vennekens and S. Verbaeten [2, §5.2] pointed out in 2003, ground acyclic probabilistic logic programs correspond to Bayesian networks. Hence, we can pass from an acyclic ground program to the associated Bayesian network in order to derive independence statements from d-separation. In the present paper we lift this procedure so as to reason about independence in open universe probabilistic logic programs.

1. Introduction

The aim of this paper is to establish an analogue of the graphical independence analysis in Bayesian networks (BN) for open universe probabilistic logic programs (PLPs). A BN stores a probability distribution in the form of a directed acyclic graph (DAG) on the involved random variables together with a table of conditional probabilities for every node in this graph. J. Pearl and T. Verma [1] developed the following criterion to derive conditional independence statements, only considering the DAG, underlying a BN:

Definition 1 (d-Separation). *Let G be a DAG and let $\boldsymbol{Z}$ be a set of nodes. Moreover, let $P := X - ... - Y$ be an undirected path in G. We call P a **d-connecting path** with respect to $\boldsymbol{Z}$ if the following assertions hold for every triple $... - R_1 - R_2 - R_3 - ...$ in P:*

i) If $R_1 \leftarrow R_2$ or $R_2 \rightarrow R_3$ in G, we have that $R_2 \notin \boldsymbol{Z}$.
ii) If $R_1 \rightarrow R_2 \leftarrow R_3$ in G, there exists a directed path $R_2 \rightarrow ... \rightarrow Z$ from R_2 to a $Z \in \boldsymbol{Z}$.

*Two sets $\boldsymbol{X}$ and $\boldsymbol{Y}$ of nodes in G are said to be **d-separated** by $\boldsymbol{Z}$ if there exists no d-connecting path between a $X \in \boldsymbol{X}$ and a $Y \in \boldsymbol{Y}$.*

Remark 1. *The "d" in the term term "d-separation" is a shorthand for "directional". ([3])*

The intuition behind this definition is that d-connecting paths indicate dependencies and indeed one obtains:

PLP 2021: The 8th Workshop on Probabilistic Logic Programming, September 20-27, 2021, Porto
kilian.rueckschloss@lmu.de (K. Rückschloß); felix.weitkaemper@lmu.de (F. Weitkämper)
© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-2970/plppaper3.pdf

Theorem 1. *Consider a BN with underlying DAG G and let $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ be sets of random variables. If $\boldsymbol{X}$ and $\boldsymbol{Y}$ are d-separated by $\boldsymbol{Z}$ in G, then $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent, conditioned on $\boldsymbol{Z}$.*

Remark 2. *Note that the converse of Theorem 1 fails in general. It is referred to as causal faithfulness assumption and plays a central role in causal structure discovery [4, §4]. We believe that formulating and verifying the causal faithfulness assumption for open universe PLPs would be an interesting direction for future work.*

Example 1. *Assume we are given a BN with underlying DAG $h_1 \to h_2 \to h_3$. In this case theorem 1 yields without further calculations that h_1 and h_3 become independent, once we observe h_2.*

In probabilistic logic programming (PLP) J. Vennekens and S. Verbaeten established a correspondence between ground acyclic programs and BNs [2, §5.2]. In particular, a ground acyclic program without function symbols induces a distribution, which can be stored in a Boolean BN on the dependency graph. This gives us the possibility to derive statements about independence, only considering the program structure:

Example 2. *Let us for instance consider the following acyclic ground LPAD-program:*

$$h_1 : \alpha_1 \leftarrow \qquad\qquad h_2 : \alpha_2 \leftarrow h_1 \qquad\qquad h_3 : \alpha_3 \leftarrow h_2$$

We see that the induced distribution can be stored in a BN with underlying graph $h_1 \to h_2 \to h_3$, i.e. by example 1 we can immediately conclude that h_1 and h_3 become independent if we observe h_2.

In this paper we want to transfer the reasoning of Example 2 to open universe PLPs, which then correspond to families of BNs. Since the field of causal structure discovery for BNs heavily relies on d-separation ([5], [6], [7]), we believe that such an independence analysis is a first step towards a new PLP-based relational causal structure discovery algorithm. Moreover, considering the work of S. Holtzen et. al. [8], it also seems that such an analysis may contribute to speed up (lifted) inference in PLP. Let us now proceed to the formulation of the main problem in this paper.

2. Formulation of the Problem

Let us consider an open universe LPAD-program $\mathbf{P}$, which is given by the clauses $H_j : \alpha \leftarrow B_1^{(j)}, ..., B_{m_j}^{(j)}$ for $1 \leq j \leq n$, i.e. the H_j are atoms, $B_1^{(j)}, ..., B_{m_j}^{(j)}$ are literals and $\alpha \in [0, 1]$ is a real number, denoting the probability for the clause to hold. Further, assume that $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{C}_3$ are finite sets of literals, which take probabilities 0 or 1 in every realization of the program $\mathbf{P}$ and let us choose head atoms H_{j_1}, H_{j_2} and H_{j_3}. For a given realization Π of $\mathbf{P}$ we denote by $\{H_{j_i} : \mathbf{C}_i\}^{\Pi}$ the definable set of all ground atoms, i.e. random variables, which we obtain from H_{j_i} under the condition that the literals in $\mathbf{C}_i$ hold true. A more formal definition of the sets $\{H_{j_i} : \mathbf{C}_i\}^{\Pi}$ will be given in Section 4.1. This paper investigates the following question:

Problem 1. *Is $\{H_{j_1} : \boldsymbol{C}_1\}^{\Pi}$ independent of $\{H_{j_2} : \boldsymbol{C}_2\}^{\Pi}$ if we condition on $\{H_{j_3} : \boldsymbol{C}_3\}^{\Pi}$ for every realization Π of the program $\boldsymbol{P}$?*

Example 3. *Let us consider storages, which consist of two sections s_1 and s_2. Further, for each storage a database $belongs(R, S)$ tells us, which rooms R belong to which section S. Since s_1 and s_2 are the only sections and each room lies in one section, we obtain the following constraint:*

$$\forall_R \neg (belongs(R, s_1) \wedge belongs(R, s_2)) \tag{1}$$

Moreover, in each room R we find tanks T, denoted by $in(T, R)$ and each tank T stores a liquid L, denoted by $stores(T, L)$. Finally, we assume a database $inflammable(L)$, indicating the inflammable liquids.

An employee E opens a tank T with probability α, denoted by $opens(E, T)$, i.e. we obtain:

$$opens(E, T) \,:\, \alpha \leftarrow \tag{2}$$

Further, if employee E opens tank T, it may be with probability β that E doesn't close the tank T properly, which then causes the tank T to leak, denoted by $leaks(T)$:

$$leaks(T) \,:\, \beta \leftarrow opens(E, T) \tag{3}$$

Moreover, an employee E smokes in a room R, denoted by $smokes(E, R)$, with probability γ:

$$smokes(E, R) \,:\, \gamma \leftarrow \tag{4}$$

If employee E smokes in a room R, which contains a leaking tank storing an inflammable liquid, this causes a fire in the corresponding section with probability δ:

$$fire(S) \,:\, \delta \leftarrow smokes(E, R), leaks(T), in(T, R), belongs(R, S), stores(T, L), inflammable(L) \tag{5}$$

In this paper we present a reasoning, which verifies for instance that the event

$$\{smokes(e, r) : e \text{ employee, } r \text{ room, } belongs(r, s_1)\}$$

of an employee smoking in a room of section s_1 is independent of the event

$$\{opens(e, t) : e \text{ employee, } t \text{ tank, } r \text{ room, } in(t, r), belongs(r, s_1)\}$$

of an employee opening a tank in section s_1. Further, we discuss how things change once we observe a fire or a leaking tank in s_1.

3. A Formalism for our Solution

Recall that events of the trivial probabilities 0 and 1 are independent of every other event. To overcome this obstruction we introduce a language, which separates domain predicates, denoting logical statements with probabilities 0 and 1 from random predicates, denoting events with a probability, properly lying between 0 and 1.

3.1. Syntax

We consider a language in two parts: a **probabilistic vocabulary** $\mathfrak{P}$ and a **domain vocabulary** $\mathfrak{D} \subseteq \mathfrak{P}$. Here $\mathfrak{P}$ is a finite multisorted relational vocabulary, i.e. it consists of a finite set of sorts, a finite set of relation symbols and a finite set of constants in those sorts, as well as a countably infinite set of variables in every sort. For every variable or constant x we will denote it's sort by $\mathfrak{s}(x)$. Further, $\mathfrak{D}$ is a subvocabulary of $\mathfrak{P}$, which contains all of the variables and constants of $\mathfrak{P}$ as well as a (possibly empty) subset of the relation symbols of $\mathfrak{P}$.

Example 4. *In Example 3 the vocabulary $\mathfrak{D}$ consists of the predicates belongs(R, S), in(T, R), stores(T, L) and inflammable(L), which we assume to be given by databases.*

From now on we fix vocabularies $\mathfrak{D} \subseteq \mathfrak{P}$ as above. As usual, an **atom** is an expression of the form $r(t_1, \dots, t_n)$, where r is a relation symbol and t_1 to t_n are constants or variables of the correct sorts and a **literal** is an expression of the form A or $\neg A$ for an atom A. It is called a **domain atom** or **literal** if r is in $\mathfrak{D}$ and a **random atom** or **literal** if r is in $\mathfrak{P} \setminus \mathfrak{D}$. A literal of the form A is called **positive** and a literal of the form $\neg A$ is called **negative**.

Example 5. *In Example 4 belongs(R, s_1) is a domain atom, whereas smokes(E, R) is a random atom.*

Formulas, as well as **existential** and **universal formulas** are defined as usual in first-order logic. Moreover, we will use the operator var(_) to refer to the free variables in an expression.

The domain vocabulary will be used to formulate constraints and conditions for our PLP. The purpose of a PLP, however, is to define distributions for the random variables, determined by the language $\mathfrak{P}$. This is done by so called random clauses, i.e. LPAD-clauses with one head atom and only random literals in their body, which we additionally annotate by existential domain formulas to reflect the circumstances under which they are available.

Definition 2 (Random Clause). *A **random clause** RC is an expression of the form*

$$(R : \alpha \leftarrow R_1, ..., R_m) \Leftarrow C$$

which is given by

*i) a random atom $R :=$ effect(RC), called the **effect** of RC*
*ii) a possibly empty set of random literals causes$(RC) := \{R_1, ..., R_m\}$, called the **causes** of RC*
*iii) a **condition** condition$(RC) := C$ of RC, which is an existential domain formula C, such that all free variables of C also occur in at least one of $R, R_1, ..., R_m$*
*iv) a real number $p(RC) := \alpha \in [0, 1]$, called the **probability** of RC.*

Finally, we define the set var(RC) *of free variables **occurring** in RC to be* var(effect(RC)) $\cup \bigcup\{\text{var}(R) : R \in \text{causes}(RC)\}$.

Remark 3. *In i) and ii) of Definition 2 we use the terminology of cause and effect to reflect that a ground program with Problog semantics denotes a structural equation model in the sense of [9, §1.4].*

Example 6. *To reason about the independence statements implied by the program in Example 3, we rephrase the clause (5), i.e. we arrange all domain literals in the body on the right side of* $\Leftarrow$ *and quantify over all variables, not occurring in the remaining literals. Hence, we obtain the following random clause RC:*

$$(fire(S) : \gamma \leftarrow smokes(E,R), leaks(T)) \Leftarrow \exists_L\, in(T,R) \wedge belongs(R,S) \wedge stores(T,L) \wedge inflammable(L) \qquad (6)$$

In the future examples we will refer to the program $\boldsymbol{P}$ *consisting of the clauses (1), (2), (3), (4) and (6) with the convention that we denote a clause of the form*
(effect(_) : p(_) $\leftarrow$ causes(_)) $\Leftarrow$ *True by* effect(_) : p(_) $\leftarrow$ causes(_).

3.2. Semantics

After having established the necessary syntax we introduce the corresponding semantics. Let us begin with the domain expressions.

3.2.1. Semantics of Domain Expressions

The semantics of domain expressions are given in a straightforward way.

We just want to highlight the unique names assumption in our definition of a structure:

A $\mathfrak{D}$**-structure** Δ consists of a sort universe $\mathfrak{s}^\Delta$ for every sort $\mathfrak{s}$, an element of the corresponding sort universe for every constant in $\mathfrak{D}$, *such that two different constants are mapped to different elements*, and a relation among the corresponding sort universes for every relation in $\mathfrak{D}$.

Whether a domain formula is **satisfied** by a given $\mathfrak{D}$-structure (under a given **interpretation** of its free variables) is determined by the usual rules of first-order logic.

3.2.2. Semantics of Probabilistic Expressions

Let us proceed with the semantics for probabilistic expressions. As a $\mathfrak{P}$-structure should extend a $\mathfrak{D}$-structure by random variables we obtain:

Definition 3 (Ground Variable & $\mathfrak{P}$-Structure).

i) Let Δ *be a* $\mathfrak{D}$*-structure. A* ***ground variable*** $G := r(x_1, \dots, x_q)$ *consists of a random predicate* $\text{pred}(G) := r$ *of arity* $\text{ar}(r) := (\mathfrak{s}_1, ..., \mathfrak{s}_q)$*, called the* ***predicate*** *of* G *and a tuple of suitable realizations* $\text{arguments}(G) := (x_1, ..., x_q) \in \prod_{i=1}^{q} \mathfrak{s}_i^\Delta$*, called the* ***arguments*** *of* G*.*

ii) A $\mathfrak{P}$***-structure*** Π *consists of a* $\mathfrak{D}$*-structure* Π*, and a distinct, non-trivially distributed, Boolean random variable* $G^\Pi := r^\Pi(x_1, ..., x_q)$ *for every ground variable* $G = r(x_1, ..., x_q)$*.*

Remark 4. *Please note, that in ii) of Definition 3 we also require a version of the unique names assumption.*

Now we can make sense of the low level constructs in the language $\mathfrak{P}$.

Definition 4 (Semantics of Random Literals). *Let Π be a $\mathfrak{P}$-structure. We define for every random atom $A := r(x_1, ..., x_q)$ and for every variable interpretation ι on* $\mathrm{var}(A)$ *the following Boolean random variables:*

$$r(x_1, ..., x_q)^\iota := r^\Pi(x_1^\iota, ..., x_q^\iota)$$

$$(\neg r(x_1, ..., x_q))^\iota := \neg r^\Pi(x_1^\iota, ..., x_q^\iota).$$

Probabilistic Logic Programs

For the semantics of clauses and programs we choose Problog semantics, since a ground Problog program is essentially the same as a structural equation model in the sense of [9, §1.4]. We start with the definition of a program:

Definition 5 (Program). *A **program** $\boldsymbol{P} := (\boldsymbol{R}, \boldsymbol{D})$ is a pair, which consists of*

i) a finite set of universal domain formulas $\boldsymbol{D}$

ii) a finite set of random clauses $\boldsymbol{R}$ with the following property:

> *For every random clause $RC \in \boldsymbol{R}$, every random literal $C \in \mathrm{causes}(RC)$, every $\mathfrak{D}$-structure Δ and every interpretation ι such that Δ satisfies $\boldsymbol{D}$ and the condition of RC under ι, there exists a sequence of random clauses $RC_0, ..., RC_n \in \boldsymbol{R}$ such that the condition of RC_i is satisfied by Δ under ι for all $1 \leq i \leq n$, $\mathrm{head}(RC_{i+1}) \in \mathrm{causes}(RC_i)$, $\mathrm{head}(RC_0) = C$ and $\mathrm{causes}(RC_n) = \emptyset$.*

*In this case $\boldsymbol{R}(\boldsymbol{P}) := \boldsymbol{R}$ is called the **random part** and $\boldsymbol{D}(\boldsymbol{P}) := \boldsymbol{D}$ is called the **deterministic part** of the program $\boldsymbol{P}$. Moreover, a $\mathfrak{D}$-structure Δ is called a **domain** of $\boldsymbol{P}$ if and only if $\Delta \models \boldsymbol{D}(\boldsymbol{P})$, i.e. if Δ satisfies the deterministic part of $\boldsymbol{P}$.*

Remark 5. *Note that the property in Definition 5 ii) is needed to ensure that that we obtain a well-defined distribution in Definition 7.*

Example 7. *In Example 6 we obtain that the deterministic part $\boldsymbol{D}(\boldsymbol{P})$ of $\boldsymbol{P}$ consists of the constraint (1). The remaining clauses of $\boldsymbol{P}$ form the random part $\boldsymbol{R}(\boldsymbol{P})$ of $\boldsymbol{P}$.*

In the present paper we want to reason on a syntactic level about the independence statements, which are implied by a program $\mathbf{P}$. To obtain a reasonable criterion we proceed as in the propositional theory [10] and reduce ourselves to those independence statements, which follow from d-separation in the corresponding BNs. Now the ground graphs are the DAGs, in which we want to apply d-separation.

Definition 6 (Ground Graph & Acyclic Program). *Let $\boldsymbol{P}$ be a program. For every domain Δ of $\boldsymbol{P}$ we define the **ground graph** $\mathrm{Graph}_\Delta(\boldsymbol{P})$ to be the directed graph, obtained by the following procedure:*

> *For every random clause $RC \in \boldsymbol{R}(\boldsymbol{P})$, for every cause $R \in \mathrm{causes}(RC)$ and for every interpretation ι on $\mathrm{var}(RC)$, satisfying $\mathrm{condition}(RC)^\iota = True$, we draw an arrow $\mathrm{effect}(RC)^\iota \leftarrow R^\iota$. In this case we say that the edge $\mathrm{effect}(RC)^\iota \leftarrow R^\iota$ is **induced** by the clause RC.*

*The program $\boldsymbol{P}$ is called **acyclic** if the ground graph $\mathrm{Graph}_\Delta(\boldsymbol{P})$ is acyclic for all domains Δ of $\boldsymbol{P}$.*

Example 8. *It is easy to see that the program* $\boldsymbol{P}$ *of Example 6 is an acyclic program, since there are only arrows from the variables leaks(_) and smokes(_, _) to fire(_) and from the variables opens(_, _) to leaks(_).*

Finally, we are in the position to give the semantics of an acyclic program.

Definition 7 (Semantics of Acyclic Programs). *Let* $\boldsymbol{P}$ *be an acyclic program. Assume the random part* $\boldsymbol{R}(\boldsymbol{P})$ *of* $\boldsymbol{P}$ *consists of the random clauses* RC_i*, which are given by*

$$\left(R_i \,:\, \alpha_i \leftarrow R_1^{(i)}, ..., R_{m_i}^{(i)}\right) \Leftarrow C_i$$

for $1 \leq i \leq m$*. Further, let* Δ *be a domain of* $\boldsymbol{P}$*. We define the* ***grounding*** $\boldsymbol{P}^{\Delta}$ *of the program* $\boldsymbol{P}$ *w.r.t.* Δ *to be the Boolean functional causal model [9, §1.4] on the set of random variables*

$$\mathscr{R}(\Delta) := \{\text{effect}(RC_i)^{\iota} : 1 \leq i \leq m, \iota \text{ interpretation on } \text{var}(RC_i) \text{ such that } \text{condition}(RC_i)^{\iota} = True\},$$

which is given by the following equation for every $R^{\iota} \in \mathscr{R}(\Delta)$*:*

$$R^{\iota} := \bigvee_{\substack{RC_i \in \boldsymbol{R}(\boldsymbol{P}) \\ \kappa \text{ interpretation on } \text{var}(RC_i) \\ \text{effect}(RC_i)^{\kappa} = R^{\iota} \\ \text{conditions}(RC_i)^{\kappa} = True}} \bigwedge_{j=1}^{m_i} \left(R_j^{(i)}\right)^{\kappa} \wedge u(RC_i, \kappa),$$

where we have that $u(RC_i, \kappa)$ *is a Boolean random variable with the distribution*

$$\mathbb{P}\left(u(RC_i, \kappa) = True\right) := \alpha_i = p(RC_i)$$

for every interpretation κ *on* $\text{var}(RC_i)$*. Besides, the random variables* $u(RC_i, \kappa)$ *are assumed to be mutually independent for every random clause* RC_i*, every interpretation* κ *on* $\text{var}(RC_i)$ *and every* $1 \leq i \leq m$*.*

Notation 1. *We say that a* $\mathfrak{P}$*-structure* Π *is* ***generated*** *by a program* $\boldsymbol{P}$ *and write* $\Pi \models \boldsymbol{P}$ *if the following assertions hold:*

i) the underlying $\mathfrak{D}$*-structure* Δ *is a domain of* $\boldsymbol{P}$
ii) the joint distribution on the random variables $\text{effect}(RC)^{\iota}$ *for* $RC \in \boldsymbol{R}(\boldsymbol{P})$*,* ι *interpretation with* $\text{condition}(RC)^{\iota} = True$ *coincides with the one induced by the functional causal model* $\boldsymbol{P}^{\Delta}$*.*

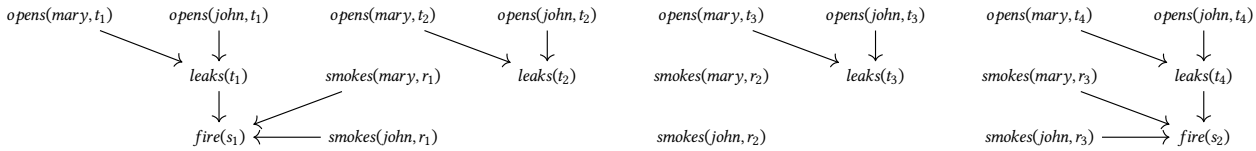
In this case Π *is called a* ***(probabilistic) realization*** *of* $\boldsymbol{P}$*.*

Remark 6. *Note that the semantics of Definition 7 do not coincide with the classical LPAD semantics. However, our semantics still cause the same BN structures after grounding. Further, note that the functional causal model* $\boldsymbol{P}^{\Delta}$ *has the causal diagram* $\text{Graph}_{\Delta}(\boldsymbol{P})$ *for every domain* Δ *of the program* $\boldsymbol{P}$*, i.e. by [9, §1.4] we obtain that* $\boldsymbol{P}^{\Delta}$ *induces a distribution, which can be stored in a BN with the ground graph* $\text{Graph}_{\Delta}(\boldsymbol{P})$ *as underlying DAG.*

Example 9. *It is easy to observe that the program* $\boldsymbol{P}$ *of Example 6 is indeed an acyclic program in our sense. Now assume we are given a specific storage, which consists of three rooms* r_1, r_2 *and* r_3*. These rooms are assigned to sections as follows:* $belongs(r_1, s_1), belongs(r_2, s_1)$ *and* $belongs(r_3, s_2)$*. Moreover, we have* 4 *tanks* t_1, t_2, t_3 *and* t_4 *with* $in(t_1, r_1), in(t_2, r_1), in(t_3, r_2)$ *and* $in(t_4, r_3)$*. The tanks contain three types of liquids* l_1, l_2 *and* l_3*, which we describe in the following way:* $stores(l_1, t_1), stores(l_2, t_2), stores(l_2, t_3)$ *and* $stores(l_3, t_4)$ *with* $inflammable(l_1)$ *and* $inflammable(l_3)$*. Finally, assume there are two employees Mary and John, which we express simply as* $mary$ *and* $john$*. In this case one checks that the storage above satisfies the deterministic part (1), i.e. we are given a domain* Δ *of the program* $\boldsymbol{P}$*. Further, for* $e \in \{john, mary\}$*,* $t \in \{t_1, t_2, t_3, t_4\}$ *and* $r \in \{r_1, r_2, r_3\}$ *the grounding* $\boldsymbol{P}^{\Delta}$ *is given by equations of the form:*

- $opens(e, t) := u(opens(E, T) : \alpha \leftarrow, \{E \mapsto e, T \mapsto t\})$ *such that* $u(opens(E, T), \{E \mapsto e, T \mapsto t\})$ *is true with probability* α
- $leaks(t) := opens(e, t) \wedge u(leaks(T) : \beta \leftarrow opens(E, T), \{E \mapsto e, T \mapsto t\})$ *such that* $u(leaks(T) : \beta \leftarrow opens(E, T), \{E \mapsto e, T \mapsto t\})$ *is true with probability* β
- $smokes(e, r) := u(smokes(E, R) : \gamma \leftarrow, \{E \mapsto e, R \mapsto r\})$ *such that* $u(smokes(E, R) : \gamma \leftarrow, \{E \mapsto e, R \mapsto r\})$ *is true with probability* γ
- $fire(s_1) := smokes(e, r_1) \wedge leaks(t_1) \wedge u(RC, \{S \mapsto s_1, E \mapsto e, R \mapsto r_1, T \mapsto t_1\})$ *such that* $u(RC, \{S \mapsto s_1, E \mapsto e, R \mapsto r_1, T \mapsto t_1\})$ *is true with probability* δ
- $fire(s_2) := smokes(e, r_3) \wedge leaks(t_4) \wedge u(RC, \{S \mapsto s_2, E \mapsto e, R \mapsto r_3, T \mapsto t_4\})$ *such that* $u(RC, \{S \mapsto s_2, E \mapsto e, R \mapsto r_3, T \mapsto t_4\})$ *is true with probability* δ

In the situation above we obtain the following ground graph $\mathrm{Graph}_{\Delta}(\boldsymbol{P})$*:*



Further, it is is easy to check that this is the causal diagram, i.e. the corresponding BN structure, of the functional causal model $\boldsymbol{P}^{\Delta}$*.*

As we now defined the necessary refinement of the LPAD-language, we can return to Problem 1.

4. Reasoning about Probabilistic Independence in Programs

Having the language of Section 3 to our disposal, we now reformulate Problem 1, i.e. we formally define the sets $\{H_{j_i} : \mathbf{C}_i\}^{\Pi}$.

4.1. Independence of Context Variables

We introduce context variables to reason about definable sets of random variables in an open universe.

Definition 8 (Context Variables). *A* ***context variable*** $C := \{\text{atom}(C) : \text{context}(C)\}$ *is a pair, which consists of a random atom* $\text{atom}(C)$*, called the* ***atom*** *of* C*, and an existential formula* $\text{context}(C)$ *with* $\text{var}(\text{context}(C)) \subseteq \text{var}(\text{atom}(C))$*, called the* ***context*** *of* C*.*

Moreover, we associate to every $\mathfrak{P}$*-structure* Π *the following set of random variables:*

$$C^{\Pi} = \{\text{atom}(C) : \text{context}(C)\}^{\Pi} := \{\text{atom}(C)^{\iota} : \iota \text{ interpretation on } \text{var}(C) \text{ with } \text{context}(C)^{\iota} = True\}$$

Finally, we define for every set $\boldsymbol{C}$ *of context variables* $\boldsymbol{C}^{\Pi} := \bigcup_{C \in \boldsymbol{C}} C^{\Pi}$.

Example 10. *Let us consider the situation in Example 6. We obtain that the pair*

$$\{leaks(T) : \exists_R \exists_L in(T, R) \wedge belongs(R, s_1) \wedge stores(T, L) \wedge inflammable(L)\}$$

is a context variable, denoting the event that in section s_1 *there is a leaking tank, which stores an inflammable liquid. Moreover, for the realization* Π *of the program* $\boldsymbol{P}$ *in Example 9 we obtain that*

$$\{leaks(T) : \exists_R \exists_L in(T, R) \wedge belongs(R, s_1) \wedge stores(T, L) \wedge inflammable(L)\}^{\Pi} = \{leaks(t_1)\}.$$

We are interested in the independence relations, which hold across all realizations of an acyclic program $\mathbf{P}$. Hence, let us fix an acyclic program $\mathbf{P}$. Altogether we are interested in the following notion:

Definition 9 (Independence of Context Variables). *We say that two sets of context variables* $\boldsymbol{X}$ *and* $\boldsymbol{Y}$ *are* ***independent, conditioned*** *on a third set of context variables* $\boldsymbol{Z}$ *if for every realization* $\Pi \models \boldsymbol{P}$ *we have that* $\boldsymbol{X}^{\Pi}$ *is independent of* $\boldsymbol{Y}^{\Pi}$*, whenever we condition on* $\boldsymbol{Z}^{\Pi}$*. In this case we write* $X \sqcup Y | Z$*.*

To transfer d-separation from the ground graphs $\text{Graph}_{\Delta}(\mathbf{P})$ to an open universe we need a representation for all d-connecting paths that might occur in a specific ground graph $\text{Graph}_{\Delta}(\mathbf{P})$ for a domain Δ of $\mathbf{P}$. Before we introduce this representation in Section 4.3, we still need to lift the notion of an interpretation to the open universe setting.

4.2. Substitutions

A substitution is the open universe analogue of an interpretation.

Definition 10 (Substitution). *A* ***substitution*** θ *is a function that associates to each* S *from a finite set of variables* $\boldsymbol{Dom}(\theta)$ *a variable or constant* S^{θ} *of the same sort.*

If $\boldsymbol{V} \subseteq \boldsymbol{Dom}(\theta)$*, we call* θ *a* ***substitution on*** $\boldsymbol{V}$*. We extend* θ *to a function on atoms, clauses and formulas. To avoid variables being substituted into or out of the scope of a quantifier, first rename all bound variable occurrences in* φ *using variable names outside the domain and the range of* θ*. Then obtain* φ^{θ} *by applying* θ *to every remaining occurrence of the variables of* $\boldsymbol{Dom}(\theta)$ *in* φ*.*

An expression B *is called a* ***specialization*** *of an expression* A *if there exists a substitution* θ *such that* $A^{\theta} = B$*.*

Example 11. *Let us reconsider Example 6. Further, let E and E' be variables of the sort employee, let S be a variable of the sort section and let R be a variable of the sort room. Then $E^\theta := E'$, $S^\theta := s_1$ and $R^\theta := R$ defines a substitution on the set $\{E, S, R\}$ and we obtain:*

$$smokes(E, R)^\theta = smokes(E', R),\ fire(S)^\theta = fire(s_1),\ fire(s_2)^\theta = fire(s_2)$$

As a preparation for the definition of the big dependence graph in Section 4.3 we need the following lemma about the existence of substitutions. Since it is an immediate consequence of the unique names assumption, its proof is omitted here.

Lemma 1. *Let A and B be random atoms such that $A^\iota = B^{\iota'}$ for interpretations ι and ι' of a $\mathfrak{P}$-structure Π. Then there exists substitutions θ and θ' such that $A^\theta = B^{\theta'}$, $\left(A^\theta\right)^\iota = A^\iota$ and such that $\left(B^{\theta'}\right)^\iota = B^{\iota'}$.*

When we come to decidability in Section 4.4, we consider the following special substitutions:

Definition 11. *A substitution θ on $\boldsymbol{V}$ is called a **relabelling** if it is injective and for all variables $S \in \boldsymbol{Dom}(\theta)$ we have that S^θ is again a variable. We say that two atoms A and B are **unifiable** and write $A \sim B$ if there exists a relabelling θ such that $A^\theta = B$.*

Notation 2. *Note that unifiability $\sim$ is an equivalence relation. We denote by $[R]$ the equivalence class of a random atom R upto unifiablity. Further, we call $[R]$ the **unification class** of R.*

Example 12. *Note that θ in Example 11 is not a relabelling, whereas $E^\rho := E'$, $R^\rho := R$ and $S^\rho := S$ would be a relabelling on $\{E, R, S\}$.*

As it turns out the decidability of our independence analysis relies on the following lemma:

Lemma 2. *There are finitely many unification classes of random atoms in $\mathfrak{P}$. Moreover, determining whether two random atoms are unifiable is decidable.*

Proof. Note that for every two random atoms we have that $r_1(x_1, ..., x_n) \sim r_2(y_1, ..., y_m)$ if and only if the following assertions hold:

i) $r_1 = r_2$, i.e. $n = m$ and $\mathfrak{s}(x_i) = \mathfrak{s}(y_i)$ for all $1 \leq i \leq n$.
ii) For all $1 \leq i \leq n$ we have that x_i is a variable if and only if y_i is.
iii) For all $1 \leq i \leq n$ we have that $x_i = y_i$ if x_i or y_i is a constant.
iv) For all $1 \leq i < j \leq n$ we have that $x_i \neq x_j$ if and only if $y_i \neq y_j$.

As there are only finitely many random predicates of finite arity and finitely many constants in $\mathfrak{P}$, we obtain the desired result. □

Now let us move on to the big dependence graph, which turns out to be the right object for reasoning about d-separation in our setting.

4.3. The Big Dependence Graph

As already mentioned, in order to reason about d-separation in an open universe we need a structure, which allows for a representation of all d-connecting paths, that might occur in a ground graph $\mathrm{Graph}_\Delta(\mathbf{P})$ for a possible domain Δ of the program $\mathbf{P}$.

Definition 12 (Big Dependence Graph). *The **big dependence graph*** $\mathrm{Graph}(\boldsymbol{P})$ *is the directed labelled graph, obtained by the following procedure:*

> *For every random clause* $RC \in \boldsymbol{R}(\boldsymbol{P})$*, for each substitution* θ *on* $\mathrm{var}(RC)$ *and for every cause* $R \in \mathrm{causes}(RC)$ *we draw an edge* $R^\theta \to \mathrm{effect}(RC)^\theta$*. An edge of this kind is said to be **induced** by RC. Moreover, we label each edge* $R^\theta \to \mathrm{effect}(RC)^\theta$*, which is induced by the random clause* $RC \in \boldsymbol{R}(\boldsymbol{P})$*, with the existential formula*
>
> $$\omega\left(R^\theta \to \mathrm{effect}(RC)^\theta\right) := \left(\exists_{\boldsymbol{y}}\, \mathrm{condition}(RC)\right)^\theta.$$
>
> *Here* $\boldsymbol{y}$ *denotes the set of variables, which occur in RC but not in* $\mathrm{effect}(RC)$ *and R.*

Remark 7. *One should read* $\omega\left(R^\theta \to \mathrm{effect}(RC)^\theta\right)$ *as follows: "For every interpretation* ι *on* $\mathrm{var}(RC)$ *the edge* $R^\iota \to \mathrm{effect}(RC)^\iota$ *exists in the ground graph* $\mathrm{Graph}_\Delta(\boldsymbol{P})$ *if* $\omega\left(R^\theta \to \mathrm{effect}(RC)^\theta\right)^\iota$ *holds true."*

Example 13. *Let us reconsider the program* $\boldsymbol{P}$ *in Example 6. Moreover, let* $(S_i)_{i\in\mathbb{N}}$, $(R_i)_{i\in\mathbb{N}}$, $(T_i)_{i\in\mathbb{N}}$ *and* $(E_i)_{i\in\mathbb{N}}$ *be enumerations of the variables of sort section, room, tank, employee, respectively. In that case, the big dependence graph* $\mathrm{Graph}(\boldsymbol{P})$ *is the infinite graph with the nodes* $fire(S_i)$, $fire(s_m)$, $leaks(T_j)$, $smokes(E_k, R_l)$, $opens(E_k, T_j)$ *for* $i, j, k, l \in \mathbb{N}$, $m = 1, 2$ *and with the edges:*

$$opens(E_k, T_j) \xrightarrow{True} leaks(T_j)$$

$$smokes(E_k, R_l) \xrightarrow{\exists_T \exists_L in(T,R_l) \wedge belongs(R_l,s_m) \wedge stores(T,L) \wedge inflammable(L)} fire(s_m)$$

$$leaks(T_j) \xrightarrow{\exists_R \exists_L in(T_j,R) \wedge belongs(R,s_m) \wedge stores(T_j,L) \wedge inflammable(L)} fire(s_m)$$

$$smokes(E_k, R_l) \xrightarrow{\exists_T \exists_L in(T,R_l) \wedge belongs(R_l,S_i) \wedge stores(T,L) \wedge inflammable(L)} fire(S_i)$$

$$leaks(T_j) \xrightarrow{\exists_R \exists_L in(T_j,R) \wedge belongs(R,S_i) \wedge stores(T_j,L) \wedge inflammable(L)} fire(S_i)$$

d-Separation for PLP

The next step will be to establish an analogue of d-separation on the big dependence graph Graph(**P**). This criterion will consist of two components: A d-connecting path in Graph(**P**) is a graph-theoretical object, which indicates possible d-connecting paths that might occur in a ground graph $\mathrm{Graph}_\Delta(\mathbf{P})$ for a domain Δ of the program **P**. Further, we assign to each d-connecting path in the big dependence graph Graph(**P**) a set of conditions, which tells us under which assumptions on the domain Δ we expect this path to appear in the corresponding ground graph $\mathrm{Graph}_\Delta(\mathbf{P})$.

Definition 13 (d-Connecting Path). *Let A and B be random atoms. Further, let $\boldsymbol{Z}$ be a set of context variables. A d-connecting path between A and B with respect to $\boldsymbol{Z}$ is a finite undirected path of the form $\mathscr{P} = A^{\theta} - ... - B^{\theta'}$ for substitutions θ and θ' on $\mathrm{var}(A)$, respectively $\mathrm{var}(B)$ with the following property:*

For every collider $... \to C \leftarrow ...$ in $\mathscr{P}$ there exists a directed path of the form

$$\mathscr{P}(C) := C \to ... \to \mathrm{atom}(Z)^{\theta(C)}$$

for a context variable $Z \in \boldsymbol{Z}$ and a substitution $\theta(C)$ on $\mathrm{var}(\mathrm{atom}(Z))$.

Example 14. *In Example 13 we have that $smokes(E_j, R_k) \to fire(s_1) \leftarrow leaks(T_i) \leftarrow opens(E_{j'}, T_i)$ yields a d-connecting path $\mathscr{P}$ between $smokes(E_1, R_1)$ and $opens(E_2, T_1)$ with respect to $\boldsymbol{Z} := \{\{fire(s_1) : True\}, \{leaks(T_1) : \exists_R in(T_1, R) \wedge belongs(R, s_2)\}\}$.*

We still need to find out under which conditions we expect a given d-connecting path in the big dependence graph $\mathrm{Graph}(\mathbf{P})$ to occur in a ground graph $\mathrm{Graph}_{\Delta}(\mathbf{P})$. This is done by the following conditions function, which we define by recursion on the structure of a d-connecting path:

Definition 14 (Conditions Function). *As before we assume that A and B are random atoms. Further, we choose a set of context variables $\boldsymbol{Z}$.*

i) If $\mathscr{P} = A^{\theta}$ is a d-connecting path of length 1 in $\mathrm{Graph}(\boldsymbol{P})$, we set $\mathrm{conditions}(\mathscr{P}) := \emptyset$.

ii) If $\mathscr{P} = A^{\theta} - B^{\theta'}$ is a d-connecting path of length 2 in $\mathrm{Graph}(\boldsymbol{P})$, we set

$$\mathrm{conditions}(\mathscr{P}) := \left\{\omega\left(A^{\theta} - B^{\theta'}\right)\right\}.$$

iii) Assume we have two d-connecting paths $\mathscr{P}' := A^{\theta} - ... - D - C$ and $\mathscr{P}'' = B^{\theta'} - ... - E - C$ such that $D \leftarrow C$ or $E \leftarrow C$. Then by Definition 13 we obtain a d-connecting path $\mathscr{P} := A^{\theta} - ... - D - C - E - ... - B^{\theta'}$, for which we define $\mathrm{conditions}(\mathscr{P})$ to be the following set:

$$\mathrm{conditions}(\mathscr{P}') \cup \mathrm{conditions}(\mathscr{P}'') \cup \{\neg\, \mathrm{context}(Z)^{\sigma} : Z \in \boldsymbol{Z}, \sigma \text{ substitution}, \ \mathrm{atom}(Z)^{\sigma} = C\}$$

iv) Finally, assume that the following two d-connecting paths, $\mathscr{P}' := A^{\theta} - ... - D \to C$ and $\mathscr{P}'' = B^{\theta'} - ... - E \to C$, yield a d-connecting path $\mathscr{P} := A^{\theta} - ... - D \to C \leftarrow E - ... - B^{\theta'}$. Then by Definition 13 there exists a directed path $\mathscr{P}(C) := C \to ... \to \mathrm{atom}(Z)^{\theta(C)}$ for a context variable $Z \in \boldsymbol{Z}$ and a substitution $\theta(C)$. In this case we define $\mathrm{conditions}(\mathscr{P})$ to be the following set:

$$\mathrm{conditions}(\mathscr{P}') \cup \mathrm{conditions}(\mathscr{P}'') \cup \{\omega(E) : E \text{ edge of } \mathscr{P}(C)\} \cup \left\{\mathrm{context}\,(Z)^{\theta(C)}\right\}$$

Example 15. *In the case of the d-connecting path $\mathscr{P}$ in Example 14 we obtain for $\mathrm{conditions}(\mathscr{P})$ the following set:*

$$\{\exists_T \exists_L in(T, R_k) \wedge belongs(R_k, s_1) \wedge stores(T, L) \wedge inflammable(L),$$

$$\exists_R \exists_L in(T_i, R) \wedge belongs(R, s_1) \wedge stores(T_i, L) \wedge inflammable(L), \neg \exists_R in(T_i, R) \wedge belongs(R, s_2), True\}$$

Now we bring the graphical and the logical component together and define the correct notion of a d-connecting path between context variables in the big dependence graph $\mathrm{Graph}(\mathbf{P})$.

Definition 15 (d-Connecting Path between Context Variables). *Let X and Y be context variables. Further, let $\boldsymbol{Z}$ be a set of context variables. A **d-connecting path** $\mathscr{P}$ between X and Y with respect to $\boldsymbol{Z}$ is a d-connecting path $\mathscr{P}$ between* $\mathrm{atom}(X)$ *and* $\mathrm{atom}(Y)$*. We call $\mathscr{P}$ a **valid** d-connecting path if the set*

$$\mathrm{conditions}(\mathscr{P}) \cup \mathrm{context}(X) \cup \mathrm{context}(Y) \cup \boldsymbol{D}(\boldsymbol{P})$$

*is satisfiable. We say that $\boldsymbol{Z}$ **d-connects** X and Y in* $\mathrm{Graph}(\boldsymbol{P})$ *if there exists a valid d-connecting path between X and Y with respect to $\boldsymbol{Z}$ in* $\mathrm{Graph}(\boldsymbol{P})$*, otherwise we say that $\boldsymbol{Z}$ **d-separates** X and Y. Finally, we say that $\boldsymbol{Z}$ **d-connects** two sets of context variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ if there exist a $X \in \boldsymbol{X}$ and a $Y \in \boldsymbol{Y}$ such that $\boldsymbol{Z}$ d-connects X and Y. Otherwise, we say that $\boldsymbol{Z}$ **d-separates** $\boldsymbol{X}$ and $\boldsymbol{Y}$.*

Example 16. *Let $X := \{opens(E,T) : \exists_R in(T,R) \wedge belongs(R,s_1)\}$, let $Y := \{smokes(E,R) : belongs(R,s_1)\}$ and let $Y' := \{smokes(E,R) : belongs(R,s_2)\}$. Obviously, the storage in Example 9 satisfies* $\mathrm{conditions}(\mathscr{P}) \cup \{belongs(R_k,s_1), \exists_R in(T_i,R) \wedge belongs(R,s_1)\} \cup \boldsymbol{D}(\boldsymbol{P})$ *for $\mathscr{P}$ as in Example 14, i.e. $\mathscr{P}$ is a valid d-connecting path between X and Y with respect to $\boldsymbol{Z}$.*

Further, we also see that $\mathrm{conditions}(\mathscr{P}) \cup \{belongs(R_k,s_2), \exists_R in(T_i,R) \wedge belongs(R,s_1)\} \cup \boldsymbol{D}(\boldsymbol{P})$ *yields a contradiction as by (1) each room is allowed to lie in at most one section, i.e. $\mathscr{P}$ is not a valid d-connecting path between X and Y' with respect to $\boldsymbol{Z}$.*

With these definitions at our hand we can prove the following results:

Lemma 3. *Let X and Y be context variables. Further, let $\boldsymbol{Z}$ be a set of context variables. Assume there exists a domain Δ of $\boldsymbol{P}$ such that $\boldsymbol{Z}^\Delta$ d-connects X^Δ and Y^Δ in the ground graph* $\mathrm{Graph}_\Delta(\boldsymbol{P})$*. Then we obtain that $\boldsymbol{Z}$ d-connects X and Y in the big dependence graph* $\mathrm{Graph}(\boldsymbol{P})$*.*

Proof. This can be proven with the help of Lemma 1 by an induction on the structure of a d-connecting path. □

Lemma 4. *Let X and Y be context variables and let $\boldsymbol{Z}$ be a set of context variables. Further, assume that $\boldsymbol{Z}$ d-connects X and Y in the big dependence* $\mathrm{Graph}(\boldsymbol{P})$*. Then there exists a domain Δ of $\boldsymbol{P}$ such that $\boldsymbol{Z}^\Delta$ d-connects X^Δ and Y^Δ in the ground graph* $\mathrm{Graph}_\Delta(\boldsymbol{P})$*.*

Proof. For every valid d-connecting path $\mathscr{P} = \mathrm{atom}(X)^\theta - \ldots - \mathrm{atom}(Y)^{\theta'}$ we obtain that

$$\mathrm{context}(X)^\theta \cup \mathrm{context}(Y)^{\theta'} \cup \mathrm{conditions}(\mathscr{P}) \cup \mathbf{D}(\mathbf{P})$$

is satisfiable. Now take Δ to be a Herbrand model [11, §11.3] of the skolemization of the existential formulas in the upper set with the corresponding interpretation ι and we obtain by construction that

$$\mathrm{atom}(X)^{\iota\circ\theta} - \ldots - \mathrm{atom}(Y)^{\iota\circ\theta'}$$

is a d-connecting path between $\mathrm{atom}(X)^{\iota\circ\theta} \in X^\Delta$ and $\mathrm{atom}(Y)^{\iota\circ\theta'} \in Y^\Delta$. □

From these two lemmas we can easily deduce the next theorem, which yields the desired generalization of d-separation to open universe PLPs.

Theorem 2. *Let $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ be sets of context variables. Then we obtain that $\boldsymbol{Z}$ d-separates $\boldsymbol{X}$ and $\boldsymbol{Y}$ in the big dependence graph* Graph($\boldsymbol{P}$) *if and only if $\boldsymbol{Z}^{\Delta}$ d-separates $\boldsymbol{X}^{\Delta}$ and $\boldsymbol{Y}^{\Delta}$ in the ground graph* $\text{Graph}_{\Delta}(\boldsymbol{P})$ *for every domain Δ of $\boldsymbol{P}$.*

If we combine Theorem 2 with Theorem 1, we get:

Corollary 1. *Let $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ be sets of context variables. If $\boldsymbol{Z}$ d-separates $\boldsymbol{X}$ and $\boldsymbol{Y}$ in the big dependence graph* Graph($\boldsymbol{P}$)*, we obtain that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent conditioned on $\boldsymbol{Z}$, i.e. $\boldsymbol{X} \sqcup \boldsymbol{Y} | \boldsymbol{Z}$.*

Our final goal is to show that the problem of determining whether two context variables are d-separated with respect to a given set of context variables is decidable.

4.4. A Normal Form for d-Connecting Paths

Here we want to investigate the following setting: *Let us fix two context variables X and Y together with a set of context variables $\boldsymbol{Z}$.*

A priori searching for a d-connecting path between X and Y in the big dependence graph Graph($\mathbf{P}$) also means searching for an arbitrary long path in an infinite graph, which would render this problem undecidable. This is the reason why we establish a normal form for d-connecting paths and reduce the problem of finding a d-connecting path to finding one in normal form.

Definition 16 (Normal form of a d-connecting path). *A d-connecting path*

$$\mathscr{P} := \text{atom}(X)^{\theta} = R_1 - R_2 - \ldots - R_n = \text{atom}(Y)^{\theta'}$$

between X and Y in the big dependence graph Graph($\boldsymbol{P}$) *is said to be in **normal form** if for all $1 \leq i < j \leq n$ we have that R_i and R_j are not unifiable, i.e. $[R_i] \neq [R_j]$.*

Example 17. *Consider the following valid d-connecting path between $\{smokes(E, R) : True\}$ and $\{opens(E, T) : True\}$ with respect to $\{fire(S) : True\}$ in the big dependence graph of Example 13:*

$$smokes(E, R) \rightarrow fire(S) \leftarrow leaks(T') \rightarrow fire(S') \leftarrow leaks(T) \leftarrow opens(E, T)$$

Note that this path is not in normal form as $fire(S)$ and $fire(S')$ are unifiable. However, in order to see that $\{fire(S) : True\}$ d-connects $\{smokes(E, R) : True\}$ and $\{opens(E, T) : True\}$, we can also take the valid d-connecting path

$$smokes(E, R) \rightarrow fire(S) \leftarrow leaks(T) \leftarrow opens(E, T),$$

which is in normal form. In Lemma 5 we generalize this observation to a calculus, which transforms each valid d-connecting path to one in normal form.

Finally, we reduce the problem of finding a d-connecting path between X and Y with respect to $\mathbf{Z}$ to the problem of finding a d-connecting path between them in normal form.

Lemma 5. *If $\boldsymbol{Z}$ d-connects X and Y, then there exists a valid d-connecting path $\mathscr{P}$ in normal form between X and Y.*

Proof. Assume $\mathscr{P} := \mathrm{atom}(X)^{\theta} - ... - R_i - ... - R_j - ... - \mathrm{atom}(Y)^{\theta'}$ is a valid d-connecting path, which is not in normal form, i.e. we may assume that R_i and R_j are unifiable for $i < j$. Hence, there exists a relabelling ρ such that $R_j^{\rho} = R_i$. By an induction on the structure of a d-connecting path one verifies that $\tilde{\mathscr{P}} := \mathrm{atom}(X)^{\theta} - ... - R_i = R_j^{\rho} - ... - \mathrm{atom}(Y)^{\rho \circ \theta'}$ is also a d-connecting path between X and Y. Moreover, observe that $\tilde{\mathscr{P}}$ is valid as the conditions function in Definition 14 is monotone in the length of a d-connecting path. Now extensionally applying the procedure above yields the desired result. □

From Lemma 5 we obtain the desired decidability of d-separation for open universe PLPs:

Theorem 3. *Let X, Y be context variables and let $\boldsymbol{Z}$ be a set of context variables. Determining whether $\boldsymbol{Z}$ d-connects X and Y is decidable.*

Proof. Note that we can apply a suitable relabelling to a d-connecting path. Hence, to determine whether X and Y are d-connected we can proceed by Lemma 5 as follows:

1.) List all unification classes $[R]$ of specializations of $\mathrm{atom}(X)$.
2.) List all d-connecting paths in normal form, which start with a R of step 1.).
3.) Delete all paths, which don't end with a specialization of $\mathrm{atom}(Y)$.
4.) Check these paths for satisfiability.

Obviously, 1.), 3.) and 4.) are decidable. To achieve 2.), we build undirected paths starting from a R of step 1.) by applying the random clauses as indicated in Definition 12. We stop when we need to run twice in the same unification class. Since by Lemma 2 there are only finitely many unification classes, this method terminates. Further, we proceed analogously to check whether these paths are d-connecting paths, i.e. to check whether the property in Definition 13 holds. □

5. Conclusion

At first we introduced a new semantics for open universe probabilistic logic programs. Subsequently, we used the correspondence between ground acyclic programs and Bayesian networks to apply the method of d-separation in our setting. In particular, we assigned to each open universe program an infinite directed labelled graph, the big dependence graph. Finally, we saw that in order to deduce independence statements it is sufficient to search through the big dependence graph for d-connecting paths in normal form and that searching for those paths is a decidable problem.

In the future it would be interesting to extend the notion of a program by incorporating Prolog programs. This would reflect logic programming inside probabilistic logic programming. Further, it would also be desirable to bring our semantics closer to the LPAD semantics. We believe that an independence analysis for LPAD-programs at compilation time can be used to build modified binary decision trees as in [8] in order to speed up (lifted) inference. Moreover, we plan to construct a causal structure discovery algorithm for open universe LPAD-programs on the basis of the big dependence graph.

References

[1] T. Verma, J. Pearl, Causal Networks: Semantics and Expressiveness, in: Uncertainty in Artificial Intelligence, volume 9 of *Machine Intelligence and Pattern Recognition*, North-Holland, 1990, pp. 69–76. doi:https://doi.org/10.1016/B978-0-444-88650-7.50011-1.

[2] 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.

[3] D. Geiger, T. Verma, J. Pearl, Identifying Independence in Bayesian Networks, Networks 20 (1990) 507–534. doi:https://doi.org/10.1002/net.3230200504. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230200504.

[4] P. Spirtes, C. Glymour, R. Scheines, Causation, Prediction, and Search, 2nd ed., MIT press, 2000. URL: https://www.researchgate.net/publication/242448131_Causation_Prediction_and_Search.

[5] T. Verma, J. Pearl, Equivalence and Synthesis of Causal Models, in: Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence, UAI '90, Elsevier Science Inc., USA, 1990, p. 255–270. doi:https://dl.acm.org/doi/10.5555/647233.719736.

[6] C. Meek, Causal inference and causal explanation with background knowledge, in: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, UAI'95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, p. 403–410. doi:https://dl.acm.org/doi/10.5555/2074158.2074204.

[7] M. Maier, Causal Discovery for Relational Domains: Representation, Reasoning, and Learning, Doctoral dissertations. 279., University of Massachusetts - Amherst, 2014. URL: https://scholarworks.umass.edu/dissertations_2/279/.

[8] S. Holtzen, G. Van den Broeck, T. Millstein, Scaling exact inference for discrete probabilistic programs, Proc. ACM Program. Lang. 4 (2020). doi:https://doi.org/10.1145/3428208.

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

[10] D. Geiger, J. Pearl, On the logic of causal models, in: Uncertainty in Artificial Intelligence, volume 9 of *Machine Intelligence and Pattern Recognition*, North-Holland, 1990, pp. 3–14. doi:https://doi.org/10.1016/B978-0-444-88650-7.50006-8.

[11] H. Ebbinghaus, J. Flum, W. Thomas, Einführung in die mathematische Logik, 6 ed., Springer Spektrum, Berlin, DE, 2018. URL: https://www.springer.com/de/book/9783662580288.