On Approximation of the Semantic Operators Determined by Bilattice-Based Logic Programs

Ekaterina Komendantskaya[1], Anthony Seda[2], and Vladimir Komendantsky[3]

[1] Department of Mathematics, University College Cork, Cork, Ireland
e.komendantskaya@mars.ucc.ie
[2] Department of Mathematics, University College Cork, Cork, Ireland
a.seda@ucc.ie
[3] Boole Centre for Research in Informatics, University College Cork, Cork, Ireland
v.komendantsky@bcri.ucc.ie * **

Abstract. We introduce the class of bilattice-based annotated logic programs (BAPs). These programs extend the general annotated programs of Kifer and Subrahmanian to the bilattice case. The immediate consequence operator $\mathcal{T}_P$ is defined for BAPs and its continuity is proven. A theorem of Seda concerning the approximation of the least fixed point of the two-valued T_P-operator is generalized to the case of BAPs. Finally, alternative bilattice-based logic programs are compared with BAPs, and theorems on the computation of the least fixed points of their semantic operators are established. This gives an extension of the theorem on approximation of the least fixed point of $\mathcal{T}_P$ to a wide class of bilattice-based logic programs.

1 Introduction

Since their introduction by Ginsberg [8], bilattices have become a well-known algebraic structure for reasoning about the sort of inconsistencies which arise when one formalizes the process of accumulating knowledge. In particular, Fitting [4, 5, 7] introduced quite general consequence operators for logic programs whose semantics are based on four-valued bilattices, and derived their basic properties.

Annotated languages are an alternative formal tool for handling, for example, the semantics of logic programs over many-valued logics and probabilistic programs, see [10, 13]. Their use, however, gives rise to the obvious question of how annotated logic programming compares with logic programming based on bilattices, see [9–11]. In this paper, we contribute to this discussion by combining the two approaches in that we make use of bilattice structures within the framework of annotated logic programs. Specifically, we introduce bilattice-based annotated logic programs (BAPs) and establish a fixed-point semantics for them. BAPs, being many-valued and quantitative in nature, enable us to work with statistical knowledge and databases which can be incomplete or inconsistent, and thereby introduce monotonic extensions of two-valued negation. But what is particularly

* The authors thank the Boole Centre for Research in Informatics (BCRI) at University College Cork for substantial support in presenting this paper.
** The authors are grateful to the referees for useful suggestions on preliminary versions of this paper.

important here is that they possess many of the desirable properties of classical logic. Thus, their model-theoretic properties can be nicely described by analogy with the model-theoretic properties of classical logic and, if some additional computational rules are introduced into the proof theory, sound and complete proof procedures based on classical binary resolution methods can be developed, although no details of this are presented here. We also show that BAPs are expressive enough to formalize the fixed-point theory of the annotation-free logic programs of Fitting [4] and of implication-based quantitative annotated programs *à la* Van Emden. Moreover, we prove that unlike semantic operators defined within the frameworks of the approaches just mentioned, the properties of the immediate consequence operator ($\mathcal{T}_P$) for BAPs guarantee that all the logical consequences of a given program are computed as its least fixed point. The operator $\mathcal{T}_P$ as it is defined in this paper is continuous and this fact allows us to obtain a generalization of a theorem of Seda [14] used in the context of approximation of $\mathcal{T}_P$ by neural networks. Thus, this paper can be seen as a step towards defining semantic operators suitable for handling probabilistic neural networks which work with conflicting sources of information.

The structure of the paper is as follows. In §2, we describe bilattices, following Ginsberg [8]. In §3, we describe the syntax and semantics of bilattice-based languages, and in particular the syntax and semantics of BAPs. In §4, we develop fixed-point theory and relate it to declarative semantics; we also establish there the approximation theorem mentioned in the previous paragraph. In §5, we compare our approach with the work of others in this area including that of Fitting, Kifer and Subrahmanian and of Van Emden. This allows us to extend the approximation result of §4 to Fitting's bilattice-based (k-existential) logic programs, and to bilattice-based quantitative programs. Finally, we summarize our conclusions in §6; §5 and §6 also contain further motivation for our work.

2 Bilattices

Definition 1. *A* bilattice $\mathbf{B}$ *is a sextuple* $(\mathbf{B}, \wedge, \vee, \oplus, \otimes, \neg)$ *such that* $(\mathbf{B}, \vee, \wedge)$ *and* $(\mathbf{B}, \oplus, \otimes)$ *are both complete lattices, and* $\neg : \mathbf{B} \to \mathbf{B}$ *is a mapping satisfying the following three properties:* $\neg^2 = Id_{\mathbf{B}}$*,* $\neg$ *is a dual lattice homomorphism from* $(\mathbf{B}, \wedge, \vee)$ *to* $(\mathbf{B}, \vee, \wedge)$*, and* $\neg$ *is the identity mapping on* $(\mathbf{B}, \oplus, \otimes)$*.*

Let $L_1 = (\mathcal{L}_1, \leq_1)$ and $L_2 = (\mathcal{L}_2, \leq_2)$ be two lattices, let x_1, x_2 denote arbitrary elements of the lattice L_1, and let y_1, y_2 denote arbitrary elements of the lattice L_2. Let $\cap_1, \cup_1$ denote the join respectively meet defined in the lattice L_1, and let $\cap_2, \cup_2$ denote the join respectively meet defined in the lattice L_2.

Definition 2. *Suppose* $L_1 = (\mathcal{L}_1, \leq_1)$ *and* $L_2 = (\mathcal{L}_2, \leq_2)$ *are complete lattices. Form the set of points* $\mathcal{L}_1 \times \mathcal{L}_2$*, and define the two orderings* $\leq_t$ *and* $\leq_k$ *on* $\mathcal{L}_1 \times \mathcal{L}_2$ *as follows.*

(1) $\langle x_1, y_1\rangle \leq_t \langle x_2, y_2\rangle$ *if and only if* $x_1 \leq_1 x_2$ *and* $y_2 \leq_2 y_1$.
(2) $\langle x_1, y_1\rangle \leq_k \langle x_2, y_2\rangle$ *if and only if* $x_1 \leq_1 x_2$ *and* $y_1 \leq_2 y_2$.

We denote the structure which results by $L_1 \odot L_2 = (\mathcal{L}_1 \times \mathcal{L}_2, \leq_t, \leq_k) = (\mathbf{B}, \leq_t, \leq_k)$*, where* $\mathbf{B}$ *denotes* $\mathcal{L}_1 \times \mathcal{L}_2$.

Having defined $L_1 \odot L_2$, we define bilattice operations on it as follows.

Definition 3. *The four bilattice operations associated with* $\leq_t$ *and* $\leq_k$ *are:* $\langle x_1, y_1\rangle \wedge \langle x_2, y_2\rangle = \langle x_1 \cap_1 x_2, y_1 \cup_2 y_2\rangle$, $\langle x_1, y_1\rangle \vee \langle x_2, y_2\rangle = \langle x_1 \cup_1 x_2, y_1 \cap_2 y_2\rangle$, $\langle x_1, y_1\rangle \otimes \langle x_2, y_2\rangle = \langle x_1 \cap_1 x_2, y_1 \cap_2 y_2\rangle$*, and* $\langle x_1, y_1\rangle \oplus \langle x_2, y_2\rangle = \langle x_1 \cup_1 x_2, y_1 \cup_2 y_2\rangle$.

Proposition 1. *[4, 8] Suppose* $\mathbf{B}$ *is a distributive bilattice. Then there are distributive lattices* L_1 *and* L_2 *such that* $\mathbf{B}$ *is isomorphic to* $L_1 \odot L_2$.

Thus, every distributive bilattice can be represented as a product of two lattices[4]. In the present article, we consider only logic programs over distributive bilattices and therefore the underlying bilattice of any program we consider will always be formed as a product of two lattices.

3 Annotated Logic Programs Based on Bilattice Structures

Let $\mathbf{B} = L_1 \odot L_2$ denote a bilattice given as the product of two complete lattices L_1, L_2 each of which is a sublattice of the lattice $([0, 1], \leq)$, where $[0, 1]$ is the unit interval of real numbers and $\leq$ is the usual linear ordering on it.

3.1 First-Order Languages Based on Bilattice Structures

Let P denote a first order logic program. We proceed next to define the underlying annotated language $\mathcal{L}$ of P as follows.

Definition 4. *The* alphabet *consists of a conventional first-order alphabet together with annotation constants* $(\alpha_1, \beta_1), (\alpha_2, \beta_2), \ldots \in \mathbf{B}$*, annotation variables* $(\tau_1, \sigma_1), (\tau_2, \sigma_2), \ldots$ *taking values in* $\mathbf{B}$*, and total continuous annotation functions* $\vartheta_1, \vartheta_2, \ldots$ *of type* $\mathbf{B}^i \to \mathbf{B}$*. Also, we allow in the language connectives* $\vee, \wedge, \oplus, \otimes$ *(corresponding to join and meet with respect to* $\leq_t$ *and* $\leq_k$*),* $\neg$*,* $\sim$*, and quantifiers* $\forall, \exists, \Pi, \Sigma$ *(corresponding to infinite meet and join with respect to* $\leq_t$ *and* $\leq_k$*).*

Note 1. Annotation constants and variables are taken as pairs of constants respectively pairs of variables reflecting the bilattice-based semantics of $\mathcal{L}$; the first and second elements of a pair accumulate belief for resp. against a fact.

[4] For comparable results concerning interlaced bilattices, see [4, 5].

Note 2. Note that $\sim$ is an alternative to bilattice negation $\neg$: the former is many-valued and is usually called "ontological" negation in comparison with the latter, two-valued, called "epistemic" negation. For careful examination of the properties of both negations see, for example, [9]. In our language $\sim$ is defined as the restriction of $\neg$ to the set $\{\langle 1, 0\rangle, \langle 0, 1\rangle\}$.

We use the word *term* with its conventional meaning in first-order languages.

Definition 5. *An* annotation term *is either a member of* $\mathbf{B}$ *or is an annotation variable or has the form* $\vartheta((\mu_1, \nu_1), \ldots, (\mu_n, \nu_n))$*, where* ϑ *is an annotation function and* $(\mu_1, \nu_1), \ldots, (\mu_n, \nu_n)$ *are annotation terms.*

Definition 6. *We will call a formula of the form* $R(t_1, \ldots, t_n)$ *an* atomic formula *or an* atom *if* R *is an* n*-ary predicate symbol and* $t_1, \ldots, t_n$ *are terms.*

Definition 7. *An* annotated formula *is defined inductively as follows:*

- *If* R *is an* n*-ary predicate symbol,* $t_1, \ldots, t_n$ *are terms, and* (μ, ν) *is an annotation term, then* $R(t_1, \ldots, t_n) : (\mu, \nu)$ *is an* annotated formula *(called an* annotated atom*).*
- *If* F *and* G *are annotated formulae, then so are* $(F \vee G)$*,* $(F \wedge G)$*,* $(F \otimes G)$*,* $(F \oplus G)$*,* $(\neg F)$ *and* $(\sim F)$*.*
- *If* F *and* G *are formulae and* (μ, ν) *is an annotation term, then* $(F \vee G) : (\mu, \nu)$*,* $(F \wedge G) : (\mu, \nu)$*,* $(F \otimes G) : (\mu, \nu)$*,* $(F \oplus G) : (\mu, \nu)$*, are annotated formulae.*
- *If* F *is an annotated formula and* x *is a variable symbol, then* $(\forall x F)$*,* $(\exists x F)$*,* $(\Pi x F)$ *and* $(\Sigma x F)$ *are annotated formulae.*

Definition 8. *The first order annotated language* $\mathcal{L}$ *given by an alphabet consists of the set of all annotated formulae constructed from the symbols of the alphabet. We will refer to* $\mathbf{B}$ *as the* underlying bilattice *of the language* $\mathcal{L}$*.*

3.2 Interpretations

Let $\mathbf{B}$ denote the bilattice underlying an annotated language $\mathcal{L}$; it will provide the set of truth values for $\mathcal{L}$.

We define the notions of "pre-interpretation J with domain D for $\mathcal{L}$" and "variable assignment V with respect to J" in the same way as one defines these notions for conventional first-order languages, see [12] for example.

Definition 9. *An* interpretation I *for* $\mathcal{L}$ *consists of a pre-interpretation* J *for* $\mathcal{L}$ *together with the assignment of a mapping* $|R| = |R|_I : D^n \longrightarrow \mathbf{B}$ *for each* n*-ary predicate symbol* R *in* $\mathcal{L}$*. Furthermore, given a variable assignment* V*, by an abuse of notation which will not cause confusion, we denote by* $|t| = |t|_{J,V}$ *the term assignment in* D *of the term* t *in* $\mathcal{L}$ *with respect to* J *and* V*.*

One further piece of notation we need is as follows: for each element $\langle\alpha,\beta\rangle$ of $\mathbf{B}$, we denote by $\chi_{\langle\alpha,\beta\rangle} : \mathbf{B} \longrightarrow \mathbf{B}$ the mapping defined by $\chi_{\langle\alpha,\beta\rangle}(\langle\alpha',\beta'\rangle) = \langle 1,0\rangle$ if $\langle\alpha,\beta\rangle \leq_k \langle\alpha',\beta'\rangle$ and $\chi_{\langle\alpha,\beta\rangle}(\langle\alpha',\beta'\rangle) = \langle 0,1\rangle$ otherwise.

Definition 10. *Let I be an interpretation with domain D for a first order annotated language $\mathcal{L}$ and let V be a variable assignment. Then an annotated formula F in $\mathcal{L}$ can be given a* truth value *$|F| = |F|_V^I$ in $\mathbf{B}$ as follows.*

- *If F is an annotated atom $R(t_1,\ldots,t_n) : (\mu,\nu)$, then the value of $|F|_V^I$ is given by $|F|_{I,V} = \chi_{\langle\mu,\nu\rangle}(|R|_I(|t_1|,\ldots,|t_n|))$.*
- *$|(\neg F) : (\mu,\nu)|_{I,V} = |F : (\nu,\mu)|_{I,V}$.*
- *$|\sim F : (\mu,\nu)|_{I,V} = \neg^*(|F : (\mu,\nu)|_{I,V})$, where the operation $\neg^*$ denotes the restriction of the bilattice operation $\neg$ to the set of values $\{\langle 1,0\rangle, \langle 0,1\rangle\}$.*
- *If F has the form $(F_1 \otimes F_2)$, where F_1 and F_2 are annotated atoms, then $|F_1 \otimes F_2|_{I,V} = |F_1|_{I,V} \otimes |F_2|_{I,V}$.*
  Note that on the left side of this equation, the symbol $\otimes$ denotes a connective in $\mathcal{L}$, and on the right side it denotes an operation of the bilattice $\mathbf{B}$.
- *If an annotated formula has the form $(F_1 \otimes F_2) : (\mu,\nu)$, then $|(F_1 \otimes F_2) : (\mu,\nu)|_{I,V} = \chi_{\langle\mu,\nu\rangle}(|F_1| \otimes |F_2|)$.*
- *If a formula has the form $\Sigma x R(x) : (\mu,\nu)$, then*

$$|\Sigma x R(x) : (\mu,\nu)|_{I,V} = \chi_{\langle\mu,\nu\rangle}\left(\sum_{d\in D} |R(d)|\right),$$

  where $\sum$ is the infinite join with respect to $\leq_k$, $|R(d)|$ receives interpretation with respect to I and $V(x/d)$, where $V(x/d)$ is V except that x is assigned d.

We omit the definitions for the remaining connectives and quantifiers, and the reader can easily complete Definition 10. We simply mention here that the connectives $\oplus$, $\otimes$, $\vee$, $\wedge$ and the quantifiers Σ, Π, $\exists$, $\forall$ correspond to the finite and infinite operations defined on bilattices as in §2. Notice that if restricted to one lattice in the obvious way, and in particular to the classical lattice {0,1}, the operations $\otimes$ and $\wedge$ are the same, as are $\oplus$ and $\vee$, $\neg$ corresponds to classical negation, and the quantifiers $\exists$ and $\forall$ are the usual ones in classical logic.

In general, the analogs of the classical truth values true and false in our set of truth values are represented by $\langle 1,0\rangle$ and $\langle 0,1\rangle$ - the greatest and least elements of the bilattice with respect to $\leq_t$. Furthermore, the definitions of satisfiable, unsatisfiable, valid and non-valid formulae and of models are standard if the classical truth values true and false in these definitions are replaced by the greatest respectively least elements of the bilattice relative to the $\leq_t$.

Definition 11. *Let I be an interpretation for $\mathcal{L}$ and let F be a closed annotated formula of $\mathcal{L}$. Then I is a* model *for F if $|F|_{I,V} = \langle 1,0\rangle$. We say that I is a model for a set S of annotated formulae if it is a model for each annotated formula of S.*

Definition 12. *Let S be a set of closed annotated formulae and let F be a closed annotated formula of $\mathcal{L}$. We say that F* is a logical consequence of *S if, for every interpretation I of $\mathcal{L}$, I is a model for S implies I is a model for F.*

Proposition 2. *Let S and F be as in the previous definitions. Then F is a logical consequence of S if and only if $S \cup \{\sim F\}$ is unsatisfiable.*

The following propositions, whose proofs use Definition 10 and monotonicity of the bilattice operations relative to $\leq_k$ and are omitted, play an important role when discussing the set of all logical consequences of a set of annotated formulae.

Proposition 3. *Let F be a formula, and fix the value $|F|_{I,V}$. If $|F : (\alpha, \beta)| = \langle 1, 0\rangle$, then $|F : (\alpha', \beta')| = \langle 1, 0\rangle$ for all $\langle \alpha', \beta'\rangle \leq_k \langle \alpha, \beta\rangle$.*

Proposition 4. $|F_1 : (\mu_1, \nu_1) \otimes \ldots \otimes F_k : (\mu_k, \nu_k)| = \langle 1, 0\rangle \iff |F_1 : (\mu_1, \nu_1) \oplus \ldots \oplus F_k : (\mu_k, \nu_k)| = \langle 1, 0\rangle \iff |F_1 : (\mu_1, \nu_1) \wedge \ldots \wedge F_k : (\mu_k, \nu_k)| = \langle 1, 0\rangle \iff$ *each* $|F_i : (\mu_i, \nu_i)| = \langle 1, 0\rangle$, $(i \in \{1, \ldots, k\})$.

Proposition 5. *If $|F_1 : (\mu_1, \nu_1) \odot \ldots \odot F_k : (\mu_k, \nu_k)|_{I,V} = \langle 1, 0\rangle$, then $|(F_1 \odot \ldots \odot F_k) : ((\mu_1, \nu_1) \odot \ldots \odot (\mu_k, \nu_k))|_{I,V} = \langle 1, 0\rangle$, where $\odot$ is any one of the connectives $\otimes, \oplus, \wedge$.*

3.3 Definition of Bilattice-Based Annotated Logic Programs

Definition 13. *An* annotated literal *is an annotated atom or its (epistemic) negation. A* positive annotated literal *is an annotated atom. A* negative annotated literal *is the (epistemic) negation of an annotated atom.*

Definition 14. *A* bilattice-based annotated logic program (BAP) *P consists of a finite set of (annotated)* program clauses *of the form*

$$A : (\mu, \nu) \leftarrow L_1 : (\mu_1, \nu_1), \ldots, L_n : (\mu_n, \nu_n),$$

where $A : (\mu, \nu)$ is an annotated atom called the head *of the clause and $L_1 : (\mu_1, \nu_1), \ldots, L_n : (\mu_n, \nu_n)$ denotes $L_1 : (\mu_1, \nu_1) \otimes \ldots \otimes L_n : (\mu_n, \nu_n)$ and is called the* body *of the clause; each $L_i : (\mu_i, \nu_i)$ is an annotated literal called an* annotated body literal *of the clause. Individual and annotation variables in the body are thought of as being existentially quantified using Σ.*

Each clause of the form $A : (\mu, \nu) \leftarrow L_1 : (\mu_1, \nu_1), \ldots, L_n : (\mu_n, \nu_n)$ can equivalently be seen as a Horn clause

$$\Pi \mu_1, \nu_1, \ldots, \Pi \mu_l, \nu_l (\Pi x_1, \ldots, \Pi x_s A \oplus \sim (\Sigma x_1, \ldots, \Sigma x_s (L_1 \otimes \ldots \otimes L_n))),$$

where A stands for $A : (\mu, \nu)$, and L_i stands for $L_i : (\mu_i, \nu_i)$.

The definitions of unit clause and of program goal are standard, see [12].

We need to mention here that any given logic program contains only a finite set of annotation constants. This is why there are classes of programs which can be interpreted by finite bilattices. For example, logic programs which do not contain annotation variables and/or annotation functions, and logic programs containing only functions which do not generate new annotation constants through the process of computing.

Example 1. Consider the following infinitely-interpreted logic program:

$$R_1(a_1) : (1, 0.5) \leftarrow$$
$$R_2(f(x)) : (\frac{\mu}{2}, \frac{\mu}{3}) \leftarrow R_1(x) : (\mu, \nu)$$
$$R_1(f(x)) : (\frac{\mu}{3}, \frac{\nu}{3}) \leftarrow R_2(x) : (\mu, \nu)$$

This program receives its interpretations from the countable bilattice whose underlying set of elements contains 0, 1, 0.5 and all the numbers which can be generated from 0, 1, 0.5 by the functions $\frac{\mu}{2}$, $\frac{\mu}{3}$, $\frac{\nu}{3}$.

4 Declarative Semantics for BAPs

Throughout this section, we let P denote a bilattice-based annotated logic program with underlying bilattice $\mathbf{B}$ and underlying first order annotated language $\mathcal{L}$.

Definition 15. *A* ground term (atom) *is a term (atom) containing no free individual variables. A* c-annotated atom *is an annotated atom containing no variable annotations. A* strictly ground atom *is an annotated atom containing no free individual variables and no variable annotations.*

Definition 16. *The* Herbrand Universe $U_{\mathcal{L}}$ *for* $\mathcal{L}$ *(or for* P*) is the set of all ground terms which can be formed out of the constants and function symbols appearing in* $\mathcal{L}$*. (In the case that* $\mathcal{L}$ *has no constants, we add some constant, a say, to form ground terms.)*

Definition 17. *The* annotation Herbrand base $B_{\mathcal{L}}$ *(or* B_P*) for* $\mathcal{L}$ *(or for* P*) is the set of all strictly ground atoms formed from predicate symbols of* $\mathcal{L}$ *with ground terms from the Herbrand universe as arguments and constants from* $\mathbf{B}$ *as c-annotations. (In case* $\mathcal{L}$ *has no annotation constants, we add some annotation constant,* $(1, 1)$ *say, to form strictly ground atoms.) Finally, we let* $ground(P)$ *denote the set of all strictly ground instances of clauses of* P.

The *Herbrand pre-interpretation* HJ is the same as in [12].

Definition 18. *A* Herbrand interpretation HI for $\mathcal{L}$ *consists of the Herbrand pre-interpretation* HJ *with domain* $U_{\mathcal{L}}$ *of* $\mathcal{L}$ *together with the following: for each* n*-ary predicate symbol in* $\mathcal{L}$*, the assignment of a mapping from* $U_{\mathcal{L}}^n$ *into* $\mathbf{B}$ *as given in Definition 10.*

Note 3. In common with conventional logic programming, each Herbrand interpretation HI for P can be identified with the subset $\{R(t_1,\ldots,t_k):(\alpha,\beta)\in B_P|R(t_1,\ldots,t_k):(\alpha,\beta)$ receives the value $\langle 1,0\rangle$ with respect to $HI\}$ of B_P it determines, where $R(t_1,\ldots,t_k):(\alpha,\beta)$ denotes a typical element of B_P. In future, this identification will be made without further mention. We let $HI_{P,\mathbf{B}}=2^{B_P}$ denote the set of all Herbrand interpretations for P, and we order $HI_{P,\mathbf{B}}=2^{B_P}$ by subset inclusion.

Proposition 6. *Let S be a set of annotated clauses and suppose S has a model. Then S has an annotation Herbrand model.*

Proof. The proof is straightforward consequence of the Note 3.

Proposition 7. *Let S be a set of annotated clauses. Then S is unsatisfiable if and only if S has no annotation Herbrand models.*

Proof. If S is satisfiable, then, according to Proposition 6, it has an annotation Herbrand model.

We call the intersection of all Herbrand models for P the least annotation Herbrand model and denote it by M_P.

The following theorem is generalization of the well-known theorem of van Emden and Kowalski [16]:

Theorem 1. *Let P be a BAP. Then $M_P=\{A:(\mu,\nu)\mid A:(\mu,\nu)$ is a logical consequence of $P\}$.*

Proof. The following proof is essentially the same as that given in [16].
We have that $A:(\mu,\nu)$ is a logical consequence of $P \iff P\cup\{\sim A:(\mu,\nu)\}$ is unsatisfiable (by Proposition 2) $\iff P\cup\{\sim A:(\mu,\nu)\}$ has no annotation Herbrand models (by Proposition 7) $\iff \sim A:(\mu,\nu)$ receives a value $\langle 0,1\rangle$ with respect to all annotation Herbrand models of $P \iff A:(\mu,\nu)$ receives a value $\langle 1,0\rangle$ with respect to all annotation Herbrand models of $P \iff A:(\mu,\nu)\in M_P$.

We define here two consequence operators, $\widehat{\mathcal{T}_P}$ and $\mathcal{T}_P$, and both compute the logical consequences of logic programs. The operator $\widehat{\mathcal{T}_P}$ does not compute all the logical consequences of P in the general case, and therefore we call it the restricted semantic operator; we will use its properties when discussing the relationship of BAPs to annotation-free and implication-based bilattice logic programs. The semantic operator $\mathcal{T}_P$ is an extended version of $\widehat{\mathcal{T}_P}$. It reflects properties established in Propositions 3, 4, 5 and it follows that $\mathcal{T}_P$ computes all the logical consequences of a given program.

Definition 19. *We define the mapping* $\widehat{\mathcal{T}}_P : HI_{P,\mathbf{B}} \to HI_{P,\mathbf{B}}$ *by* $\widehat{\mathcal{T}}_P(HI) = \{A : (\mu,\nu) \in B_P | A : (\mu,\nu) \leftarrow L_1 : (\mu_1,\nu_1),\ldots,L_n : (\mu_n,\nu_n)$ *is a strictly ground instance of a clause in* P*, and* $\{L_1 : (\mu_1,\nu_1),\ldots,L_n : (\mu_n,\nu_n)\} \subseteq HI\}$.

Definition 20. *We define the mapping* $\mathcal{T}_P : \mathrm{HI}_{P,\mathbf{B}} \to \mathrm{HI}_{P,\mathbf{B}}$ *as follows:* $\mathcal{T}_P(\mathrm{HI})$ *is the set of all* $A : (\mu,\nu) \in B_P$ *such that*

1. *either* $A : (\mu,\nu) \leftarrow L_1 : (\mu_1,\nu_1),\ldots,L_n : (\mu_n,\nu_n)$ *is a strictly ground instance of a clause in* P *and* $\{L_1 : (\mu_1',\nu_1'),\ldots,L_n : (\mu_n',\nu_n')\} \subseteq \mathrm{HI}$*, and for each* (μ_i',ν_i') *one of the following holds:*
   (a) $(\mu_i',\nu_i') \geq_k (\mu_i,\nu_i)$,
   (b) $(\mu_i',\nu_i') \geq_k (\mu_j,\nu_j) \otimes \ldots \otimes (\mu_l,\nu_l)$, $i,j,l \in \{1,\ldots,n\}$*, whenever* $L_j = L_l$;
2. *or there are annotated strictly ground atoms* $A : (\mu_1^*,\nu_1^*),\ldots,A : (\mu_k^*,\nu_k^*) \subseteq \mathrm{HI}$ *such that* $\langle\mu,\nu\rangle \leq_k \langle\mu_1^*,\nu_1^*\rangle \oplus \ldots \oplus \langle\mu_k^*,\nu_k^*\rangle$.

Note 4. It follows from Proposition 3 that whenever $F : (\mu,\nu) \in HI$ and $(\mu',\nu') \leq_k (\mu,\nu)$, then $F : (\mu',\nu') \in HI$. Note that this holds in particular for $\widehat{\mathcal{T}}_P(HI)$ and for $\mathcal{T}_P(HI)$.

Note also that the item $1a$ in Definition 20 is the same as given in Definition 19, and reflects the property stated in Proposition 3; items 1b and 2 extend the Definition of $\widehat{\mathcal{T}_P}$ in order to reflect the results established in Propositions 4 and 5.

The operator $\mathcal{T}_P$ is monotone. This fact follows from the monotonicity of annotation functions and the monotonicity of the connectives with respect to the ordering $\leq_k$. Thus, we define a transfinite sequence as follows.

Definition 21. *We set* $\mathcal{T}_P \uparrow 0 = \emptyset$, $\mathcal{T}_P \uparrow \alpha = \mathcal{T}_P(\mathcal{T}_P \uparrow (\alpha - 1))$*, if* α *is a successor ordinal, and* $\mathcal{T}_P \uparrow \alpha = \sum\{\mathcal{T}_P \uparrow \beta : \beta < \alpha\}$*, if* α *is a limit ordinal.*

The next example displays the difference between $\widehat{\mathcal{T}_P}$ and $\mathcal{T}_P$.

Example 2. Consider the logic program P based on the bilattice $\mathbf{B} = L_1 \odot L_2$, where $L_1 = L_2 = (\{0,\frac{1}{3},\frac{2}{3},1\},\leq)$, with $0 \leq \frac{1}{3} \leq \frac{2}{3} \leq 1$.

$$
\begin{aligned}
R_1(a) &: (0,\frac{2}{3}) \leftarrow \\
R_1(a) &: (\frac{1}{3},\frac{1}{3}) \leftarrow \\
R_3(a) &: (0,0) \leftarrow \\
R_2(x_1) &: (\tau,\sigma) \leftarrow R_1(a) : (\tau,\sigma) \\
R_4(x_2) &: (1,1) \leftarrow R_3(x_2) : (0,\frac{1}{3}), R_3(x_2) : (\frac{1}{3},0)
\end{aligned}
$$

Now consider the computation of the least fixed points of $\widehat{\mathcal{T}_P}$ and of $\mathcal{T}_P$.
$\widehat{\mathcal{T}_P} \uparrow 0 = \emptyset$.

$\widehat{\mathcal{T}_P} \uparrow 1 = \{R_1(a) : (0, \frac{2}{3}), R_1(a) : (\frac{1}{3}, \frac{1}{3}), R_3(a) : (0, 0)\}$.
$\widehat{\mathcal{T}_P} \uparrow 2 = \{R_1(a) : (0, \frac{2}{3}), R_1(a) : (\frac{1}{3}, \frac{1}{3}), R_3(a) : (0, 0), R_2(a) : (0, \frac{2}{3}), R_2(a) : (\frac{1}{3}, \frac{1}{3})\}$, and this is the least fixed point of $\widehat{\mathcal{T}_P}$.

Compare it with the computation of the least fixed point of $\mathcal{T}_P$. After each formula we put in square brackets the number of a rule from the definition of $\mathcal{T}_P$ which enables us to compute that formula.
$\mathcal{T}_P \uparrow 0 = \emptyset$.
$\mathcal{T}_P \uparrow 1 = \{R_1(a) : (0, \frac{2}{3})[1a], R_1(a) : (\frac{1}{3}, \frac{1}{3})[1a], R_3(a) : (0, 0)[1a]\}$.
$\mathcal{T}_P \uparrow 2 = \{R_1(a) : (0, \frac{2}{3})[1a], R_1(a) : (\frac{1}{3}, \frac{1}{3})[1a], R_3(a) : (0, 0)[1a], R_1(a) : (\frac{1}{3}, \frac{2}{3})[2], R_2(a) : (0, \frac{2}{3})[1a], R_2(a) : (\frac{1}{3}, \frac{1}{3})[1a], R_4(a) : (1, 1)[1b]\}$.
$\mathcal{T}_P \uparrow 3 = \{R_1(a) : (0, \frac{2}{3})[1a], R_1(a) : (\frac{1}{3}, \frac{1}{3})[1a], R_3(a) : (0, 0)[1a], R_1(a) : (\frac{1}{3}, \frac{2}{3})[2], R_2(a) : (0, \frac{2}{3})[1a], R_2(a) : (\frac{1}{3}, \frac{1}{3})[1a], R_4(a) : (1, 1)[1b], R_2(a) : (\frac{1}{3}, \frac{2}{3})[2]\}$.

Note that all the formulae marked by 1b and 2 are not contained in the least fixed point of $\widehat{\mathcal{T}_P}$, but if we take into consideration Propositions 3, 4 and 5, we see that these formulae are logical consequences of P.

Example 3. Consider the logic program from Example 1. The least fixed point of this program (for $\widehat{\mathcal{T}_P}$ and for $\mathcal{T}_P$) is
$\{R_1(a_1) : (1, 0.5), R_2(f(a_1)) : (\vartheta_1(1), \vartheta_2(0.5))$,
$R_1(f(f(a_1))) : (\vartheta_3(\vartheta_1(1)), \vartheta_4(\vartheta_2(0.5))), \ldots$,
$R_1(f^{n-1}(a_1)) : (\vartheta_3^n(\vartheta_1^{n-1} \ldots (\ldots((1))\ldots)), \vartheta_4^n(\vartheta_2^{n-1} \ldots (\ldots((0.5))\ldots)))$,
$R_2(f^n(a_1)) : (\mu', \nu'), \ldots\}$, where $n \in \omega$, $\mu' = \vartheta_1^n(\vartheta_3^{n-1}(\vartheta_1^{n-2} \ldots (\ldots((1))\ldots)))$, $\nu' = \vartheta_2^n(\vartheta_4^{n-1}(\vartheta_2^{n-2} \ldots (\ldots((0.5))\ldots)))$, and ϑ_1, ϑ_2, ϑ_3, ϑ_4 stand for the functions $\frac{\mu}{2}$, $\frac{\nu}{3}$, $\frac{\mu}{3}$, $\frac{\nu}{3}$ respectively.

4.1 Approximation of $\mathcal{T}_P$

The following theorem is important and plays a fundamental role when discussing computation of the least fixed points of $\mathcal{T}_P$.

Theorem 2. *The mapping $\mathcal{T}_P$ is continuous.*

Proof. Let X be a directed subset of 2^{B_P}. In order to show that $\mathcal{T}_P$ is continuous, we have to show that $\mathcal{T}_P(\bigcup X) = \bigcup \mathcal{T}_P(X)$. By directedness, it follows that $\{L_1 : (\alpha_1, \beta_1), \ldots, L_n : (\alpha_n, \beta_n)\} \subseteq \bigcup X$ if and only if $\{L_1 : (\alpha_1, \beta_1), \ldots, L_n : (\alpha_n, \beta_n)\} \subseteq HI$, for some $HI \in X$.
Then $A : (\alpha, \beta) \in \mathcal{T}_P(\bigcup X)$
$\iff A : (\alpha, \beta) \leftarrow L_1 : (\alpha_1, \beta_1), \ldots, L_n : (\alpha_n, \beta_n)$ is a strictly ground instance of a clause in P and $\{L_1 : (\alpha_1, \beta_1), \ldots, L_n : (\alpha_n, \beta_n)\} \subseteq \bigcup X$
$\iff A : (\alpha, \beta) \leftarrow L_1 : (\alpha_1, \beta_1), \ldots, L_n : (\alpha_n, \beta_n)$ is a strictly ground instance of a clause in P and $\{L_1 : (\alpha_1, \beta_1), \ldots, L_n : (\beta_n, \alpha_n)\} \subseteq HI$ for some $HI \in X$
$\iff A : (\alpha, \beta) \in \mathcal{T}_P(HI)$, for some $HI \in X$
$\iff A : (\alpha, \beta) \in \bigcup \mathcal{T}_P(X)$.

Proposition 8. *Let* HI *be an annotation Herbrand interpretation for P. Then* HI *is a model for P if and only if $\mathcal{T}_P(\mathrm{HI}) \subseteq \mathrm{HI}$.*

Proof. HI is a model for P if and only if for each strictly ground instance $A : (\mu, \nu) \leftarrow A_1 : (\mu_1, \nu_1), \ldots, A_k : (\mu_k, \nu_k)$ of each clause in P we have $\{A_1 : (\mu_1, \nu_1), \ldots, A_k : (\mu_k, \nu_k)\} \subseteq \mathrm{HI}$ implies $A : (\mu, \nu) \in \mathrm{HI}$ and $A_j : (\mu_1, \nu_1) \circ \ldots \circ A_j : (\mu_n, \nu_n) \in HI$ implies $A_j : ((\mu_1, \nu_1) \circ \ldots \circ (\mu_n, \nu_n)) \in HI$, where $\circ$ is either of $\oplus$ or $\otimes$. But this holds if and only if $\mathcal{T}_P(\mathrm{HI}) \subseteq \mathrm{HI}$.

Now, using Kleene's theorem and Theorem 2, we may assert that $\mathrm{lfp}(\mathcal{T}_P) = \mathcal{T}_P \uparrow \omega$. Indeed, we have the following generalization of a well-known theorem due to van Emden and Kowalski [16].

Theorem 3. *$M_P = lfp(\mathcal{T}_P) = \mathcal{T}_P \uparrow \omega$.*

Proof. $M_P = \mathrm{glb}\{\mathrm{HI} \mid \mathrm{HI}$ is a Herbrand model for $P\} = \mathrm{glb}\{\mathrm{HI} \mid \mathcal{T}_P(\mathrm{HI}) \subseteq \mathrm{HI}\}$ (by Proposition 8)$= \mathrm{lfp}(\mathcal{T}_P)$ (by definition of the least fixed point) $= \mathcal{T}_P \uparrow \omega$ (by Theorem 2 and Kleene's theorem).

Definition 22. *Let P be a BAP and let G be a goal $\leftarrow A_1 : (\mu_1, \nu_1), \ldots A_k : (\mu_k, \nu_k)$. An* answer *for $P \cup \{G\}$ is a substitution $\theta\lambda$ for individual and annotation variables of G. We say that $\theta\lambda$ is a* correct answer *for $P \cup \{G\}$ if $\Pi((A_1 : (\mu_1, \nu_1), \ldots A_k : (\mu_k, \nu_k))\theta\lambda)$ is a logical consequence of P.*

Definition 23. *Let $l : B_P \to \mathbb{N}$ be a level mapping with the property that, for each $n \in \mathbb{N}$, we can effectively find the set of all $A \in B_P$ satisfying $l(A) = n$.*

Definition 24. *[6] Let $HI_{P,\mathbf{B}}$ be the set of all interpretations for P. We define the ultrametric $d : HI_{P,\mathbf{B}} \times HI_{P,\mathbf{B}} \to \mathbb{R}$ as follows: if $HI_1 = HI_2$, we set $d(HI_1, HI_2) = 0$, and if $HI_1 \neq HI_2$, we set $d(HI_1, HI_2) = 2^{-N}$, where N is such that HI_1 and HI_2 differ on some ground atom of level N and agree on all atoms of level less then N.*

The following theorem on approximation of least fixed points is a generalization of the theorem of Seda ([14]) to the case of BAPs.

Theorem 4. *Let P be an arbitrary BAP, let HI denote the least fixed point of $\mathcal{T}_P$ and suppose that we are given $\varepsilon > 0$. Then there exists a finite program $\overline{P} = \overline{P}(\varepsilon)$ (a finite subset of ground(P)) such that $d(\overline{HI}, HI) < \varepsilon$, where $\overline{HI}$ denotes the least fixed point of $\mathcal{T}_{\overline{P}}$.*

Proof. The proof is essentially the same as that given in [14], and uses Definitions 17, 18, 3, 20 and Theorem 2 instead of the corresponding definitions and results used in the proof of [14].

5 Relation to Other Bilattice-Based Logic Programs

In this section, we consider two alternative approaches to bilattice-based logic programming, as follows.

- Fitting's language, see [4] for example, does not contain any annotations, but contains all possible connectives and quantifiers from Definition 4.
- Implication-based logic programs contain annotated arrows instead of having all the literals annotated. These programs may be seen as a generalization of Van Emden's approach [11, 15] to the bilattice-based case.

We consider here the relationship between $\mathcal{T}_P$ and the semantic operators of Fitting and Van Emden. As a result, we extend Theorem 4 on approximation of $\mathcal{T}_P$ to these two kinds of logic programs. The results of this section may be seen as a revision, formalization and further development of some ideas of Kifer and Subrahmanian [10] on the relationship between general annotated programs (GAPs) and the logic programs of Fitting and Van Emden. For example, we extend their results concerning translation of Fitting's connectives into GAPs by allowing Σ in BAPs, and this gives us a translation of the k-existential fragment of Fitting's logic programs into BAPs. We also show that in spite of the assertion in [10], it is not always the case that the least fixed point of the T_P-operator of Van Emden is finite, and this is why we revise the proof of equality of the least fixed points of T_P and of $\widehat{\mathcal{T}_P}$.

Definition 25. *We call the set of clauses of the form $A \leftarrow F$, where all individual variables appearing in F are quantified by Σ, and F is a formula consisting of literals connected by $\vee$, $\wedge$, $\oplus$ and $\otimes$, a* Fitting bilattice-based (k-existential) logic program*. Constant propositional symbols denoting elements of the bilattice are also allowed in the language. Clauses having the form $A \leftarrow$ are thought of as being completed as $A \leftarrow (1, 1)$.*

Definition 26. *We call the set of all clauses of the form*

$$A \leftarrow \Sigma x_1, \ldots, \Sigma x_k (L_1 \otimes \ldots \otimes L_n),$$

where each L_i is a literal and $x_1, \ldots x_k$ are all the individual variables appearing in $L_1 \otimes \ldots \otimes L_n$, the Horn-clause fragment of a Fitting bilattice-based logic program.

All Fitting formulae receive interpretation from the set of elements of the bilattice $\mathbf{B}$, see [4, 5] for further explanations and definitions. We define next the immediate consequence operator for Fitting's logic programs as follows.

Definition 27. *[4]*

$$\Phi_P(I)(A) = \begin{cases} I(B) & \text{if } A \leftarrow B \in ground(P) \\ A & \text{if } A \in \mathbf{B} \end{cases}$$

Definition 28. *We say that the annotated clause* $A : (\tau_1 \otimes \ldots \otimes \tau_n, \sigma_1 \otimes \ldots \otimes \sigma_n) \leftarrow \Sigma x_1, \ldots, \Sigma x_k(L_1 : (\tau_1, \sigma_1), \ldots, L_n : (\tau_n, \sigma_n))$*, where the* (τ_i, σ_i) *are variable annotations, translates the Fitting Horn clause* $A \leftarrow \Sigma x_1, \ldots, \Sigma x_k(L_1 \otimes \ldots \otimes L_n)$*.*

Lemma 1. *A formula* A *receives interpretation* $\langle \alpha, \beta \rangle$ *in the least fixed point of* Φ_P *for a Fitting Horn-clause bilattice program* P *if and only if* $A : (\alpha, \beta) \in \mathrm{lfp}(\widehat{\mathcal{T}_{P^F}})$ *for the BAP* P^F *obtained from* P *by using Definition 28.*

Proof. The proof follows by induction on the number of iterations of T_P and $\widehat{\mathcal{T}_{P^F}}$. Note that in this proof and the next, $\langle \alpha, \beta \rangle$ refers to an element of the bilattice, and (α, β) refers to annotations.

We prove first that if a formula A receives interpretation $\langle \alpha, \beta \rangle$ in the least fixed point of Φ_P for a Fitting horn-clause bilattice program P, then $A : (\alpha, \beta) \in \mathrm{lfp}(\widehat{\mathcal{T}_{P^F}})$ for the BAP P^F obtained from P using Definition 28.

Basis step. Assume $\mathrm{lfp}(\Phi_P) = \Phi_P \uparrow 1$ and $(\Phi_P \uparrow 1)(I)(A) = \langle \alpha, \beta \rangle$. This means $A \leftarrow (\alpha, \beta) \in \mathrm{ground}(P)$. But then there is a clause $A : (\alpha, \beta) \leftarrow \in \mathrm{ground}(P^F)$, and hence $A : (\alpha, \beta) \in \widehat{\mathcal{T}_{P^F}} \uparrow 1$.

Inductive step. Suppose that whenever $(\Phi \uparrow k)(I)(B_i) = \langle \alpha_i, \beta_i \rangle$, for $i = 1, \ldots, k$, then all the $B_i : (\alpha_i, \beta_i) \in \widehat{\mathcal{T}_{P^F}} \uparrow k$.
Consider $\Phi \uparrow (k+1)$. Let $(\Phi \uparrow (k+1))(I)(A) = \langle \alpha, \beta \rangle$. Then $A \leftarrow B_1 \otimes \ldots \otimes B_k \in \mathrm{ground}(P)$ and $(\Phi \uparrow k)(I)(B_1 \otimes \ldots \otimes B_k) = \langle \alpha, \beta \rangle$. For each B_i, $i = 1, \ldots, k$, let $I(B_i) = \langle \alpha_i, \beta_i \rangle$. Then $\langle \alpha, \beta \rangle = (\langle \alpha_1, \beta_1 \rangle \otimes \ldots \otimes \langle \alpha_k, \beta_k \rangle)$ $(*)$.

Since $A \leftarrow B_1 \otimes \ldots \otimes B_k \in \mathrm{ground}(P)$, we have that $A : (\alpha, \beta) \leftarrow B_1 : (\alpha_1, \beta_1), \ldots, B_k : (\alpha_k, \beta_k) \in \mathrm{ground}(P^F)$. Using our induction hypothesis, we have that each $B_i : (\alpha_i, \beta_i) \in \widehat{\mathcal{T}_{P^F}} \uparrow k$. Hence, we obtain $A : (\alpha_1, \beta_1 \otimes \ldots \otimes \alpha_k, \beta_k) \in \widehat{\mathcal{T}_{P^F}} \uparrow (k+1)$ and, using $(*)$, we have that $A : (\alpha, \beta) \in \widehat{\mathcal{T}_{P^F}} \uparrow (k+1)$.

Now we need to prove that if $A : (\alpha, \beta) \in \mathrm{lfp}(\widehat{\mathcal{T}_{P^F}})$ for the BAP P^F obtained from P using Definition 28, then A receives interpretation $\langle \alpha, \beta \rangle$ in the least fixed point of Φ_P for a Fitting horn-clause bilattice program P.

Basis step. Let $\mathrm{lfp}(\widehat{\mathcal{T}_{P^F}})$ be obtained at the first iteration of $\widehat{\mathcal{T}_{P^F}}$, and that $A : (\alpha, \beta) \in \widehat{\mathcal{T}_{P^F}} \uparrow 1$. Then there is a clause $A : (\alpha, \beta) \leftarrow \in \mathrm{ground}(P^F)$. But then, according to the definition of P^F, $A \leftarrow (\alpha, \beta) \in \mathrm{ground}(P)$. Hence, $(\Phi_P \uparrow 1)(I)(A) = \langle \alpha, \beta \rangle$.

Inductive step. Suppose for k we have that if $B_i : (\alpha_i, \beta_i) \in \widehat{\mathcal{T}_{P^F}} \uparrow k$, then $(\Phi \uparrow k)(I)(B_i) = \langle \alpha_i, \beta_i \rangle$, $(i \in \{1 \ldots k\})$. Let $A : (\alpha, \beta) \in \widehat{\mathcal{T}_{P^F}} \uparrow (k+1)$. Then there is a clause $A : (\alpha, \beta) \leftarrow B_1 : (\alpha_1', \beta_1'), \ldots, B_k : (\alpha_k', \beta_k') \in \mathrm{ground}(P^F)$, and each $(\alpha_i', \beta_i') \geq_k (\alpha_i, \beta_i)$, each $B_i : (\alpha_i', \beta_i') \in \widehat{\mathcal{T}_{P^F}} \uparrow k$ and, according to the definition of P^F, $(\alpha, \beta) = (\alpha_1', \beta_1' \otimes \ldots \otimes \alpha_k', \beta_k')$ $(*)$. This clause translates the clause $A \leftarrow B_1 \otimes \ldots \otimes B_k \in \mathrm{ground}(P)$. Now, using our induction hypothesis, for each B_i we have that $(\Phi \uparrow k)(I)(B_i) = \langle \alpha_i', \beta_i' \rangle$. But then, according to the definitions of I and Φ_P, $(\Phi \uparrow (k+1))(I)(A) = \langle \alpha_1', \beta_1' \rangle \otimes \ldots \otimes \langle \alpha_k', \beta_k' \rangle$, and, since we have $(*)$, it follows that $(\Phi \uparrow (k+1))(I)(A) = \langle \alpha, \beta \rangle$.

Definition 29. *We define the following process of translation of Fitting clauses (cf. Definition 25) into annotated clauses.*

1. *If F consists of literals $L_1, \ldots, L_n$, substitute them by annotated literals $L_1 : (\tau_1, \sigma_1), \ldots, L_n : (\tau_n, \sigma_n)$, where each (τ_i, σ_i) is a variable annotation. We call the resulting formula F'.*
2. *Each clause of the form $A \leftarrow F'$ should be transformed into*

$$A : (\vartheta((\tau_1, \ldots, \tau_n), (\sigma_1, \ldots, \sigma_n))) \leftarrow F''$$

   using the following rule: if $A \leftarrow F'$ is

$$A \leftarrow L_1 : (\tau_1, \sigma_1), \ldots, L_k : (\tau_k, \sigma_k) \circ L_{k+1} : (\tau_{k+1}, \sigma_{k+1}), \ldots, L_n : (\tau_n, \sigma_n),$$

   then replace it by $A : (((\tau_1, \sigma_1), \ldots, (\tau_k, \sigma_k)) \circ ((\tau_{k+1}, \sigma_{k+1}), \ldots (\tau_n, \sigma_n))) \leftarrow L_1 : (\tau_1, \sigma_1), \ldots, L_k : (\tau_k, \sigma_k), L_{k+1} : (\tau_{k+1}, \sigma_{k+1}), \ldots, L_n : (\tau_n, \sigma_n)$, where $\circ$ stands for one of $\vee$, $\wedge$ and $\oplus$.

Note that the symbols $\vee$, $\wedge$, $\oplus$ appearing in the bodies of the clauses denote connectives in the language, and the symbols $\vee$, $\wedge$, $\oplus$ appearing in the heads of the clauses denote operations defined on the underlying bilattice. We say that the resulting annotated clause

$$A : (\vartheta((\tau_1, \ldots, \tau_n), (\sigma_1, \ldots, \sigma_n))) \leftarrow F''$$

translates the Fitting (k-existential) clause $A \leftarrow F$.

Theorem 5. *A formula A receives a value $\langle \alpha, \beta \rangle$ in the least fixed point of Φ_P for a Fitting logic program P if and only if $A : (\alpha, \beta) \in \mathrm{lfp}(\widehat{\mathcal{T}_{P^F}})$ for the BAP P^F obtained from P using the rules from Definition 29.*

Proof. The proof is obtained by induction on the number of iterations of the semantic operators and uses Lemma 1 and Definition 29.

Next, we extend Van Emden's implication-based approach to the case when a bilattice instead of the unit interval of reals is taken as the underlying structure of a logic program. We also show that the resulting extended programs can be transformed into BAPs.

Definition 30. *A quantitative logic program, see [15], is a set of definite clauses of the form*

$$A \leftarrow\!\boxed{f}\!- B_1, \ldots, B_n,$$

where f is a factor or threshold taken from the interval $[0, 1]$ of reals. Atoms $B_1, \ldots, B_n$ are thought of as being connected using &.

The atoms A, $B_1, \ldots, B_n$ receive their interpretation from the interval $[0,1]$, and the value of the head A of a clause is computed as $g \times \min(|B_1|, \ldots, |B_n|)$, where $\min(|B_1|, \ldots, |B_n|)$ is the minimum value of the atoms $B_1, \ldots, B_n$.

Definition 31. *An* implication-based bilattice normal logic program P *consists of a finite set of* program clauses *of the form*

$$A \leftarrow\!\boxed{f, g}\!- L_1, \ldots, L_n,$$

where A is an atomic formula called the head *of the clause, the $L_1, \ldots, L_n$ are literals forming the* body *of the clause, and f, g are factors or thresholds taken from the interval $[0,1]$ of reals. The literals $L_1, \ldots, L_n$ are thought of as being connected using $\otimes$.*

All formulae receive their interpretation from the bilattice $\mathbf{B} = L_1 \odot L_2$, where $L_1 = L_2 = ([0,1], \leq_k, \leq_t)$. The value of the head of a clause A is computed as $f \times \min(|L_1|^1, \ldots, |L_n|^1), g \times \min(|L_1|^2, \ldots, |L_n|^2)$, where $\min(|L_1|^i, \ldots, |L_n|^i)$ is the minimum value of the literals $L_1, \ldots, L_n$, and the index $i \in \{1, 2\}$ denotes the first or second elements of a member of the bilattice. Empty bodies are thought of as having interpretation $\langle 1, 1 \rangle$.

Definition 32. *We say that an annotated clause*

$$A : (f \times \min(\tau_1, \ldots \tau_n), g \times \min(\sigma_1, \ldots \sigma_n)) \leftarrow L_1 : (\tau_1, \sigma_1), \ldots, L_n : (\tau_n, \sigma_n),$$

where each τ_i, σ_i is a variable, translates the clause

$$A \leftarrow\!\boxed{f, g}\!- L_1, \ldots, L_n$$

from the implication-based bilattice logic program.

We give here the definition of the immediate consequence operator T'_P for the implication-based bilattice logic programs – it generalizes Van Emden's T_P operator to the bilattice case in the obvious way.

Definition 33. *We define* $T'_P(I)(A) = \{\mathrm{lub}_k \langle (f \times \min\{|L_i|_1\}), (g \times \min\{|L_i|_2\} \rangle |$ $i \in \{1, \ldots, n\}$ *and* $A \leftarrow\!\boxed{f, g}\!- L_1, \ldots, L_n$ *is a variable-free instance in* $P\}$.

The next definition is applicable only to logic programs which can be either finitely interpreted or interpreted by infinite bilattices whose subsets always have lub with respect to the k-ordering.

Definition 34. *We denote by* $\mathrm{lfp}^*(\widehat{\mathcal{T}_P})$ *the set of formulae obtained from the* $lfp(\widehat{\mathcal{T}_P})$ *through the following process: if we have strictly ground formulae* $A : (\alpha_1, \beta_1), A : (\alpha_2, \beta_2), \ldots \in lfp(\widehat{\mathcal{T}_P})$, *then we replace them by* $A : (\alpha, \beta)$, *where* $(\alpha, \beta) = \mathrm{lub}_k(\langle \alpha_1, \beta_1 \rangle, \langle \alpha_2, \beta_2 \rangle, \ldots)$ *in the bilattice* $\mathbf{B}$.

Theorem 6. *A formula A receives the value $\langle \alpha, \beta \rangle$ in the least fixed point of (T'_P) for an implication-based bilattice normal logic program P if and only if $A : (\alpha, \beta) \in \mathrm{lfp}^*(\widehat{\mathcal{T}_{P^{VE}}})$ for the BAP P^{VE} whose clauses are obtained from P as described in Definition 32.*

Sketch of proof The proof proceeds by induction on the number of iterations of T_P and $\widehat{\mathcal{T}_P^{VE}}$ needed to reach the least fixed point. We use the fact that in programs P^{VE}, annotation functions are not contained in the bodies of clauses and, as a consequence, the annotation functions in the heads generate a descending sequence (relative to $\leq_k$) of elements of $\mathbf{B}$ which tends to $\langle 0, 0 \rangle$. Any subset of this bilattice has a lub_k.

Note that the proof of the analogous theorem in [10] uses the argument that $\mathrm{lfp}(T_P)$ is always finite. This is not always the case here. Consider Examples 1 and 3: this annotated logic program can be seen as a translation of Van Emden's bilattice-based logic program into an annotated program, but it can be interpreted only by an infinite bilattice.

Theorems 5 and 6 exhibit classes of bilattice-based logic programs which can be translated into BAPs and, consequently, whose least fixed point can be approximated according to Theorem 4.

There is another interesting consequence of Lemma 1, Theorems 5 and 6. Since all these theorems were proven using the restricted operator $\widehat{\mathcal{T}_P}$ and we showed in §4 that, unlike the extended semantic operator $\mathcal{T}_P$, $\widehat{\mathcal{T}_P}$ does not compute all the logical consequences of P, it is straightforward to show that the Fitting semantic operator and the operator *á la* Van Emden do not compute all the logical consequences of bilattice-based logic programs in the general case.

Example 4. Consider the following simple example of a Horn clause Fitting program P: $B \leftarrow (1, 0)$, $B \leftarrow (0, 1)$, $A \leftarrow B$ and the corresponding implication-based logic program P: $B \leftarrow \boxed{1, 0} -$, $B \leftarrow \boxed{0, 1} -$, $A \leftarrow \boxed{1, 1} - B$.

The least fixed point of Φ_P (and of T'_P) gives us $I(B) = \langle 1, 0 \rangle, I(B) = \langle 0, 1 \rangle, I(A) = \langle 1, 0 \rangle, I(A) = \langle 0, 1 \rangle$. According to Definitions 28 and 32 we have the following translation of this programs into BAP P^F (P^{VE}): $B : (1, 0) \leftarrow$, $B : (0, 1) \leftarrow$, $A : (\tau, \sigma) \leftarrow B : (\tau, \sigma)$.

The least fixed point of $\widehat{\mathcal{T}_{P^F}}$ $(\widehat{\mathcal{T}_{P^{VE}}})$ gives the following set: $\{B : (1, 0), B : (0, 1), A : (1, 0), A : (0, 1)\}$. But the least fixed point of $\mathcal{T}_{P^F}$ is $\{B : (1, 0), B : (0, 1), A : (1, 0), A : (0, 1), B : (1, 1), A : (1, 1)\}$. Clearly, if we consider all the clauses within a logic program as connected using $\otimes$, and if we use the result established in Propositions 4, 5, we obtain $B : (1, 1), A : (1, 1)$ as the logical consequence of P^F (P^{VE}), and $I(A) = \langle 1, 1 \rangle, I(B) = \langle 1, 1 \rangle$ as the logical consequence of P.

This also shows that the general annotated programs of Kifer and Subrahmanian cannot be generalized to the bilattice-based case as it was done in [10] without losing correctness of the computation of the least fixed point of their semantic operator. However, the so-called completion for Fitting logic programs can improve this situation. Each Fitting logic program can be completed if a program is represented as a set of ground instances of clauses and all clauses $A \leftarrow \mathrm{body}_1, \ldots, A \leftarrow \mathrm{body}_n$ having the same head A are replaced by one clause $A \leftarrow \mathrm{body}_1 \oplus \ldots \oplus \mathrm{body}_n$. In this case, Φ_P computes all the logical consequences of the program P and the translation of P into a BAP always gives a program whose semantic operator $\mathcal{T}_P$ never uses the rules captured in items 1b and 2 of Definition 20. This means that for such programs, $\mathcal{T}_P$ and $\widehat{\mathcal{T}_P}$ are the same, and all the results established for $\mathcal{T}_P$ hold for $\widehat{\mathcal{T}_P}$. [5]

6 Conclusions and further work

The main part of our work is devoted to a careful examination of the declarative and fixed-point semantics of bilattice-based annotated logic programs. In particular, we have shown that unlike the usual approach to many-valued logic programming semantics (see, for example, [5], [7], [15], [10] and many others), the immediate consequence operator for bilattice-based annotated logic programs cannot be obtained as a simple extension of the classical semantic operator. We have given some examples displaying the immediate consequence operators as defined in [5], [7], [15], [10], and shown that these operators do not compute all the logical consequences of a program, but only some of them. Finally, we proposed an original definition of the immediate consequence operator computing all the logical consequences of a given bilattice-based annotated logic program, and proved its continuity. A separate section is devoted to theorems displaying how the fixed points of semantic operators defined by Fitting, Van Emden, and Kifer and Subrahmanian can be computed by means of the new semantic operator within the framework of BAPs. We do not consider the issue of implementation here, but note that the close relationship of our work to that of [5], [7], [15], [10] should make this possible.

The declarative semantics for BAPs as it is defined in §4 allows us to propose an SLD-resolution for BAPs and prove its soundness and completeness relative to our semantics. Like the resolution procedures given in [9] for lattice-based logics, this SLD-resolution is enriched with additional rules reflecting the properties of the extended semantic operator for BAPs and is an alternative to the constrained resolution for general annotated logic programs of Kifer and Subrahmanian, see [10].

Theorem 4 on approximation of the least fixed point of the semantic operator for BAPs can be seen as a step towards establishing suitable artificial neural networks for the class of logic programs described in this paper, see [2] and

[5] The same completion given for implication-based logic programs would destroy soundness of T'_P.

[14], for example. The main requirement which is usually made when building such neural network architecture is to preserve its finite properties. This is why bilattice-based annotated logic programs, which provide a continuous semantic operator and an approximation theorem for it, can be seen as a contribution towards creating universal neural networks which can handle probabilistic, incomplete and inconsistent knowledge.

Another field of possible extension of our results is to relate BAPs to probabilistic logic programs, as they were defined and studied, for example, in [1], [3] and other papers.

Finally, it would be interesting to show how we can extend BAPs to logic programs with interval-based annotations, and thereby establish linear programming for them. Such work would relate to [11] and [13] and others and would use many of the results established by these authors.

References

1. F. Bacchus. Lp, a logic for representing and reasoning with statistical knowledge. *Computational Intelligence*, 6:209–231, 1990.
2. A. d'Avila Garsez, K. B. Broda, and D. M. Gabbay. *Neural-Symbolic learning Systems. Foundations and applications*. Springer-Verlag, 2002.
3. R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1,2):78–128, 1990.
4. M. C. Fitting. Bilattices in logic programming. In G. Epstein, editor, *The twentieth International Symposium on Multiple-Valued Logic*, pages 238–246. IEEE, 1990.
5. M. C. Fitting. Bilattices and the semantics of logic programming. *Journal of logic programming*, 11:91–116, 1991.
6. M. C. Fitting. Metric methods: Three examples and a theorem. *The Journal of Logic Programming*, 21:113–127, 1994.
7. M. C. Fitting. Fixpoint semantics for logic programming — A survey. *Theoretical computer science*, 278(1-2):25–51, 2002.
8. M. L. Ginsberg. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.
9. M. Kifer and E. L. Lozinskii. Ri: A logic for reasoning with inconsistency. In *Proceedings of the 4th IEEE Symposium on Logic in Computer Science (LICS)*, pages 253–262, Asilomar, 1989. IEEE Computer Press.
10. M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of logic programming*, 12:335–367, 1991.
11. L. V. S. Lakshmanan and F. Sadri. On a theory of probabilistic deductive databases. *Theory and Practice of Logic Programming*, 1(1):5–42, January 2001.
12. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd edition, 1987.
13. R. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and computation*, 101(2):150–201, 1992.
14. A. K. Seda. On the integration of connectionist and logic-based systems. In *Proceedings of MFCSIT2004*, Electronic Notes in Theoretical Computer Science, Elsevier, 2004. To appear.
15. M. van Emden. Quantitative deduction and fixpoint theory. *Journal of Logic Programming*, 3:37–53, 1986.
16. M. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of the Assoc. for Comp. Mach.*, 23:733–742, 1976.