Biresiduated Multi-Adjoint Logic Programming*

Jesús Medina,[1] Manuel Ojeda-Aciego,[1] A. Valverde,[1] and Peter Vojtáš[2]

[1] Dept. Matemática Aplicada. Universidad de Málaga.
Tel: (+34) 952 132 871. Fax: (+34) 952 132 766
{jmedina,aciego,a_valverde}@ctima.uma.es
[2] Institute of Computer Science, Czech Academy of Sciences.
Tel: (++421) 55 62 209 49. Fax: (++421) 55 62 221 24
vojtas@cs.cas.cz

Abstract. Multi-adjoint logic programs were recently proposed as a generalisation of monotonic and residuated logic programs introduced, in that simultaneous use of several implications in the rules and rather general connectives in the bodies are allowed.
In this work, the possible existence of biresiduated pairs is considered, later, on the resulting framework of biresiduated multi-adjoint logic programming, a procedural semantics is given and a completeness result is proved. As a consequence, alternative hypotheses are obtained for a previous quasi-completeness theorem for multi-adjoint logic programs.

Keywords: Fuzzy logic programming, Biresiduation, Completeness

Topic(s): Models of reasoning (under uncertainty). Foundations of AI and representation of knowledge

* Paper track submission

Biresiduated Multi-Adjoint Logic Programming

Jesús Medina,[1] Manuel Ojeda-Aciego,[1] A. Valverde,[1] and Peter Vojtáš[2]

[1] Dept. Matemática Aplicada. Universidad de Málaga.⋆⋆⋆
{jmedina,aciego,a_valverde}@ctima.uma.es
[2] Institute of Computer Science, Czech Academy of Sciences. †
vojtas@cs.cas.cz

Abstract. Multi-adjoint logic programs were recently proposed as a generalisation of monotonic and residuated logic programs introduced in [3], in that simultaneous use of several implications in the rules and rather general connectives in the bodies are allowed.
In this work, the possible existence of biresiduated pairs is considered, later, on the resulting framework of biresiduated multi-adjoint logic programming, a procedural semantics is given and a completeness result is proved. As a consequence, alternative hypotheses are obtained for a previous quasi-completeness theorem for multi-adjoint logic programs.

1 Introduction

During the past several decades there has been a proliferation of logics, many of which have been motivated by the problem of reasoning in situations where information may be vague or uncertain. Such reasoning has been called inexact or fuzzy or approximate reasoning. Here we propose a lattice-valued logic programming paradigm that we call biresiduated multi-adjoint, which permits the articulation of vague concepts and, moreover, has the property that the truth of an argument can diminish as the number of inferences in it increases.

A comprehensive introduction to the logical foundations of fuzzy reasoning can be found in [4]. A somewhat categorical discussion of inexact reasoning can be seen in [5] where the properties of a generalized conjunction operator links the implication operator to the conjunction operator (via a condition usually referred to as adjointness), and promotes the idea of taking truth values in a lattice.

Multi-adjoint logic programming was introduced as a refinement of both initial work in [11] and residuated logic programming [3]. It allows for very general connectives in the body of the rules, and sufficient conditions for the continuity of its semantics are known. Such an approach is interesting for applications: for instance, in [10] a system is presented where connectives are learnt from different users' examples; one can imagine a scenario in which knowledge is described by a many-valued logic program where connectives can be general aggregation operators (conjunctors, disjunctors, arithmetic mean, weighted sum, ...), even different aggregators for different users and, in

⋆⋆⋆ Partially supported by Spanish DGI project BFM2000-1054-C02-02.
† Partially supported by Czech project GACR 201/00/1489

addition, the program is expected to adequately manage different implications for different purposes.

The special features of multi-adjoint logic programs are: (1) a number of different implications are allowed in the bodies of the rules, (2) sufficient conditions for continuity of its semantics are known, and (3) the requirements on the lattice of truth-values are weaker that those for the residuated approach.

It is important to recall that many different "and" and "or" operations have been proposed for use in fuzzy logic. It is therefore important to select, for each particular application, the operations which are the best for this particular application. Several papers discuss the optimal choice of "and" and "or" operations for fuzzy control, when the main criterion is to get the stablest control. In reasoning application, however, it is more appropriate to select operations which are the best in reflecting human reasoning, i.e., operations which are "the most logical". In this paper, we build on the fact that conjunctors in multi-adjoint logic programs need not be either commutative or associative and, thus, consider the possibility of including a further generalisation of the framework, allowing for *biresiduation*, in the sense of [1]. This way, each conjunctor in our multi-adjoint setting may potentially have two "lateral" residuated implications. Yet another reason for introducing biresiduation is that fuzzy logic in a narrow sense [6] is still an open system and thus, new connectives can and should be introduced. A natural question then arises, whether the basic syntactico-semantical properties are not harmed by this generalisation.

The purpose of this work is to provide a procedural semantics to the paradigm of biresiduated multi-adjoint logic programming. The introduction of reductants will allow us to prove a completeness result with respect to greatest correct answers. This result also allows to obtain alternative hypotheses for a quasi-completeness result, given in [9], without using the *supremum property* on the lattice of truth-values. Due to the limitation of the number of pages, proofs are not included in this paper, the interested reader is referred to [8].

2 A motivating example

In fuzzy logic there is a well developed theory of t-norms, t-conorms and residual implications. The objective of this section is to show some interesting non-standard connectives to motivate the consideration of a more general class of connectives in fuzzy logic. The motivation is the following:

When evaluating the relevance of answers to a given query it is common to use some subjective interpretation of human preferences in a granulated way. This is, fuzzy truth-values usually describe steps in the degree of perception (numerous advocations of this phenomenon have been pointed out by Zadeh). This is connected to the well-known fact that people can only distinguish finitely many degrees of quality (closeness, cheapness, ...) or quantity in control. Thus, in practice, although we use the product t-norm $\&_p(x, y) = x \cdot y$, we are actually working with a piece-wise constant approximation of it. In this generality, it is possible to work with approximations of t-norms and/or conjunctions learnt from data by a neural net like, for instance, those in [10].

If we are looking for a hotel which is close to downtown, with reasonable price and being a new building, then classical fuzzy approaches would assign a user "his" particular interpretation of "close", "reasonable" and "new". As, in practice, we can only recognize finitely many degrees of being close, reasonable, new, then the corresponding fuzzy sets have a stepwise shape. It is just a matter of representation that the outcome is done by means of intervals of granulation and/or indistinguishability. This motivates our lattice-valued approach, namely, the set of truth-values will be considered to be a lattice:

- Generated by a partition of the real unit interval $[0, 1]$.
- With all subintervals of $[0, 1]$.
- With all the probability distributions on $[0, 1]$.

Regarding the use of non-standard connectives, just consider that a variable represented by x can be observed with m different values, then surely we should be working with a regular partition of $[0, 1]$ of m pieces. This means that a given value x should be fitted to this "observation" scale as the least upper bound with the form k/m (analytically, this corresponds to $\big(\lceil m \cdot x \rceil\big)/m$ where $\lceil \_ \rceil$ is the ceiling function). A similar consideration can be applied to both, variable y and the resulting conjunction; furthermore, it might be possible that each variable has different granularity.

Formally, assume in x-axis we have a partition into n pieces, in y-axis into m pieces and in z-axis into k pieces. Then the approximation of the product conjunction looks like

Definition 1. *Denote* $(z)_p = \dfrac{\lceil p \cdot z \rceil}{p}$ *and define, for naturals* $n, m, k > 0$

$$C^k_{n,m}(x, y) = \big((x)_n \cdot (y)_m\big)_k$$

Connectives $C^k_{n,m}(x, y)$ can be non-associative and can be non-commutative, as the following example shows:

Example 1.

1. For instance $C = C^{10}_{10,10}$ is not associative

$$C(0.7, C(0.7, 0.3)) = C(0.7, (0.21)_{10}) = C(0.7, 0.3) = (0.21)_{10} = 0.3$$
$$C(C(0.7, 0.7), 0.3) = C((0.49)_{10}, 0.3) = C(0.5, 0.3) = (0.15)_{10} = 0.2$$

2. $C^4_{10,5}(x, y)$ is not commutative.

$$C^4_{10,5}(0.82, 0.79) = ((0.82)_{10} \cdot (0.79)_5)_4 = (0.9 \cdot 0.8)_4 = (0.72)_4 = 0.75$$
$$C^4_{10,5}(0.79, 0.82) = ((0.79)_{10} \cdot (0.82)_5)_4 = (0.8 \cdot 1)_4 = 1$$

As previously stated, to model precision and granularity, it is reasonable to work with partitions of $[0, 1]$. In fact, in this case, the set of truth values is a finite linearly ordered set. In practical applications it happens that we change the perspective and work with finer and/or coarser partition. This is a special case studied in domain theory [2], in which one of the most fundamental questions is about the representation of a real number: a common approach to this problem is to identify each real number r with a collection of intervals whose intersection is $\{r\}$.

3 Preliminary definitions

The preliminary concepts required to formally define the syntax of biresiduated multi-adjoint logic programs are built on those of the "monoresiduated" multi-adjoint case [9]; to make this paper as self-contained as possible, the necessary definitions about multi-adjoint structures are included below.

Definition 2. *Let $\langle L, \preceq\rangle$ be a complete lattice. A* multi-adjoint lattice *$\mathcal{L}$ is a tuple $(L, \preceq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n)$ satisfying the following items:*

1. *$\langle L, \preceq\rangle$ is bounded, i.e. it has bottom and top elements;*
2. *$\top \,\&_i\, \vartheta = \vartheta \,\&_i\, \top = \vartheta$ for all $\vartheta \in L$ for $i = 1, \ldots, n$;*
3. *$(\leftarrow_i, \&_i)$ is an adjoint pair in $\langle L, \preceq\rangle$ for $i = 1, \ldots, n$; i.e.*
   (a) *Operation $\&_i$ is increasing in both arguments,*
   (b) *Operation $\leftarrow_i$ is increasing in the first argument and decreasing in the second argument,*
   (c) *For any $x, y, z \in P$, we have that $x \preceq (y \leftarrow_i z)$ holds if and only if $(x \,\&_i\, z) \preceq y$ holds.*

The existence of multiple pairs satisfying property 3 (when $n > 1$) in the previous definition justifies the term *multi-adjoint*. Note that residuated lattices are a special case of multi-adjoint lattice, in which the underlying poset has the structure of complete lattice which has monoidal structure wrt $\&$ and $\top$, and only one adjoint pair is present.

In this paper, we will stress on the fact that we are working with a non-commutative $\&$ operator; now, the roles of the left hand side and the right hand side of $\&$ can be different and they could lead two different implications through the adjoint property. The biresiduated structure is obtained by allowing, for each adjoint conjunctor, two "sided" adjoint implications, as detailed in the following definition.

Definition 3. *Let $\langle L, \preceq\rangle$ be a complete lattice. A* biresiduated multi-adjoint lattice *$\mathcal{L}$ is a tuple $(L, \preceq, \swarrow^1, \nwarrow_1, \&_1, \ldots, \swarrow^n, \nwarrow_n, \&_n)$ satisfying the following items:*

1. *$\langle L, \preceq\rangle$ is bounded, i.e. it has bottom and top elements;*
2. *$\top \,\&_i\, \vartheta = \vartheta \,\&_i\, \top = \vartheta$ for all $\vartheta \in L$ for $i = 1, \ldots, n$;*
3. *$(\swarrow^i, \nwarrow_i, \&_i)$ satisfies the following properties, for all $i = 1, \ldots, n$; i.e.*
   (a) *Operation $\&_i$ is increasing in both arguments,*
   (b) *Operations $\swarrow^i, \nwarrow_i$ are increasing in the first argument and decreasing in the second argument,*
   (c) *For any $x, y, z \in P$, we have that*

$$x \preceq y \swarrow^i z \quad \text{if and only if} \quad x \,\&_i\, z \preceq y$$
$$x \preceq y \nwarrow_i z \quad \text{if and only if} \quad z \,\&_i\, x \preceq y$$

It is the last condition (3c), which makes this algebraic structure suitable for being used in a logical context, for it can be interpreted as a multiple-valued *modus ponens*-like inference rule. Actually, property (3c) in the definition guarantees soundness of our computational model with respect to this rule, as we will see later.

From the point of view of expressiveness, it is interesting to allow extra operators to be involved with the operators in the biresiduated multi-adjoint lattice. The formal construction makes use of the constructions and terminology of universal algebra, such as Ω-algebra over a signature (or graded set) Ω, in order to define formally the syntax and the semantics of the languages we will deal with. Anyway, for the sake of simplicity in the presentation, we will rephrase all the needed concepts in more standard terms.

The structure which allows the possibility of using additional operators is that of a *biresiduated multi-adjoint Ω-algebra* which can be understood as an extension of a biresiduated multi-adjoint lattice containing a number of extra operators, which are required to be monotonic in each argument.

We will be working with two Ω-algebras: the first one, $\mathfrak{F}$, to define the syntax of our programs, and the second one, $\mathfrak{L}$, to host the manipulation of the truth-values of the formulas in the programs. To avoid possible name-clashes, we will denote the interpretation of an operator symbol ω in Ω under $\mathfrak{L}$ as $\dot{\omega}$ (a dot on the operator), whereas ω itself will denote its interpretation under $\mathfrak{F}$.

Definition 4. *A* biresiduated multi-adjoint logic program *(in short* a program*) on an Ω-algebra $\mathfrak{F}$ with values in a lattice $\mathfrak{L}$ is a set $\mathbb{P}$ of rules of the form $\langle A \swarrow^i \mathcal{B}, \vartheta\rangle$ or $\langle A \nwarrow_i \mathcal{B}, \vartheta\rangle$ such that:*

1. *The* head *of the rule, A, is a propositional symbol of Π;*
2. *The* body *formula, $\mathcal{B}$, is a formula of $\mathfrak{F}$ built from propositional symbols $B_1, \ldots, B_n$ ($n \geq 0$) which contains no implication symbol;*
3. *The* confidence factor *ϑ is an element (a truth-value) of L.*

As usual, facts *are rules with body $\top$, and a* query *(or* goal*) is a propositional symbol intended as a question $?A$ prompting the system.*

As usual, an *interpretation* is a mapping $I\colon \Pi \to L$. Note that each of these interpretations can be uniquely extended to the whole set of formulas, $\hat{I}\colon F_\Omega \to L$. The set of all interpretations of the formulas defined by the Ω-algebra $\mathfrak{F}$ in the Ω-algebra $\mathfrak{L}$ is denoted $\mathcal{I}_\mathfrak{L}$.

The ordering $\preceq$ of the truth-values L can be easily extended to $\mathcal{I}_\mathfrak{L}$, which also inherits the structure of complete lattice. The minimum element of the lattice $\mathcal{I}_\mathfrak{L}$, which assigns $\bot$ to any propositional symbol, will be denoted $\triangle$.

A rule of a biresiduated multi-adjoint logic program is satisfied whenever its truth-value is greater or equal than the confidence factor associated with the rule. Formally:

Definition 5.

1. *An interpretation $I \in \mathcal{I}_\mathfrak{L}$* satisfies *$\langle A \swarrow^i \mathcal{B}, \vartheta\rangle$ if and only if $\vartheta \,\dot{\&}_i\, \hat{I}\,(\mathcal{B}) \preceq I\,(A)$.*
2. *An interpretation $I \in \mathcal{I}_\mathfrak{L}$* satisfies *$\langle A \nwarrow_i \mathcal{B}, \vartheta\rangle$ if and only if $\hat{I}\,(\mathcal{B}) \,\dot{\&}_i\, \vartheta \preceq I\,(A)$.*
3. *An interpretation $I \in \mathcal{I}_\mathfrak{L}$ is a* model *of a program $\mathbb{P}$ iff all weighted rules in $\mathbb{P}$ are satisfied by I.*
4. *An element $\lambda \in L$ is a* correct answer *for a query $?A$ and a program $\mathbb{P}$ if for any interpretation $I \in \mathcal{I}_\mathfrak{L}$ which is a model of $\mathbb{P}$ we have $\lambda \preceq I(A)$.*

Note that, for instance, $\bot$ is always a correct answer for any query and program.

The immediate consequences operator, given by van Emden and Kowalski, can be easily generalised to the framework of biresiduated multi-adjoint logic programs.

Definition 6. *Let $\mathbb{P}$ be a program, the* immediate consequences operator $T_{\mathbb{P}}$ *maps interpretations to interpretations, and for an interpretation I and propositional variable A is defined by*

$$T_{\mathbb{P}}(I)(A) = \sup\left\{\{\vartheta \mathbin{\dot{\&}_i} \hat{I}(\mathcal{B}) \mid \langle A \swarrow^i \mathcal{B}, \vartheta\rangle \in \mathbb{P}\} \cup \{\hat{I}(\mathcal{B}) \mathbin{\dot{\&}_i} \vartheta \mid \langle A \nwarrow_i \mathcal{B}, \vartheta\rangle \in \mathbb{P}\}\right\}$$

As usual, the semantics of a biresiduated multi-adjoint logic program is characterised by the post-fixpoints of $T_{\mathbb{P}}$; that is, an interpretation I of $\mathcal{I}_{\mathfrak{L}}$ is a model of a program $\mathbb{P}$ iff $T_{\mathbb{P}}(I) \sqsubseteq I$. The proof simply follows from the adjunction property, and it is remarkable that the result is still true even without any further assumptions on conjunctors (definitely they need not be commutative and associative). In this generality, it is possible to work with approximations of t-norms and/or conjunctions learnt from data by a neural net like, for instance, those in [10].

4 Procedural semantics of biresiduated logic programs

It can be shown that the $T_{\mathbb{P}}$ operator is continuous under very general hypotheses (the proof of this fact can be found in [8]), therefore the least model can be reached in at most countably many iterations. Now, it is worth to define a procedural semantics which allows us to actually construct the answer to a query against a given program.

In the following, we will be working in a hybrid Ω-algebra made up from the elements of the lattice and propositional symbols as basic elements, and operators in a subset $\Omega' \subset \Omega$ obtained by removing all the implication symbols.

For the formal description of the computational model, we will consider an extended language $\mathfrak{F}'$ defined by the signature Ω' whose carrier is included in the disjoint union $\Pi \cup L$; this way we can work syntactically with propositional symbols and with the truth-values they represent.

Definition 7. *Let $\mathbb{P}$ be a program on an Ω-algebra $\mathfrak{L}$ with carrier L and V the set of truth values of the rules in $\mathbb{P}$. The* extended language $\mathfrak{F}'$ *is the corresponding Ω'-algebra of formulas freely generated from the disjoint union of Π and V.*

We will refer to the formulas in the language $\mathfrak{F}'$ simply as *extended formulas.*

Our computational model will take a query (an atom), and will provide a lower bound of the value of A under any model of the program. Intuitively, the computation proceeds by, somehow, substituting propositional symbols by lower bounds of their truth-value until, eventually, an extended formula with no propositional symbol is obtained, which will be interpreted in the lattice to get the computed answer.

Given a program $\mathbb{P}$, we define the following admissible rules for transforming any extended formula.

Definition 8. Admissible rules *are defined as follows:*

R1a *Substitute an atom A in an extended formula by $(\vartheta \,\&_i\, \mathcal{B})$ whenever there exists a rule $\langle A \swarrow^i \mathcal{B}, \vartheta\rangle$ in $\mathbb{P}$.*

R1b *Substitute an atom A in an extended formula by $(\mathcal{B} \,\&_i\, \vartheta)$ whenever there exists a rule $\langle A \nwarrow_i \mathcal{B}, \vartheta\rangle$ in $\mathbb{P}$.*

R2 *Substitute an atom A in an extended formula by $\bot$.*

R3 *Substitute an atom A in an extended formula by ϑ whenever there exists a fact $\langle A \swarrow^i \top, \vartheta\rangle$ or $\langle A \nwarrow_i \top, \vartheta\rangle$ in $\mathbb{P}$.*

Note that if an extended formula turns out to have no propositional symbols, then it can be directly interpreted in the as an element in $\mathfrak{L}$, rather than like a formula. This justifies the following definition of *computed answer*.

Definition 9. *Let $\mathbb{P}$ be a program in a language interpreted on a lattice $\mathcal{L}$ and let $?A$ be a goal. An element $\lambda \in L$ is said to be a* computed answer *if there is a sequence $G_0, \ldots, G_{n+1}$ such that*

1. *$G_0 = A$ and $G_{n+1} = @(r_1, \ldots, r_m)$ where[1] $r_i \in L$ for all $i = 1, \ldots m$, and $\lambda = \dot{@}(r_1, \ldots, r_m)$.*
2. *Every G_i, for $i = 1, \ldots, n$, is a formula in $\mathfrak{F}'$.*
3. *Every G_{i+1} is inferred from G_i by exactly one of the admissible rules.*

Note that our procedural semantics, instead of being refutation-based (this is not possible, since negation is not allowed in our approach), is oriented to obtaining a bound of the optimal correct answer of the query.

5 Greatest answers. Reductants

The definition of correct answer is not entirely satisfactory in that $\bot$ is always a correct answer. Actually, we should be interested in the greatest confidence factor we can assume on the query, consistently with the information in the program, instead of in the set of its lower bounds. Therefore we will stress on the *greatest correct answer*, Λ_A, for a program $\mathbb{P}$ and a query $?A$.

The following theorem states that the greatest correct answer is reached by the least fix-point of the $T_{\mathbb{P}}$ operator.

Theorem 1. *Given a complete lattice L, a program $\mathbb{P}$ and a propositional symbol A, we have that $T^{\omega}_{\mathbb{P}}(\triangle)(A)$ is the greatest correct answer, Λ_A.*

The proof follows from the facts that the least fix-point is also the least model of a program; the second states a characterisation of correct answers in terms of the $T_{\mathbb{P}}$ operator, that is, $\lambda \in L$ is a correct answer for a program $\mathbb{P}$ and a query $?A$ iff $\lambda \preceq T^{\omega}_{\mathbb{P}}(\triangle)(A)$.

Regarding the computation of the greatest correct answer, it might well be the case that for some lattices, our procedural semantics cannot compute the greatest correct answer, as in the following example adapted from [7]:

[1] Here the r_i represent all the variables occurring in the G_{n+1}, $@$ represents the composition, as functions in the lattice, of all the operators inserted by rules R1a and R1b and $\dot{@}$ is the interpretation of $@$ as operator in $\mathcal{L}$.

Example 2. Consider L to be the powerset of a two-element set $\{a, b\}$ ordered by inclusion, and a program $\mathbb{P}$ with rules $\langle A \leftarrow B, a\rangle$ and $\langle A \leftarrow B, b\rangle$ and fact $\langle B \leftarrow \top, \top\rangle$. Assuming that the adjoint conjunction to $\leftarrow$ has the usual boundary conditions, then the greatest correct answer to the query $?A$ is $\top$, since it has to be an upper bound of all the models of the program, therefore it has to be greater than both a and b. But the only computed answers are either a or b.

We can cope with this problem by generalising the concept of reductant [7], that is, whenever we have a finite number of rules $\langle A \leftarrow_{j_i} \mathcal{D}_i, \vartheta_i\rangle$ for $i = 1, \ldots, k$, then there should exist another rule which allows us to get the greatest possible value of A under that set of rules.

Any rule $\langle A \leftarrow_{j_i} \mathcal{D}_i, \vartheta_i\rangle$ contributes a value of the form either $\vartheta_i \mathbin{\dot{\&}_i} b_i$ or $b_i \mathbin{\dot{\&}_i} \vartheta_i$ in the calculation of the lower bound for the truth-value of A, thus we would like to have the possibility of reaching the supremum of all the contributions, in the computational model, in a single step. This leads to the following definition.

Definition 10. *Let $\mathbb{P}$ be a program; assume that the set of rules in $\mathbb{P}$ with head A can be written as $\langle A \swarrow^{i_j} \mathcal{B}_j, \vartheta_j\rangle$ for $j = 1, \ldots, n$, and $\langle A \nwarrow_{k_l} \mathcal{C}_l, \theta_l\rangle$ for $l = 1, \ldots, m$, and contains at least a proper rule; a* reductant for A *is any rule*

$$\langle A \swarrow @(\mathcal{B}_1, \ldots, \mathcal{B}_n, \mathcal{C}_1, \ldots, \mathcal{C}_m), \top\rangle$$

where $\swarrow$ is any implication symbol and the operator $@$ is defined as

$$\dot{@}(b_1, \ldots, b_n, c_1, \ldots, c_m) = \sup\{\vartheta_1 \&_{i_1} b_1, \ldots, \vartheta_n \&_{i_n} b_n,\ c_1 \&_{k_1} \theta_1, \ldots, c_m \&_{k_m} \theta_m\}$$

If there were just facts with head A, but no proper rule, then the expression above does not give a well-formed formula. In this case, the reductant is defined to be a fact which aggregates all the knowledge about A, that is,

$$\langle A \swarrow \top, \sup\{\vartheta_1, \ldots, \vartheta_n\}\rangle$$

As a consequence of the definition, and the boundary conditions in the definition of biresiduated multi-adjoint lattice, the choice of the implication to represent the corresponding reductant is irrelevant for the computational model. Therefore, in the following, we will assume that our language has a distinguished implication to be selected in the construction of reductants, leading to the so-called *canonical reductants*.

It is immediate to prove that the rule constructed in the definition above, in presence of proper rules, behaves as a reductant (in the standard sense) for A in $\mathbb{P}$, in that it provides in a single step the greatest amount of information about A which can be obtained from the rules (and facts) with head A.

It will be interesting to consider only programs which contain all its reductants, but this might be a too heavy condition on our programs; the following proposition shows that it is not true, therefore we can assume that a program contains all its reductants, since its set of models is not modified.

Proposition 1. *Any reductant for a program $\mathbb{P}$ is satisfied by any model of $\mathbb{P}$.*

Note that we have followed just traditional techniques of logic programming, and discarded non-determinism by using reductants.

6 Completeness results

The objective of this section is to introduce a completeness theorem with respect to the greatest correct answers which allows for an alternative proof of a general biresiduated version of Theorem 2 below, originally introduced in [9]:

Theorem 2. *Assume L has the supremum property,*[2] *then for every correct answer $\lambda \in L$ for a program $\mathbb{P}$ and a query $?A$, and arbitrary $\varepsilon \prec \lambda$ there exists a computed answer δ such that $\varepsilon \prec \delta$.*

The main idea here is considering the possibility of obtaining an "optimal" answer for a query to a program. Obviously, the greatest computed answer (if any) would be considered as the *optimal answer*, and a procedure for computing it (assuming it exists) is described in the following proposition.

Proposition 2. *Given a program $\mathbb{P}$ and a query $?A$, construct a computation sequence by using the following procedure:*

1. *Apply R2 only on atoms B for which there are neither rules nor facts with head B.*
2. *Use the canonical reductant, otherwise.*

Provided the procedure terminates, then the resulting computed answer is indeed the optimal answer λ_A.

This proposition is interesting both from a practical and from a theoretical point of view. For the former, it gives a procedure for computing the optimal answer for a query to a program; for the latter, it is the key to prove that, under the assumption of continuity of the fix-point semantics and the termination of the procedure, the least fix-point is indeed the optimal answer which, in turn, is the greatest computed answer.

Theorem 3. *If the procedure in Proposition 2 terminates for query $?A$ and $\mathbb{P}$, then there exists a greatest computed answer which equals the greatest correct answer. In other words, $\lambda_A = T^{\omega}_{\mathbb{P}}(\triangle)(A) = \Lambda_A$.*

The theorem above is obviously interesting on its own, for it shows that the minimal model for a program $\mathbb{P}$ on each propositional symbol A is the greatest computed answer for query $?A$ and program $\mathbb{P}$. Furthermore, as a consequence of this result, we can obtain Theorem 2 without assuming the supremum property.

7 Conclusions and future work

We have presented a procedural semantics for the general theory of biresiduated multi-adjoint logic programming, and a completeness result w.r.t. the greatest correct answers.

Although the central topics of this paper are mainly at the theoretical level, a number of applications are envisaged for the obtained results, for instance the integration

[2] A lattice L is said to satisfy the *supremum property* if for all directed set $X \subset L$ and for all ε we have that if $\varepsilon < \sup X$ then there exists $\delta \in X$ such that $\varepsilon < \delta \leq \sup X$.

of information retrieval and database systems, in this context, a by-product of our results is the possibility of defining a fuzzy relational algebra and a fuzzy Datalog, by the completeness theorem then one would have that the computational power of the fuzzy relational algebra is the same that the expressive power of fuzzy Datalog; another research line which could benefit from these results is multiple-valued resolution for which it is not possible to deal with Horn clauses and refutation, mainly due to the fact that $A \wedge \neg A$ can have strictly positive truth-value, but also to the fact that material implication (the truth value function of $\neg A \vee B$) has not commutative adjoint conjunctor. As our approach does not require adjoint conjunctors to be commutative, it would allow the development of a sound and complete graded resolution.

As future work, from the theoretical side, we will investigate termination conditions for the given procedure and express them in terms of supremum-like properties of the underlying truth-values lattice; it is expected to link our approach with the interesting approach taken in [3]; from the not-so-theoretical side, a practical evaluation of the proposed approach has to be performed, to evaluate possible optimisation techniques.

References

1. C.J. van Alten. Representable biresiduated lattices. *Journal of Algebra*, 247:672–691, 2002.
2. K. Ciesielski, R. Flagg, and R. Kopperman. Polish spaces, computable approximations, and bitopological spaces. *Topology and applications*, 119(3):241–256, 2002.
3. C.V. Damásio and L. Moniz Pereira. Monotonic and residuated logic programs. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01*, pages 748–759. Lect. Notes in Artificial Intelligence, 2143, 2001.
4. D. Dubois, J. Lang, and H. Prade. Fuzzy sets in approximate reasoning, part 2: Logical approaches. *Fuzzy Sets and Systems*, 40:203–244, 1991.
5. J. Goguen. The logic of inexact concepts. *Synthese*, 19:325–373, 1969.
6. P. Hájek. *Metamathematics of Fuzzy Logic*. Trends in Logic. Kluwer Academic, 1998.
7. M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. of Logic Programming*, 12:335–367, 1992.
8. J. Medina, M. Ojeda-Aciego, A. Valverde, and P. Vojtáš. Biresiduated multi-adjoint logic programming. Technical Report MA-02-01, Dept. Matemática Aplicada. Univ. Málaga, 2002. Available at `http://www.satd.uma.es/aciego/TR/bires-tr.pdf`.
9. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A procedural semantics for multi-adjoint logic programming. In *Progress in Artificial Intelligence, EPIA'01*, pages 290–297. Lect. Notes in Artificial Intelligence 2258, 2001.
10. E. Naito, J. Ozawa, I. Hayashi, and N. Wakami. A proposal of a fuzzy connective with learning function. In P. Bosc and J. Kaczprzyk, editors, *Fuzziness Database Management Systems*, pages 345–364. Physica Verlag, 1995.
11. P. Vojtáš and L. Paulík. Soundness and completeness of non-classical extended SLD-resolution. In *Extensions of Logic Programming, ELP'96*, pages 289–301. Lect. Notes in Comp. Sci. 1050, 1996.