An Algebraic Notion of Conditional Independence, and its Application to Knowledge Representation (Preliminary Report)

Jesse Heyninck

Open Universiteit, the Netherlands
University of Cape Town, South-Africa

Abstract
Conditional independence is a crucial concept supporting adequate modelling and efficient reasoning in probabilistics. In knowledge representation, the idea of conditional independence has also been introduced for specific formalisms, such as propositional logic and belief revision. In this paper, the notion of conditional independence is studied in the algebraic framework of approximation fixpoint theory. This gives a language-independent account of conditional independence that can be straightforwardly applied to any logic with fixpoint semantics. It is shown how this notion allows to reduce global reasoning to parallel instances of local reasoning. Furthermore, relations to existing notions of conditional independence are discussed and the framework is applied to normal logic programming.

1. Introduction

Over the last decades, conditional independence was shown to be a crucial concept supporting adequate modelling and efficient reasoning in probabilistics [1]. It is the fundamental concept underlying network-based reasoning in probabilistics, which has been arguably one of the most important factors in the rise of contemporary artificial intelligence. Even though many reasoning tasks on the basis of probabilistic information have a high worst-case complexity due to their semantic nature, network-based models allow an efficient computation of many concrete instances of these reasoning tasks thanks to local reasoning techniques. Conditional independence has also been investigated for several approaches in knowledge representation, such as propositional logic [2, 3], belief revision [4, 5] and conditional logics [6]. For many other central formalisms in KR, such a study has not been undertaken.

Due to the wide variety of formalisms studied in knowledge representation, it is often beneficial yet challenging to study a concept in a language-independent manner. Indeed, such language-independent studies avoid having to define and investigate the same concept for different formalisms. In recent years, a promising framework for such language-independent investigations is the algebraic *approximation fixpoint theory* (AFT) [7], which conceives of KR-formalisms as operators over a lattice (such as the immediate consequence operator from logic programming). Approximation fixpoint theory can represent a wide variety of KR-formalisms (see [8] for an overview), and was shown to be a fruitful framework for language-independent studies of concepts such as splitting [9], groundedness [10], equivalence [11] and non-determinism [12].

In this paper, we give an algebraic, operator-based account of conditional independence. Such an algebraic account is applicable to any formalism that admits an operator-based characterization, such as the ones mentioned above as well as any future instantiations of AFT. A main results of the paper is the fact that conditional independence allows to split the search for fixpoints of an (approximation) operator over conditionally indepedent modules. As a proof-of-concept, the framework is applied to normal logic programs, and it is shown that there are strong connections with several existing works.

Outline of the Paper: The necessary preliminaries on logic programming (Section 2.1), lattices (Section 2.2 and approximation fixpoint theory (Section 2.3) are introduced in Section 2. The concept of conditional independence of sub-lattices w.r.t. an operator is introduced and studied in Section 3. This concept is applied to approximation operators in Section 4. The usefulness of this theory is shown in Section 5, where it is applied to the semantics of normal logic programs. Finally, related work is discussed in Section 6, after which the paper is concluded (Section 7).

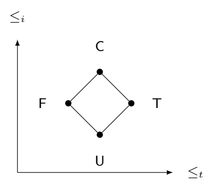
21st International Workshop on Nonmonotonic Reasoning, September 2–4, 2023, Rhodes, Greece
jesse.heyninck@ou.nl (J. Heyninck)
https://sites.google.com/view/jesseheyninck (J. Heyninck)
0000-0002-3825-4052 (J. Heyninck)

2. Background and Preliminaries

In this section, we recall the necessary basics of logic programming, abstract algebra and AFT.

2.1. Logic Programming

We assume a set of atoms $\mathcal{A}$ and a language $\mathcal{L}$ built up from atoms, conjunction $\wedge$ and negation $\neg$. A (propositional) logic program $\mathcal{P}$ (a dlp, for short) is a finite set of rules of the form $p \leftarrow \psi$, where p (the rule's head) is an atoms, and ψ (the rule's body) is a (propositional[1]) formula that may include the propositional constants T (representing truth), F (falsity), U (unknown), and C (contradictory information). A rule is called *normal* if its body is a conjunction of literals (i.e., atomic formulas or negated atoms). A program is *normal* if it consists only of normal rules; It is *positive* (or *definite*) if there are no negations in the rules' bodies. The set of atoms occurring in $\mathcal{P}$ is denoted $\mathcal{A}_\mathcal{P}$. We use the following four-valued bilattice:



We also assume a $\leq_t$-involution $-$ on $\leq_t$ (i.e, $-\mathsf{F} = \mathsf{T}$, $-\mathsf{T} = \mathsf{F}$, $-\mathsf{U} = \mathsf{U}$ and $-\mathsf{C} = \mathsf{C}$). A *four-valued interpretation* of a program $\mathcal{P}$ is a pair (x, y), where $x \subseteq \mathcal{A}_\mathcal{P}$ is the set of the atoms that are assigned a value in $\{\mathsf{T}, \mathsf{C}\}$ and $y \subseteq \mathcal{A}_\mathcal{P}$ is the set of atoms assigned a value in $\{\mathsf{T}, \mathsf{U}\}$.[2] Interpretations are compared by the *information order* $\leq_i$, where $(x, y) \leq_i (w, z)$ iff $x \subseteq w$ and $z \subseteq y$ (sometimes called "precision" order), and by the *truth order* $\leq_t$, where $(x, y) \leq_t (w, z)$ iff $x \subseteq w$ and $y \subseteq z$ (increased 'positive' evaluations). Truth assignments to complex formulas are then recursively defined as follows:

- $(x, y)(p) = \begin{cases} \mathsf{T} & \text{if } p \in x \text{ and } p \in y, \\ \mathsf{U} & \text{if } p \notin x \text{ and } p \in y, \\ \mathsf{F} & \text{if } p \notin x \text{ and } p \notin y, \\ \mathsf{C} & \text{if } p \in x \text{ and } p \notin y. \end{cases}$
- $(x, y)(\neg\phi) = -(x, y)(\phi)$,
- $(x, y)(\psi \wedge \phi) = lub_{\leq_t}\{(x, y)(\phi), (x, y)(\psi)\}$,

A four-valued interpretation of the form (x, x) may be associated with a *two-valued* (or *total*) interpretation x, in which for an atom p, $x(p) = \mathsf{T}$ if $p \in x$ and $x(p) = \mathsf{F}$ otherwise. We say that (x, y) is a *three-value* (or *consistent*) interpretation, if $x \subseteq y$. Note that in consistent interpretations there are no C-assignments.

We now consider semantics for lp's. First, given a two-valued interpretation, an extension to dlp's of the immediate consequence operator for normal programs [13] is defined as follows:

Definition 1. *Given a dlp $\mathcal{P}$ and a two-valued interpretation x, we define:*

$$IC_\mathcal{P}(x) = \{p \in \mathcal{A}_\mathcal{P} \mid p \leftarrow \psi \in \mathcal{P}, (x, x)(\psi) = \mathsf{T}\}.$$

For a four-valued interpretation (x, y), we define:

$$\mathcal{IC}^l_\mathcal{P}(x, y) = \{p \mid p \leftarrow \psi \in \mathcal{P}, (x, y)(\psi) \in \{\mathsf{T}, \mathsf{C}\}\}$$
$$\mathcal{IC}^u_\mathcal{P}(x, y) = \{p \mid p \leftarrow \psi \in \mathcal{P}, (x, y)(\psi) \in \{\mathsf{U}, \mathsf{T}\}\}$$
$$IC_\mathcal{P}(x, y) = (\mathcal{IC}^l_\mathcal{P}(x, y), \mathcal{IC}^u_\mathcal{P}(x, y))$$

Thus, denoting by $2^\mathcal{A}$ the powerset of $\mathcal{A}$, $IC_\mathcal{P}$ is an operator on the lattice $\langle 2^\mathcal{A}, \subseteq\rangle$ that derives all heads of rules with true bodies.

Another common way of providing semantics to dlp's is by the following reduct [14]:

Definition 2. *The GL-transformation $\frac{\mathcal{P}}{(x,y)}$ of an nlp $\mathcal{P}$ w.r.t. a consistent interpretation (x, y), is the positive program obtained by replacing, in every rule $p \leftarrow \bigwedge_{i=1}^m q_i \wedge \bigwedge_{j=1}^n \neg r_j \in \mathcal{P}$, any negated literal $\neg r_i$ $(1 \leq i \leq k)$ by: (1) F if $(x, y)(r_i) = \mathsf{T}$, (2) T if $(x, y)(r_i) = \mathsf{F}$, and (3) U if $(x, y)(r_i) = \mathsf{U}$. In other words, replacing $\neg r_i$ by $(x, y)(\neg r_i)$ An interpretation (x, y) is a* three-valued stable model *of $\mathcal{P}$ iff it is the $\leq_t$-minimal model of $\frac{\mathcal{P}}{(x,y)}$. If $x = y$, (x, y) is called a* two-valued *stable model of $\mathcal{P}$.*

The $\leq_i$-minimal (shown in [15] to be unique for normal logic programs) is called the *well-founded model.* We denote it by $\mathsf{WF}(\mathcal{P})$ and will, in the case $\mathsf{WF}(\mathcal{P})$ is total, abuse notation to denote $\mathsf{WF}(\mathcal{P}) = x$ if $\mathsf{WF}(\mathcal{P}) = (x, x)$.

2.2. Lattices and sub-lattices

We recall some necessary preliminaries on set theory and (sub-)lattices. A lattice is a partially ordered set $L = \langle \mathcal{L}, \leq\rangle$ where every two elements $x, y \in \mathcal{L}$ have a least upper $x \sqcup y$ and a greatest lower bound $x \sqcap y$. A lattice is complete if every set $X \subseteq \mathcal{L}$ has

[1] For simplicity and due to lack of space, we restrict ourselves to the propositional case.

[2] Somewhat skipping ahead, the intuition here is that x (y) is a lower (upper) approximation of the true atoms.

a least upper (denoted $\sqcup X$) and a greatest lower bound (denoted $\sqcap X$).

Let I be a set, which we call the *index set*, and for each $i \in I$, let S_i be a set. The product set $\otimes_{i\in I} S_i$ is the following set of functions:

$$\bigotimes_{i\in I} S_i = \{f \mid f : I \to \bigcup_{i\in I} S_i \text{ s.t. } \forall i \in I : \ f(i) \in S_i\}$$

Intuitively, the product set $\bigotimes_{i\in I} S_i$ contains all ways of selecting one element of every set S_i. For example, for the sets $S_1 = \{\emptyset, \{p\}\}$ and $S_2 = \{\emptyset, \{q\}\}$, $\bigotimes_{i\in\{1,2\}} S_i$ contains, among others, f and f' with $f(1) = f(2) = \emptyset$ and $f'(1) = \emptyset$ and $f'(2) = \{q\}$. For a finite set $I = \{1, \ldots, n\}$, the product $\otimes_{i\in I} S_i$ is (isomorphic to) the cartesian product $S_1 \times \ldots \times S_n$.

If each S_i is partially ordered by some $\leq_i$, this induces the product order $\leq_\otimes$ on $\otimes_{i\in I} S_i$: for all $x, y \in \otimes_{i\in I} S_i$, $x \leq_\otimes y$ iff for all $i \in I$, $x(i) \leq_i y(i)$. Where a distinction is required, we will denote the product order over S_i by $\otimes^I$. It can be easily shown that if all $\langle S_i, \leq_i\rangle$ are (complete) lattices, then $\langle \otimes_{i\in I} S_i, \leq_\otimes\rangle$ is also a (complete) lattice. We call this the *product lattice* of the lattices S_i.

We denote, for $x \in \bigotimes_{i\in I} S_i$ and $i \in I$, $x_{|i} \in S_i$ as $f(i)$, and for $J \subseteq I$ we denote $x_{|I}$ by $\bigotimes_{i\in J} x_i$. For example, using S_1 and S_2 as in the example above, $\emptyset \times \{q\}_{|1} = \emptyset$. Likewise, we denote by $x_i \otimes x_j$ the element $x \in S_i \otimes S_j$ s.t. $x_{|k} = x_k$ for $k = i, j$, and we lift this to sets as usual.

2.3. Approximation Fixpoint Theory

We now recall basic notions from approximation fixpoint theory (AFT), as described by Denecker, Marek and Truszczynski [16].

Given a lattice $L = \langle \mathcal{L}, \leq\rangle$, we let $L^2 = \langle \mathcal{L}^2, \leq_i, \leq_t\rangle$ be the structure (called *bilattice*), in which $\mathcal{L}^2 = \mathcal{L} \times \mathcal{L}$, and for every $x_1, y_1, x_2, y_2 \in \mathcal{L}$,

- $(x_1, y_1) \leq_i (x_2, y_2)$ if $x_1 \leq x_2$ and $y_1 \geq y_2$,
- $(x_1, y_1) \leq_t (x_2, y_2)$ if $x_1 \leq x_2$ and $y_1 \leq y_2$.

An *approximating operator* $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$ of an operator $O_\mathcal{L} : \mathcal{L} \to \mathcal{L}$ is an operator that maps every approximation (x, y) of an element z to an approximation (x', y') of another element $O(z)$, thus approximating the behavior of the approximated operator O.

Definition 3. *Let $O_\mathcal{L} : \mathcal{L} \to \mathcal{L}$ and $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$. (1) $\mathcal{O}$ is* $\leq_i$-monotonic, *if when $(x_1, y_1) \leq_i (x_2, y_2)$, also $\mathcal{O}(x_1, y_1) \leq_i \mathcal{O}(x_2, y_2)$; (2) $\mathcal{O}$ is* approximating, *if it is $\leq_i$-monotonic and for any $x \in \mathcal{L}$, $(\mathcal{O}(x, x))_1 = (\mathcal{O}(x, x))_2$;*[3] *(3) $\mathcal{O}$ is an* approximation *of $O_\mathcal{L}$, if it is $\leq_i$-monotonic and $\mathcal{O}$* extends *O, that is: $(\mathcal{O}(x, x))_1 = (\mathcal{O}(x, x))_2 = O_\mathcal{L}(x)$.*

To avoid clutter, we will also denote $(\mathcal{O}(x, y))_1$ by $\mathcal{O}_l(x, y)$ and $(\mathcal{O}(x, y))_2$ by $\mathcal{O}_u(x, y)$.

The *stable operator*, defined next, is used for expressing the semantics of many non-monotonic formalisms. Given a complete lattice $L = \langle \mathcal{L}, \leq\rangle$, let $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$ be an approximating operator. $\mathcal{O}_l(\cdot, y) = \lambda x.\mathcal{O}_l(x, y)$, i.e.: $\mathcal{O}_l(\cdot, y)(x) = \mathcal{O}_l(x, y)$ (and similarly for the upper bound operator $\mathcal{O}_u$). The *stable operator for $\mathcal{O}$* is: $S(\mathcal{O})(x, y) = (\mathit{lfp}(\mathcal{O}_l(., y)), \mathit{lfp}(\mathcal{O}_u(x, .))$. We also denote the components $\mathit{lfp}(\mathcal{O}_l(., y))$ and $\mathit{lfp}(\mathcal{O}_u(x, .)$ of the stable operator by $C(\mathcal{O}_l)(y)$ respectively $C(\mathcal{O}_u)(x)$.

Stable operators capture the idea of minimizing truth, since for any $\leq_i$-monotonic operator $\mathcal{O}$ on $\mathcal{L}^2$, fixpoints of the stable operator $S(\mathcal{O})$ are $\leq_t$-minimal fixpoints of $\mathcal{O}$ [16, Theorem 4]. Altogether, we obtain the following notions:

Given a complete lattice $L = \langle \mathcal{L}, \leq\rangle$, let $\mathcal{O} : \mathcal{L}^2 \to \mathcal{L}^2$ be an approximating operator. We call: (1) (x, y) a *Kripke-Kleene fixpoint* of $\mathcal{O}$ if $(x, y) = \mathrm{lfp}_{\leq_i}(\mathcal{O}(x, y))$; (2) (x, y) a *three-valued stable* fixpoint of $\mathcal{O}$ if $(x, y) = S(\mathcal{O})(x, y)$; (3) (x, x) a *two-valued stable fixpoints* of $\mathcal{O}$ if $(x, x) = S(\mathcal{O})(x, x)$; (4) (x, y) the *well-founded fixpoint* of $\mathcal{O}$ if it is the $\leq_i$-minimal (three-valued) stable model fixpoint of $\mathcal{O}$. It has been shown that every approximation operator admits a unique $\leq_i$-minimal stable fixpoint [16]. Pelov, Denecker and Bruynooghe [18] show that for normal logic programs, the fixpoints based on the four-valued immediate consequence operator $\mathcal{IC}_\mathcal{P}$ (recall Definition 1) for a logic program give rise to the following correspondences: the three-valued stable models coincides with the three-valued semantics as defined by Przymusinski [15], the well-founded model coincides with the homonymous semantics [15, 19], and the two-valued stable models coincide with the two-valued (or total) stable models of a logic program.

3. Conditional Independence

Conditional independence in an operator-based setting is meant to formalize the idea that for the application of an operator to a lattice consisting

[3]In some papers [16], an approximation operator is defined as a symmetric $\leq_i$-monotonic operator, i.e. a $\leq_i$-monotonic operator s.t. for every $x, y \in \mathcal{L}$, $\mathcal{O}(x, y) = (\mathcal{O}_l(x, y), \mathcal{O}_l(y, x))$ for some $\mathcal{O}_l : \mathcal{L}^2 \to \mathcal{L}$. However, the weaker condition we take here (taken from [17] is actually sufficient for most results on AFT.

of three sub-lattices L_1, L_2 and L_3, full information about $\mathcal{L}_3$ allows us to ignore $\mathcal{L}_2$ when applying O to $\mathcal{L}_1 \otimes \mathcal{L}_3$. In more detail, it means that $O : \bigotimes_{i\in\{1,2,3\}} S_i \to \bigotimes_{i\in\{1,2,3\}} S_i$ can be decomposed in two operators $O_{1,3} : S_1 \otimes S_3 \to S_1 \otimes S_3$ and $O_{2,3} : S_2 \otimes S_3 \to S_2 \otimes S_3$ s.t. for any $x = x_1 \otimes x_2 \otimes x_3$ $O(x) = O_{1,3}(x_1 \otimes x_3) \otimes O_{2,3}(x_2 \otimes x_3)_{|2}$.

Definition 4. *Let O be an operator on the product lattice $\otimes_{i\in\{1,2,3\}} S_i$. The lattices* L_1 and L_2 are independent w.r.t. L_3 according to O *(in symbols: $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$) if there exist operators $O_{1,3} : \otimes_{i\in\{1,3\}}\mathcal{L}_i \to \otimes_{i\in\{1,3\}}\mathcal{L}_i$ and $O_{2,3}\ \otimes_{i\in\{2,3\}}\mathcal{L}_i \to \otimes_{i\in\{2,3\}}\mathcal{L}_i$ s.t. for $i, j \in \{1, 2\}$, $i \neq j$, and for every $x_i \otimes x_3 \in \mathcal{L}_i \otimes \mathcal{L}_3$ and for every $x_j \in \mathcal{L}_j$ it holds that: $O(x_i \otimes x_j \otimes x_3)_{|i,3} = O_{i,3}(x_i \otimes x_3)$.*

Thus, two sub-lattices L_1 and L_2 are independent w.r.t. L_3 according to O if, once we have full information about L_3, information about L_2 does not contribute anything in the application of O when restricted to L_1 (and vice versa).

Example 1. *Consider the logic program $\mathcal{P}$ using atoms for* `inf`*ected,* `vac`*cinated and* `c`o`nt`a`ct`*:*

$$r_1 : \ \texttt{inf(b)} \leftarrow \texttt{inf(a)}, \texttt{cnct(a,b)}, \text{not}\, \texttt{vac(b)}.$$
$$r_2 : \ \texttt{inf(c)} \leftarrow \texttt{inf(a)}, \texttt{cnct(a,c)}, \text{not}\, \texttt{vac(c)}.$$
$$r_3 : \ \texttt{inf(a)}., \ r_4 : \ \texttt{cnct(a,b)}., \ r_5 : \ \texttt{cnct(a,c)}.$$

Notice that, as soon as we know that `infected(a)`*. is the case, we can decompose the search for models into two independent parts, as can also be seen in the dependency graph in figure 1.*

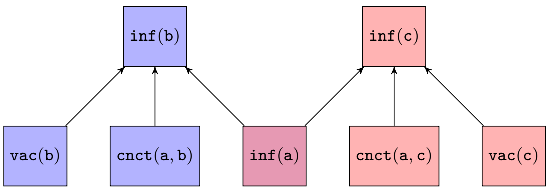


Figure 1: A dependency graph for the program $\mathcal{P}_1$.

As a product lattice consisting of power sets of sets $\mathcal{A}_1, \ldots, \mathcal{A}_3$ is isomorphic to the powerset of the union of these sets $\mathcal{A}_1 \cup \ldots \cup \mathcal{A}_3$, we shall use them interchangeably. We let:

$$\begin{array}{rcl} \mathcal{A}_1 & = & \{\texttt{inf(i)}, \texttt{cnct(j,i)}, \texttt{vac(i)}\} \\ \mathcal{A}_2 & = & \{\texttt{inf(th)}, \texttt{cnct(j,th)}, \texttt{vac(th)}\} \\ \mathcal{A}_3 & = & \{\texttt{inf(j)}\} \end{array}$$

We see that $2^{\mathcal{A}_1} \perp_{IC_{\mathcal{P}_1}} 2^{\mathcal{A}_2} \mid 2^{\mathcal{A}_3}$, by observing that:

$IC_{\mathcal{P}}^{\mathcal{A}_1,\mathcal{A}_3} = IC_{\mathcal{P}^{\mathcal{A}_1,\mathcal{A}_3}}$ *and* $IC_{\mathcal{P}}^{\mathcal{A}_2,\mathcal{A}_3} = IC_{\mathcal{P}^{\mathcal{A}_2,\mathcal{A}_3}}$ *where $\mathcal{P}^{\mathcal{A}_1,\mathcal{A}_3} = \{r_1, r_3, r_4\}$ and $\mathcal{P}^{\mathcal{A}_2,\mathcal{A}_3} = \{r_2, r_3, r_5\}$ It is easily verified that for every $x_j \subseteq \mathcal{A}_j$ $(j = 1, 2, 3)$, it holds that $IC_{\mathcal{P}}(x_1 \cup x_2 \cup x_3) \cap (\mathcal{A}_i \cup \mathcal{A}_3) = IC_{\mathcal{P}^{\mathcal{A}_i,\mathcal{A}_3}}(x_i \cup x_3)$ for any $i = 1, 2$.*

We now show structural similarities with conditional independence known from probability theory:

Fact 1. *Let an operator O on the product lattice $\otimes_{i\in\{1,2,3\}}\mathcal{L}_i$ s.t. $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ be given. Then for any $x_1 \otimes x_2 \otimes x_3 \in \bigotimes_{i\in\{1,2,3\}} \mathcal{L}_i$, it holds that:*

$$\begin{aligned} O(x_1 \otimes x_2 \otimes x_3) &= O_{1,2}(x_1 \otimes x_3) \otimes O_{2,3}(x_2 \otimes x_3)_{|2} \\ &= O_{1,2}(x_1 \otimes x_3)_{|1} \otimes O_{2,3}(x_2 \otimes x_3). \end{aligned}$$

Furthermore, for any $i, j = 1, 2$, $i \neq j$, $x_i \in \mathcal{L}_i, x_j, x'_j \in \mathcal{L}_j$ and $x_3 \in \mathcal{L}_3$ it holds that:

$$O(x_i \otimes x_j \otimes x_3)_{|i,3} = O(x_i \otimes x'_j \otimes x_3)_{|i,3}$$

However, this notion of conditional independence does show some differences with conditional independence as known from probability theory. For example, not all semi-graphoid-properties [1] are satisfied. In more detail, whereas *symmetry* (i.e. $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ implies $L_2 \perp\!\!\!\perp_O L_1 \mid L_3$) is obviously satisfied, the properties of *decomposition* (i.e. $L_1 \perp\!\!\!\perp_O L_2 \otimes L_3 \mid \emptyset$ implies $L_1 \perp\!\!\!\perp_O L_2 \mid \emptyset$) and *weak union* (i.e. $L_1 \perp\!\!\!\perp_O L_2 \otimes L_3 \mid \emptyset$ implies $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$) are not satisfied. Regarding decomposition, it should be noted that this property is undefined as we assume conditional independence over decompositions of the complete lattice. A violation of weak union is illustrated in the following example:

Example 2. *Consider the logic program $\mathcal{P} = \{a \leftarrow ; b \leftarrow \neg c; c \leftarrow \neg b\}$. Note that $2^{\{a\}} \perp\!\!\!\perp_{IC_{\mathcal{P}}} 2^{\{b,c\}} \mid \emptyset$. Yet it does* not *hold that $2^{\{a\}} \perp\!\!\!\perp_{IC_{\mathcal{P}}} 2^{\{a\}} \mid 2^{\{b\}}$, as:*

$$\begin{array}{lll} IC_{\mathcal{P}}(\{a\}) \cap \{a, c\} & = \{a, c\} & \neq \\ IC_{\mathcal{P}}(\{a, b\}) \cap \{a, c\} & = \{a\} & \end{array}$$

The reason for the failure of weak union is that we are not only interested in the behaviour of the operator O w.r.t. the conditionally independent sub-lattices L_1 and L_2, but also take into account the conditional pivot L_3. This is to be contrasted with probabilistic conditional independence where the defining condition $p(x_1 \mid x_3) = p(x_1 \mid x_2, x_3)$ only talks about L_1. The reason that here conditional pivots are taken into account is that we are interested in fixpoints of an operator. It might be interesting to look at a weaker notion of conditional independence w.r.t. operators that does not consider the conditional pivot in the output of the operator (and indeed, it is not hard to see that weak union is satisfied for such a notion), but due to our focus on fixpoints, we restrict attention to the stronger notion here.

We now undertake a study of the properties of operators that respect conditional independencies. We first note the following useful fact:

Lemma 1. *Let an operator O on the product lattice $\otimes_{i\in\{1,2,3\}}\mathcal{L}_i$ s.t. $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ and $i,j = 1,2$, $i \neq j$ be given. Then $O_{2,3}(x_2 \otimes x_3)_{|3} = O_{1,3}(x_1 \otimes x_3)_{|3}$.*

Proof. As $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$, for any $x_2 \in \mathcal{L}_2$, $O(x_1 \otimes x_2 \otimes x_3) = O_{1,3}(x_1 \otimes x_3) \otimes O_{2,3}(x_2 \otimes x_3)_{|2} = O_{1,3}(x_1 \otimes x_3)_{|1} \otimes O_{2,3}(x_2 \otimes x_3)$, which implies $O_{1,3}(x_1 \otimes x_3)_{|3} = O_{2,3}(x_2 \otimes x_3)_{|3}$. □

Fixpoints of an operator O respecting independence of L_1 and L_2 w.r.t. L_3 can be obtained by combining the fixpoints of $O_{1,3}$ and $O_{2,3}$. Thus, the search for fixpoints can be split into two parallel problems with a smaller search space.

Proposition 1. *Let an operator O on the product lattice $\otimes_{i\in\{1,2,3\}}S_i$ s.t. $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ be given. Then $x = O(x)$ iff $x_1 \otimes x_3 = O(x_1 \otimes x_3)$ and $x_2 \otimes x_3 = O(x_2 \otimes x_3)$.*

Proof. For the $\Rightarrow$-direction, suppose that $x = O(x)$. Since $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$, $O_{i,3}(x_i \otimes x_3) = O(x)_{|i,3} = x_{|i,3} = x_i \otimes x_3$ (for $i = 1,2$). For the $\Leftarrow$-direction, suppose that $x_1 \otimes x_3 = O(x_1 \otimes x_3)$ and $x_2 \otimes x_3 = O(x_2 \otimes x_3)$. As $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$, $O(x_1 \otimes x_2 \otimes x_3) = O_{1,3}(x_1 \otimes x_3) \otimes O_{2,3}(x_2 \otimes x_3)_{|2} = x_1 \otimes x_2 \otimes x_3 = x$. □

Monotonicity is preserved when moving between a product lattice and its components:

Proposition 2. *Let an operator O on the product lattice $\otimes_{i\in\{1,2,3\}}S_i$ s.t. $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ be given. Then $O : \bigotimes_{i\in\{1,2,3\}}\mathcal{L}_i \to \bigotimes_{i\in\{1,2,3\}}\mathcal{L}_i$ is $\leq_\otimes$-monotonic iff $O_{i,3} : \mathcal{L}_i \otimes \mathcal{L}_3 \to \mathcal{L}_i \otimes \mathcal{L}_3$ is $\leq^{i,3}_\otimes$-monotonic for $i = 1,2$.*

Proof. In what follows we let $I = \{1,2,3\}$. For the $\Rightarrow$-direction, suppose that O is $\leq^I_\otimes$-monotonic and consider some $x^1_1 \otimes x^1_3 \leq^{1,3}_\otimes x^2_1 \otimes x^2_3$. Notice that $O(x^1_1 \otimes x_2 \otimes x^1_3) \leq_\otimes O(x^2_1 \otimes x_2 \otimes x^2_3)$ for any $x_2 \in L_2$ (as O is $\leq^I_\otimes$-monotonic). This means that $O_{1,3}(x^1_1 \otimes x^1_3) \leq^{1,3}_\otimes O_{1,3}(x^2_1 \otimes x^2_3)$ by definition of $\leq_\otimes$ and since $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$. For the $\Leftarrow$-direction, suppose that $O_{i,3}$ are $\leq$-monotonic for $i = 1,2$. Consider some $x^1, x^2 \in \bigotimes_{i\in\{1,2,3\}} S_i$ with $x^1 \leq^I_\otimes x^2$. Then $O_{i,3}(x^1_{|i,3}) \leq^{i,3}_\otimes O_{i,3}(x^1_{|i,3})$ for $i = 1,2$ which implies $O_{1,3}(x^1_{|1,3}) \otimes O_{2,3}(x^1_{|2,3})_{|2} \leq^I_\otimes O_{1,3}(x^2_{|1,3}) \otimes O_{2,3}(x^2_{|2,3})_{|2}$ by definition of $\leq^I_\otimes$. With conditional independence, $O(x^j) = O_{1,3}(x^j_{|1,3}) \otimes O_{2,3}(x^j_{|2,3})_{|2}$ for $j = 1,2$. Thus, $O(x^1) \leq^I_\otimes O(x^2)$. □

Likewise, at least for monotonic operators over complete lattices, the *least* fixed points can be obtained by combining the least fixed points of conditionally independent sub-lattices:

Proposition 3. *Let a $\leq_\otimes$-monotonic operator O on the complete product lattice $\otimes_{i\in\{1,2,3\}}S_i$ s.t. $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ be given. Then x is a least fixed point of O iff $x_{|i,3}$ is a least fixed point of $O_{i,3}$ (for $i = 1,2$).*

Proof. Suppose first that $x_1 \otimes x_2 \otimes x_3$ is the least fixed point of O. We show that $x_1 \otimes x_3$ is a least fixed point of $O_{1,3}$ (which suffices with symmetry). With Proposition 1, $x_1 \otimes x_3 = O_{1,3}(x_1 \otimes x_3)$. It thus suffices to show that for any fixed point $x'_1 \otimes x'_3$ of $O_{1,3}$, $x'_1 \otimes x'_3 \geq x_1 \otimes x_3$. Assume thus that $x'_1 \otimes x'_3 = O_{1,3}(x'_1 \otimes x'_3)$. First, observe that $\bot_2 \otimes x_3 \leq x'_2 \otimes x_3$ for any x'_2. By Lemma 1, $O_{2,3}(x'_2 \otimes x'_3)_{|3} = O_{1,3}(x'_1 \otimes x'_3) = x'_3$ for any $x'_2 \in \mathcal{L}_2$. Thus, $O_{2,3}(\bot_2 \otimes x'_3) \geq \bot_2 \otimes x'_3$, i.e. $\bot_2 \otimes x'_3$ is a pre-fixpoint of $O_{2,3}$. We can apply $O_{2,3}$ inductively, and, as it is a $\leq$-monotonic operator (Proposition 2), a fixpoint is guaranteed to exist. Thus, there is some $x'_2 \in \mathcal{L}_2$ s.t. $O(x'_2 \otimes x'_3) = x'_2 \otimes x'_3$. This means that $x'_1 \otimes x'_2 \otimes x'_3$ is a fixpoint of O, which implies $x_1 \otimes x_2 \otimes x_3 \leq_\otimes x'_1 \otimes x'_2 \otimes x'_3$, which on its turn implies (by definition of $\leq_\otimes$), $x_1 \otimes x_3 \leq x'_1 \otimes x'_3$.

Suppose now that $x_{|i,3}$ is a least fixed point of $O_{i,3}$ (for $i = 1,2$). With Proposition 1, $x_1 \otimes x_2 \otimes x_3$ is a fixed point of O. We show that for any fixed point x' of O, $x' \geq x_1 \otimes x_2 \otimes x_3$. Indeed, with Proposition 1, $x'_{|i,3}$ is a fixed point of $O_{i,3}$, which implies that $x'_{|i,3} \geq x_{|i,3}$. By definition of $\leq_\otimes$, $x'_1 \otimes x'_2 \otimes x'_3 \geq x_1 \otimes x_2 \otimes x_3$. □

4. Conditional Independence and Approximation Fixpoint Theory

The notion of conditional independence is immediately applicable to approximation operators. In this section, we will derive results on the modularisation of AFT-based semantics based on the results derived in the previous section.

As observed in previous work [9], the bilattice $\mathcal{L}^2$ of a product lattice $\mathcal{L} = \otimes_{i\in I}\mathcal{L}_i$ is isomorphic to the product lattice of bilattices $\bigotimes_{i\in I}\mathcal{L}^2_i$, and we will sometimes move between these two constructs without further remarks [9].

As an approximation operator is a $\leq_i$-monotonic operator, we immediately obtain that the search for Kripke-Kleene fixpoints, as well as any regular fixpoints, can be split on the basis of conditional independence:

Proposition 4. *Let an approximation operator $\mathcal{O}$ over a bilattice of the product lattice $\otimes_{i\in\{1,2,3\}}\mathcal{L}_i$ be given s.t. $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{O}} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$. Then the following hold:*

- *(x, y) is the Kripke-Kleene fixpoint of $\mathcal{O}$ iff $(x_{|i,3}, y_{|i,3})$ is the Kripke-Kleene fixpoint of $\mathcal{O}_{i,3}$ for $i = 1, 2$.*
- *(x, y) is a fixpoint of $\mathcal{O}$ iff $(x_{|i,3}, y_{|i,3})$ is a fixpoint of $\mathcal{O}_{i,3}$ for $i = 1, 2$.*

Proof. This is an immediate consequence of Propositions 1, 2 and 3. □

We now turn our considerations to the stable operators. As a preliminary, we investigate the relation between an approximation operator and the lower and upper-bound component of this operator when it comes to respecting indpendencies. It turns out that the component operators $\mathcal{O}_l$ and $\mathcal{O}_u$ respect conditional independencies, and, vice-versa, that the respect of the two component operators of conditional independencies implies respect of these independencies by the approximation operator:

Proposition 5. *Let an approximation operator $\mathcal{O}$ over a bilattice of the product lattice $\otimes_{i\in\{1,2,3\}}\mathcal{L}_i$ be given. Then $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{O}} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$ iff $\mathcal{L}_1 \perp\!\!\!\perp_{\mathcal{O}_l} \mathcal{L}_2 \mid \mathcal{L}_3$ and $\mathcal{L}_1 \perp\!\!\!\perp_{\mathcal{O}_u} \mathcal{L}_2 \mid \mathcal{L}_3$.*

Proof. For the ⇒-direction, suppose that $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{O}} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$, i.e. there are some $\mathcal{O}_{i,3} : \mathcal{L}_i \otimes \mathcal{L}_3 \to \mathcal{L}_i \otimes \mathcal{L}_3$ s.t. $\mathcal{O}((x_1, y_1) \otimes (x_2, y_2) \otimes (x_3, y_3))_{|i,3} = \mathcal{O}_{i,3}((x_i, y_i) \otimes (x_3, y_3))$ (for $i = 1, 2$). As $\mathcal{O}((x_1, y_1) \otimes (x_2, y_2) \otimes (x_3, y_3)) = (\mathcal{O}_l(x_1 \otimes x_2 \otimes x_3, y_1 \otimes y_2 \otimes y_3), \mathcal{O}_u(x_1 \otimes x_2 \otimes x_3, y_1 \otimes y_2 \otimes y_3))$, this means there are some $(\mathcal{O}_l)_{i,3}$ and $(\mathcal{O}_u)_{i,3}$ s.t. $\mathcal{O}_\dagger(x_1 \otimes x_2 \otimes x_3, y_1 \otimes y_2 \otimes y_3)_{|i,3} = (\mathcal{O}_\dagger)_{i,3}(x_i \otimes x_3, y_i \otimes y_3)$ for $\dagger \in \{l, u\}$ and $i = 1, 2$, which implies $\mathcal{L}_1 \perp\!\!\!\perp_{\mathcal{O}_l} \mathcal{L}_2 \mid \mathcal{L}_3$ and $\mathcal{L}_1 \perp\!\!\!\perp_{\mathcal{O}_u} \mathcal{L}_2 \mid \mathcal{L}_3$.

The ⇐-direction is similar. □

We can now show that stable operators respect the conditional independencies respected by the approximation operator from which they are derived:

Proposition 6. *Let an approximation operator $\mathcal{O}$ over a bilattice of the complete product lattice $\otimes_{i\in\{1,2,3\}}\mathcal{L}_i$ be given s.t. $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{O}} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$. Then $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{C}(\mathcal{O}_l)} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$ and $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{C}(\mathcal{O}_u)} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$.*

Proof. As $L_1^2 \perp\!\!\!\perp_{\mathcal{O}} L_2^2 \mid L_3^2$, with Proposition 5, $L_1 \perp\!\!\!\perp_{\mathcal{O}_l} L_2 \mid L_3$. As $\mathcal{O}_l(., y)$ is $\leq$-monotonic ([16, Proposition 6]), $\text{lfp}(\mathcal{O}_l(., y)) = \text{lfp}((\mathcal{O}(., y))_{1,3}) \otimes (\text{lfp}((\mathcal{O}_l(., y))_{2,3})_{|2})$ (Proposition 3). As $(\mathcal{O}_l(x, y)) = (\mathcal{O}_l)_{1,3}(x_{|1,3}, y_{|1,3}) \otimes (\mathcal{O}_u)_{2,3}(x_{|2,3}, y_{|2,3})_{|2}$ for any $x \in \mathcal{L}$ (in view of $L_1 \perp\!\!\!\perp_{\mathcal{O}_l} L_2 \mid L_3$), we see that $\mathcal{O}_l(., y)))_{1,3} = (\mathcal{O}_l)_{1,3}(., y_{|1,3})$. Thus, $\text{lfp}(\mathcal{O}_l(., y)) = \text{lfp}((\mathcal{O}_l)_{1,3})(., y_{1,3})) \otimes \text{lfp}((\mathcal{O}_l)_{2,3}(., y_{2,3}))_{|2}$. As $\mathcal{C}(\mathcal{O}_l)(y) = \text{lfp}(\mathcal{O}(., y)$ for any $y \in \mathcal{L}$, this concludes the proof. □

Proposition 7. *Let an approximation operator $\mathcal{O}$ over a bilattice of the product lattice $\otimes_{i\in\{1,2,3\}}S_i$ be given s.t. $\mathcal{L}_1^2 \perp\!\!\!\perp_{\mathcal{O}} \mathcal{L}_2^2 \mid \mathcal{L}_3^2$. Then:*

1. *(x, y) is a fixpoint of $S(\mathcal{O})$ iff $(x_{|i,3}, y_{|i,3})$ is a fixpoint of $S(\mathcal{O}_{i,3})$ for $i = 1, 2$.*
2. *(x, y) is the well-founded fixpoint of $\mathcal{O}$ iff $(x_{|i,3}, y_{|i,3})$ is the well-founded fixpoint of $\mathcal{O}_{i,3}$ for $i = 1, 2$.*

Proof. This follows immediately from Propositions 3 and 6. □

5. Application to Logic Programs

In this section, we apply the theory developed in the previous section to normal logic programs. We can avoid clutter with a slight abuse of notation by writing $\mathcal{A}_1 \perp\!\!\!\perp_{\mathcal{P}} \mathcal{A}_2 \mid \mathcal{A}_3$ to denote $2^{\mathcal{A}_1} \perp\!\!\!\perp_{\mathcal{IC}_{\mathcal{P}}} 2^{\mathcal{A}_2} \mid 2^{\mathcal{A}_3}$ (for any $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \subseteq \mathcal{A}_{\mathcal{P}}$).

We first define what we call the *marginalisation* of a program w.r.t. a set of atoms:

Definition 5. *Let a normal logic program $\mathcal{P}$ and some $\mathcal{A} \subseteq \mathcal{A}_{\mathcal{P}}$ be given. We define $\mathcal{P}_{\mathcal{A}}$ as the program obtained by replacing in every rule $r \in \mathcal{P}$ every occurrence of an atom $p \in \mathcal{A}$ by $\bot$.*

For example, $\{p \leftarrow q, r, \neg s\}_{\{r,s\}} = \{p \leftarrow q, \bot, \top\}$. Given a program $\mathcal{P}$ inducing a conditional independence $\mathcal{A}_1 \perp\!\!\!\perp_{\mathcal{P}} \mathcal{A}_2 \mid \mathcal{A}_3$, the marginalisation $\mathcal{P}_{\mathcal{A}_2}$ gives use the immediate consequence operator for the sublattice $\mathcal{A}_1 \cup \mathcal{A}_3$:

Proposition 8. *Let a normal logic program $\mathcal{P}$ be given for which $\mathcal{A}_{\mathcal{P}}$ is partitioned into $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ s.t. $\mathcal{A}_1 \perp\!\!\!\perp_{\mathcal{P}} \mathcal{A}_2 \mid \mathcal{A}_3$. Then $\mathcal{IC}_{\mathcal{P}}^{i,3} = \mathcal{IC}_{\mathcal{P}_{\mathcal{A}_j}}$ (for $i, j = 1, 2$, $i \neq j$).*

Proof. Consider some arbitrary but fixed $i, j = 1, 2$, $i \neq j$. In view of $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$, $\mathcal{IC}_{\mathcal{P}}^{i,3}(x_i \cup x_3) = \mathcal{IC}_{\mathcal{P}}(x_i \cup x_j \cup x_3)_{|i,3}$ for any $x_j \subseteq \mathcal{A}_j$. As $\mathcal{IC}_{\mathcal{P}_{\mathcal{A}}}(x_i \cup \emptyset \cup x_3) = \mathcal{IC}_{\mathcal{P}}(x_i \cup \emptyset \cup x_3)$, this concludes the proof. □

We start by working out what the results in the previous sections mean for the semantics of logic programs. In particular, the search for supported, (partial) stable and well-founded models can be split up along conditionally independent sub-alphabets:

Corollary 1. *Let a normal logic program $\mathcal{P}$ be given for which $\mathcal{A}_\mathcal{P}$ is partitioned into $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ s.t. $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$. $x_1 \cup x_2 \cup x_3$ is a supported (respectively three-valued stable) model of $\mathcal{P}$ iff $x_i \cup x_3$ is a supported (respectively three-valued stable) model of $\mathcal{P}_{|\mathcal{A}_i \cup \mathcal{A}_3}$ (for $i = 1, 2$). The well-founded model of $\mathcal{P}$ can be obtained as $(x_1 \cup x_2 \cup x_3, y_1 \cup y_2 \cup y_3)$, where $(x_i \cup x_3, y_i \cup y_3)$ is the well-founded model of $\mathcal{P}_{\mathcal{A}_j}$ (for $i, j = 1, 2$, $i \neq j$).*

We now make some observations on how to detect conditional independencies in a logic program. We first need some further preliminaries. The *dependency order* for a logic program $\mathcal{P}$, $\leq^\mathcal{P}_{\text{dep}} \subseteq \mathcal{A}_\mathcal{P} \times \mathcal{A}_\mathcal{P}$, is defined as $p \leq^\mathcal{P}_{\text{dep}} q$ iff there is some $r \in \mathcal{P}$ where q is the head of r and p occurs in the body of r. The *dependency graph*, denoted $\text{DP}(\mathcal{P})$ of $\mathcal{P}$ is the corresponding Hasse diagram of $\leq^\mathcal{P}_{\text{dep}}$.

A first conjecture could be that, a sufficient criterion fo $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$ is that $\mathcal{A}_3$ graphically separates $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e. given $\text{DP}(\mathcal{P}) = \langle \mathcal{A}_\mathcal{P}, V\rangle$, a set $\mathcal{A}_3$ s.t. $\langle \mathcal{A}_\mathcal{P} \setminus \mathcal{A}_3, V \cap ((\mathcal{A}_\mathcal{P} \setminus (\mathcal{A}_3) \times (\mathcal{A}_\mathcal{P} \setminus (\mathcal{A}_3))$ consists of two disconnected subgraphs $\langle \mathcal{A}_1, V \cap (\mathcal{A}_1 \times \mathcal{A}_1)\rangle$ and $\langle \mathcal{A}_2, V \cap (\mathcal{A}_2 \times \mathcal{A}_1)\rangle$ induces the conditional independence $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 | \mathcal{A}_3$. However, this conjecture is too naive:

Example 3. *Consider the program $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ where $\mathcal{P}_1 = \{a_1 \leftarrow \neg b_1; b_1 \leftarrow \neg a_1; e \leftarrow b_1\}$ and $\mathcal{P}_2 = \{a_2 \leftarrow \neg b_2; b_2 \leftarrow \neg a_2; e \leftarrow b_2\}$. This program has the following dependency graph:*

$$\begin{tikzcd} a_1 \arrow[r, leftrightarrow] & b_1 \arrow[r] & e & b_2 \arrow[l] \arrow[r, leftrightarrow] & a_2 \end{tikzcd}$$

We could conjecture the independency $\{a_1, b_1\} \perp\!\!\!\perp_\mathcal{P} \{a_2, b_2\}|\{e\}$, but this does not hold, as

$$\begin{aligned} IC_\mathcal{P}(\{a_1, b_2\})_{|\{a_1,b_1,e\}} &= \{a_1, e\} \\ \neq IC_\mathcal{P}(\{a_1\}) &= \{a_1\}. \end{aligned}$$

A slightly more complicated graphical criterion is a sufficient condition, though. In more detail, if $\mathcal{A}_3$ graphically seperates $\mathcal{A}_1$ and $\mathcal{A}_2$ in $\text{DP}(\mathcal{P})$, and if the program is stratified in a lower layer $\mathcal{A}_3$ and a higher layer $\mathcal{A}_1 \cup \mathcal{A}_2$, then the conditional independency $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$ holds:

Proposition 9. *Let a logic program $\mathcal{P}$ with $\text{DP}(\mathcal{P}) = \langle \mathcal{A}_\mathcal{P}, V\rangle$ be given s.t. the following conditions hold*

1. *there is some $\mathcal{A}_3 \subseteq \mathcal{A}_\mathcal{P}$ s.t. $\langle \mathcal{A}_\mathcal{P} \setminus \mathcal{A}_3, V \cap ((\mathcal{A}_\mathcal{P} \setminus (\mathcal{A}_3) \times (\mathcal{A}_\mathcal{P} \setminus (\mathcal{A}_3))$ consists of two disconnected subgraphs $\langle \mathcal{A}_1, V \cap (\mathcal{A}_1 \times \mathcal{A}_1)\rangle$ and $\langle \mathcal{A}_2, V \cap (\mathcal{A}_2 \times \mathcal{A}_2)\rangle$, and*
2. *for every $a \in \mathcal{A}_3$ and $b \in \mathcal{A}_i$ ($i = 1, 2$), $b <_{\text{dep}} a$.*

Then $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$ holds.

Proof. Suppose the conditions of the proposition hold. Then for any $p \leftarrow \bigwedge \Delta \wedge \bigwedge \Theta^\neg$ (where $\Theta^\neg = \{\neg p \mid p \in \Theta\}$), (1) $\Delta \cup \Theta \cap \mathcal{A}_i \neq \emptyset$ implies $p \in \mathcal{A}_i$ (for $i = 1, 2$), and (2) $\Delta \cup \Theta \cup \{p\} \subseteq \mathcal{A}_i \cup \mathcal{A}_3$ (for $i = 1, 2$). From (1), it follows that †: $\mathcal{IC}_\mathcal{P}(x_1 \cup x_2 \cup x_3)_{|3} = IC_{\mathcal{P}_{\mathcal{A}_1 \cup \mathcal{A}_2}}(x_3)$ for any $x_i \subseteq \mathcal{A}_i$ ($i = 1, 2, 3$). From (2) and †, it then follows that: $\mathcal{IC}_\mathcal{P}(x_1 \cup x_2 \cup x_3)_{|i,3} = \mathcal{IC}_{\mathcal{P}_{\mathcal{A}_k}}(x_i \cup x_3)$ for any $x_j \subseteq \mathcal{A}_j$ ($j = 1, 2, 3$) and $i, k = 1, 2$ and $i \neq k$. □

This gives an example of how conditional independence can, at least partially, be identified on the basis of the syntax of a logic program. The search for more comprehensive, potentially even necessary, criteria for identifying conditional independencies are an avenue for future work.

6. Related Work

In this section, related work is discussed. We first discuss Darwiche's notion of conditional independence [2], stratification as studied in approximation fixpoint theory [9] and treewidth-based decompositions of logic programs in detail, and then make shorter comparisons to other related works.

Darwiche's Logical Notion of Independence In the context of classical logic, a notion of conditional independence was proposed by Darwiche [2]. Darwiche assumes a database Δ (i.e. a set of propositional formulas), which is used as a background theory for inferences. The idea behind conditional independence is then that a database Δ sanctions the independence of two sets of atoms x_1 and x_2 conditional on a third set of atoms x_3 if, given full information about x_3, inferences about x_1 are independent from any information about x_2. In other words, given a set of formulas Δ and three disjoint sets of atoms x_1, x_2 and x_3 be given, $x_1 \perp\!\!\!\perp^{\mathsf{D}}_\Delta x_2 \mid x_3$ iff for every formula ϕ_1 based on x_1, ϕ_2 based on x_2 and complete conjunction of literals ϕ_3 based on x_3 s.t. $\Delta \cup \{\phi_3, \phi_2\}$ is consistent, the following holds:

$$\Delta \cup \{\phi_3\} \models \phi_1 \quad \text{iff} \quad \Delta \cup \{\phi_2, \phi_3\} \models \phi_1$$

Even though the application of our notion of conditional independence to operators ranging over sets of possible worlds (which is required to give an operator-based characterisation of propositional logic) is outside the scope of this paper, we can nevertheless show a close connection between our notion of conditional independence and the one formulated by Darwiche by defining inference based on a logic program as follows (which gives rise to a special case of *simple-minded output* as known from input/output logics [20]):

Definition 6. *Given a logic program $\mathcal{P}$ and formulas ϕ, ψ based on $\mathcal{A}_\mathcal{P}$, we define: $\phi \models_\mathcal{P} \psi$ if for every $x \subseteq \mathcal{A}_\mathcal{P}$ s.t. $x(\phi) = \mathsf{T}$, $IC_\mathcal{P}(x)(\psi) = \mathsf{T}$.*

We can now show that our notion of conditional independence implies Darwiche's notion of conditional independence, interpreted in the setting of inference based on logic programs:

Proposition 10. *Let a program $\mathcal{P}$ for which $\mathcal{A}_\mathcal{P}$ is partitioned into $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ s.t. $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$, some ϕ_1 based on $\mathcal{A}_1$, some $\mathcal{A}_2$ based on $\mathcal{A}_2$ and a complete conjunction of literals ϕ_3 based on $\mathcal{A}_3$ be given. Then $\phi_3 \models_\mathcal{P} \phi_1$ iff $\phi_3 \wedge \phi_2 \models_\mathcal{P} \phi_1$.*

Proof. Suppose that the assumptions of this proposition holds. The $\Rightarrow$-direction is immediate as $\models_\mathcal{P}$ is monotonic.

Suppose now that $\phi_3 \wedge \phi_2 \models_\mathcal{P} \phi_1$. Then for every $x_1 \cup x_2 \cup x_3 \subseteq \mathcal{A}_\mathcal{P}$ s.t. $x_1 \cup x_2 \cup x_3(\phi_3 \wedge \phi_2) = \mathsf{T}$, $IC_\mathcal{P}(x_1 \cup x_2 \cup x_3)(\phi_1) = \mathsf{T}$. Notice that there is a single $x_3 \subseteq \mathcal{A}_3$ s.t. $x_3(\phi_3)$ and $x_1 \cup x_2 \cup x_3(\phi_3 \wedge \phi_2) = \mathsf{T}$ is independent of x_1 (i.e. $x_1^\star \cup x_2 \cup x_3(\phi_3 \wedge \phi_2) = \mathsf{T}$ for any $x_1^\star \subseteq \mathcal{A}_1$). As $\mathcal{A}_1 \perp\!\!\!\perp_\mathcal{P} \mathcal{A}_2 \mid \mathcal{A}_3$, $IC_\mathcal{P}(x_1 \cup x_2^\star \cup x_3)_{|1,3} = IC_\mathcal{P}(x_1 \cup x_2^\star \cup x_3)_{|1,3}$ for any $x_2^\star \subseteq \mathcal{A}_2$, we see that for any $x_1' \subseteq \mathcal{A}_1$, $IC_\mathcal{P}(x_1' \cup x_2^\star \cup x_3)_{|1}(\phi_1) = \mathsf{T}$, which implies $\phi_3 \models_\mathcal{P} \phi_1$. □

Splitting Operators A concept related to conditional independence studied in approximation fixpoint theory is that of *stratification* [9]. This work essentially generalizes the idea of *splitting* as known from logic programming, where the idea is to divide a logic program in layers such that computations in a given layer only depend on rules in the layer itself or layers below. For example, the program $\{q \leftarrow\sim r; r \leftarrow\sim s; s \leftarrow\sim p\}$ can be stratified in the layers $\{p\}, \{s, r\}, \{q\}$. This concept was formulated purely algebraically by Vennekens, Gilis and Denecker [9]. Our study of conditional independence took inspiration from this work in using product lattices as an algebraic tool for dividing lattices, and many proofs and results in our paper are similar to those shown for stratified operators [9]. Conceptually, stratification and conditional independence seem somewhat orthogonal, as conditional independence allows to divide a lattice "horizontally" into independent parts, whereas stratification allows to divide a lattice "vertically" in layers that incrementally depend on each other. It might be therefore rather surprising that conditional independence can be seen as a special case of stratification.

We first recall the definitions on stratifiability. First, we denote, for a product lattice $\bigotimes_{i\in I} L_i$, $x \in \bigotimes_{i\in I} L_i$ and $j \in I$, $x_{|\leq j} = x_{|\{i\in I | i \leq j\}}$. An operator is *stratifiable* (over $\bigotimes_{i\in I} L_i$) iff for every $x^1, x^2 \in \bigotimes_{i\in I} L_i$ and every $j \in I$, if $x^1_{|\leq j} = x^2_{|\leq j}$ then $O(x)_{|\leq j} = O(y)_{|\leq j}$.

Proposition 11. *Let a $\leq_\otimes$-monotonic operator O on the product lattice $\bigotimes_{i\in\{1,2,3\}} L_i$ be given. Then $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$ iff O is stratifiable over $L_1 \otimes (L_2 \otimes L_3)$ and $L_2 \otimes (L_1 \otimes L_3)$.*

Proof. For the $\Rightarrow$-direction, suppose that $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$. Suppose that $x^1, x^2 \in \bigotimes_{i\in\{1,2,3\}} L_i$ and that $x_1^1 \otimes x_3^1 = x_1^2 \otimes x_3^1$. Then, as $L_1 \perp\!\!\!\perp_O L_2 \mid L_3$, $O(x^1)_{|\{1,3\}} = O^{1,3}(x_1^1 \otimes x_3^1) = O^{1,3}(x_1^2 \otimes x_3^2) = O(x^2)_{|\{1,3\}}$.

For the $\Leftarrow$-direction, suppose that O is stratifiable over $L_1 \otimes (L_2 \otimes L_3)$ and $L_2 \otimes (L_1 \otimes L_3)$. Then we can define $O(x_1 \otimes \otimes x_3) = O(x_1 \otimes x_2 \otimes x_3)_{|i,3}$ for any $x_2 \in L_2$ as $O(x_1 \otimes x_2 \otimes x_3)_{|i,3} = O(x_1 \otimes x_2' \otimes x_3)_{|i,3}$ for any $x_2' \in L_2$. □

On the other hand, stratification does not, in general, imply conditional independence, as conditional independence requires symmetry:

Example 4. *Consider $\mathcal{P} = \{q \leftarrow\sim r; r \leftarrow\sim s; s \leftarrow\sim p\}$. Then $\mathcal{P}$ can be stratified in the layers $\{p\}, \{s, r\}, \{q\}$ yet $\{q\}$ is not conditionally independent from any of the other atoms.*

Decomposing Logic Programs A lot of effort has been devoted to the study of the *paramterization* of the computational complexity of various computational tasks using *treewidth decompositions* as a parameter [21]. These results show that the computational effort required in solving a problem is not a function of the overall size of the problem, but rather of certain structural parameters of the problem, i.e. the treewidth of a certain representation of the problem. These techniques have also been successfully applied to answer set programming [22]. In these works, the treewidth of the tree decomposition of the dependence graph $\mathsf{DP}(\mathcal{P})$ and incidence graph (which also contains vertices for rules) of a logic program are used as parameters to obtain fixed-parameter tractability results. We first notice that a treewidth-decomposition does not always indicate a conditional independence (we refer here to the relevant literature for background on treewidth-decompositions [22]). Indeed, Example 3 provides a case in point, as the tree-decomposition would suggest the conditional independence $\{a_1, b_1\} \perp\!\!\!\perp_{\mathcal{P}} \{a_2, b_2\} \mid \{e\}$, which does not hold. On the other hand, given that we phrased conditional independence semantically, some decompositions are not visible using the purely syntactic approach from [22]:

Example 5. *Let $\mathcal{P} = \{p \leftarrow q, \sim q; q \leftarrow p, \sim p; q \leftarrow \sim r; p \leftarrow \sim s\}$, with the following dependency graph:*

$$\begin{tikzcd} r \arrow[r] & q \arrow[r, leftrightarrow] & p & s \arrow[l] \end{tikzcd}$$

The only treewidth decomposition is the following:

$$\begin{tikzcd} & \{p, q\} \arrow[dl, dash] \arrow[dr, dash] & \\ \{q, r\} & & \{p, s\} \end{tikzcd}$$

However (since the rules $p \leftarrow q, \sim q$ and $q \leftarrow p, \sim p$ are never applicable), it can be verified that $\{q, r\} \perp\!\!\!\perp_{\mathcal{P}} \{p, s\} \mid \emptyset$.

Thus, the exact relationships between conditional independence and treewidth decompositions seem rather intricate and remain to be investigated.

Other operator-based formalisms have been analysed in terms of treewidth decompositions [23, 24]. A benefit of our operator-based approach is that all results are purely algebraic and therefore language-independent, which means that applications to specific formalisms are derived as straightforward corollaries. Furthermore, the results for AFT-based semantics, which subsume many KR-formalisms (see [8] for an overview), are not restricted to the total stable fixpoints, in contrast to many studies on fixed-parameter tractability. An investigation into the benefits to computational complexity on the basis of conditional independence is one of the most important avenues for future work, and we conjecture that fixed parameter tractability results based on the decomposition in modules using conditional independence will be obtainable.

Other Related Work Conditional independence has been investigated in several other logic-based frameworks, such as (iterated) belief revision [5, 25], conditional logics [26] and formal argumentation [27, 28]. The benefit of our work is that the algebraic nature allows for the straightforward application to other formalisms with a fixpoint semantics.

7. Conclusion

In this paper, the concept of conditional independence, well-known from probability theory, was formulated and studied for operators. This allows to use this concept to a wide variety of formalisms for knowledge representation that admit an operator-based characterisation. As a proof-of-concept, we have applied it to the semantics of normal logic programs.

There exist several fruitful avenues for future work. Firstly, we will investigate whether and how modularisation based on conditional independence can be used to obtain purely algebraic fixed-parameter results. Secondly, we want to investigate related notions of independence, such as context-specific independence [29]. A third avenue for future work is a more extensive application of the theory to concrete formalisms, both in breadth (by applying the theory to further formalisms) and in depth (e.g. by investigating more syntactic methods to identify conditional independencies, and by evaluating the computational gain experimentally).

References

[1] J. Pearl, D. Geiger, T. Verma, Conditional independence and its representations, Kybernetika 25 (1989) 33–44.

[2] A. Darwiche, A logical notion of conditional independence: properties and applications, Artificial Intelligence 97 (1997) 45–82.

[3] J. Lang, P. Liberatore, P. Marquis, Conditional independence in propositional logic, Artificial Intelligence 141 (2002) 79–121.

[4] G. Kern-Isberner, J. Heyninck, C. Beierle, Conditional independence for iterated belief revision, in: L. D. Raedt (Ed.), Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, 2022, pp. 2690–2696. doi:10.24963/ijcai.2022/373, main Track.

[5] M. J. Lynn, J. P. Delgrande, P. Peppas, Using conditional independence for belief revision, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, 2022, pp. 5809–5816.

[6] Jesse Heyninck, G. Kern-Isberner, T. A. Meyer, J. Haldimann, C. Beierle, Conditional syntax splitting for non-monotonic inference operators, in: Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI'23), 2023.

[7] M. Denecker, V. Marek, M. Truszczyński, Uniform semantic treatment of default and au-

toepistemic logics, Artificial Intelligence 143 (2003) 79–122.

[8] J. Heyninck, B. Bogaerts, Non-deterministic approximation operators: ultimate operators, semi-equilibrium semantics and aggregates (full version), CoRR abs/2305.10846 (2023). URL: https://doi.org/10.48550/arXiv.2305.10846. doi:10.48550/arXiv.2305.10846. arXiv:2305.10846.

[9] J. Vennekens, D. Gilis, M. Denecker, Splitting an operator: Algebraic modularity results for logics with fixpoint semantics, ACM Transactions on computational logic (TOCL) 7 (2006) 765–797.

[10] B. Bogaerts, J. Vennekens, M. Denecker, Grounded fixpoints and their applications in knowledge representation, Artificial Intelligence 224 (2015) 51 – 71. doi:10.1016/j.artint.2015.03.006.

[11] M. Truszczyński, Strong and uniform equivalence of nonmonotonic theories–an algebraic approach, Annals of Mathematics and Artificial Intelligence 48 (2006) 245–265.

[12] J. Heyninck, O. Arieli, B. Bogaerts, Non-deterministic approximation fixpoint theory and its application in disjunctive logic programming, CoRR abs/2211.17262 (2022). URL: https://doi.org/10.48550/arXiv.2211.17262. doi:10.48550/arXiv.2211.17262. arXiv:2211.17262.

[13] M. H. van Emden, R. A. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (1976) 733–742.

[14] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New generation computing 9 (1991) 365–385.

[15] T. C. Przymusinski, The well-founded semantics coincides with the three-valued stable semantics, Fundamenta Informaticae 13 (1990) 445–463.

[16] M. Denecker, V. Marek, M. Truszczyński, Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning, in: Logic-based Artificial Intelligence, volume 597 of *The Springer International Series in Engineering and Computer Science*, Springer, 2000, pp. 127–144.

[17] M. Denecker, V. W. Marek, M. Truszczynski, Ultimate approximations in nonmonotonic knowledge representation systems, in: Proceedings of the Eights International Conference on Principles of Knowledge Representation and Reasoning, 2002, pp. 177–190.

[18] N. Pelov, M. Denecker, M. Bruynooghe, Well-founded and stable semantics of logic programs with aggregates, Theory and Practice of Logic Programming 7 (2007) 301–353.

[19] A. Van Gelder, K. A. Ross, J. S. Schlipf, The well-founded semantics for general logic programs, Journal of the ACM 38 (1991) 619–649.

[20] D. Makinson, L. van der Torre, What is input/output logic?, in: Foundations of the Formal Sciences II: Applications of Mathematical Logic in Philosophy and Linguistics, Papers of a Conference held in Bonn, November 10–13, 2000, Springer, 2003, pp. 163–174.

[21] G. Gottlob, F. Scarcello, M. Sideri, Fixed-parameter complexity in ai and nonmonotonic reasoning, Artificial Intelligence 138 (2002) 55–86.

[22] J. K. Fichte, M. Hecher, M. Morak, S. Woltran, Answer set solving with bounded treewidth revisited, in: Logic Programming and Nonmonotonic Reasoning: 14th International Conference, LPNMR 2017, Espoo, Finland, July 3-6, 2017, Proceedings 14, Springer, 2017, pp. 132–145.

[23] J. K. Fichte, M. Hecher, I. Schindler, Default logic and bounded treewidth, Information and Computation 283 (2022) 104675.

[24] W. Dvořák, R. Pichler, S. Woltran, Towards fixed-parameter tractable algorithms for abstract argumentation, Artificial Intelligence 186 (2012) 1–37.

[25] G. Kern-Isberner, J. Heyninck, C. Beierle, Conditional independence for iterated belief revision, in: 31st International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence, 2022, pp. 2690–2696.

[26] J. Heyninck, G. Kern-Isberner, T. Meyer, Conditional syntax splitting, lexicographic entailment and the drowning effect (2022).

[27] T. Rienstra, M. Thimm, K. Kersting, X. Shao, Independence and d-separation in abstract argumentation, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, volume 17, 2020, pp. 713–722.

[28] S. A. Gaggl, S. Rudolph, H. Strass, On the decomposition of abstract dialectical frameworks and the complexity of naive-based semantics, Journal of Artificial Intelligence Research 70 (2021) 1–64.

[29] C. Boutilier, N. Friedman, M. Goldszmidt, D. Koller, Context-specific independence in bayesian networks, in: Proc. 12th Conf. on Uncertainty in Artificial Intelligence (UAI'96), 1996, pp. 115–123.