Declarative and Operational Semantics for Bilattice-based Annotated Logic Programs

Ekaterina Komendantskaya
Department of Mathematics, UCC, Cork, Ireland
e.komendantskaya@mars.ucc.ie

Anthony Seda
Department of Mathematics, UCC, Cork, Ireland
a.seda@ucc.ie

Abstract

We introduce the class of normal bilattice-based annotated first-order logic programs (BAPs) and develop declarative and operational semantics for them. Thus, we discuss properties of the associated immediate consequence operators and establish their fixed-point theory. In addition, SLD-resolution for these programs is defined and its soundness and completeness established.
Keywords: Annotated logic programming, bilattices, declarative and operational semantics, semantic operators, fixed points, SLD-resolution.

1 Introduction

Since their introduction by Ginsberg, bilattices have become a well-known algebraic structure for reasoning about the sort of inconsistencies which arise when one formalizes the process of accumulating information from different sources.

We introduce here a declarative and operational semantics for bilattice-based annotated logic programs (BAPs), which can briefly be described as first-order logic programs interpreted by arbitrary (possibly infinite) distributive bilattices.

We obtain a continuous semantic operator for BAPs which computes the least Herbrand models for BAPs and this gives us declarative (fixed-point) semantics.

Further, we establish sound and complete SLD-resolution for BAPs. As far as we know, this is the first sound and complete proof procedure for first-order infinitely interpreted (bi)lattice-based annotated logic programs. Compare, for example, our results with those obtained for constrained resolution for GAPs, which was shown to be incomplete, see [2], or with sound and complete (SLD)-resolutions for finitely-interpreted annotated logic programs (these logic programs do not contain annotation variables and annotation functions), see, for example,[1].

2 Bilattice-Based Logic Programming

We define an annotated bilattice-based language $\mathcal{L}$ to consist of individual variables, constants, functions and predicate symbols together with annotation terms which can consist of variables, constants and/or functions over a bilattice. We allow six connectives and two quantifiers, as follows: $\oplus, \otimes, \vee, \wedge, \neg, \sim, \Sigma, \Pi$.

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*). Annotated atoms can be combined to form complex formulae using the connectives and quantifiers.

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 $A : (\mu, \nu)$ denotes 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.

In [3], we showed how the remaining connectives $\oplus, \vee, \wedge$ can be introduced into BAPs. Let D, v, and J denote respectively a domain of (pre-)interpretation, a variable assignment and a pre-interpretation for a given language, see [4]. An interpretation I for $\mathcal{L}$ consists of J together with the following mappings. The first mapping $\mathcal{I}$ assigns $|R|_{\mathcal{I},v} : D^n \longrightarrow \mathbf{B}$ to each n-ary predicate symbol R in $\mathcal{L}$. Further, for each element $\langle \alpha, \beta \rangle$ of $\mathbf{B}$, we define a mapping $\chi_{\langle \alpha, \beta \rangle} : \mathbf{B} \longrightarrow \mathbf{B}$, where $\chi_{\langle \alpha, \beta \rangle}(\langle \alpha', \beta' \rangle) = \langle 1, 0 \rangle$ 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. The mapping χ is used to evaluate annotated formulae. Thus, if F is an annotated atom $R(t_1, \ldots, t_n) : (\mu, \nu)$, then the value of F is given by $I(F) = \chi_{\langle \mu, \nu \rangle}(|R|_{\mathcal{I},v}(|t_1|_v, \ldots, |t_n|_v))$. Furthermore, using χ we can proceed to give interpretation to complex annotated formulae in the standard way, see [3]. All the connectives of the language are put into correspondence with bilattice operations, and in particular quantifiers correspond to infinite bilattice operations. We call the composition of the two mappings $\mathcal{I}$ and χ an *interpretation* for the bilattice-based annotated language $\mathcal{L}$ and for simplicity of notation denote it by I.

We introduce a semantic operator $\mathcal{T}_P$ for BAPs, prove its continuity and show that it computes at its least fixed point the least Herbrand model for a given BAP. Indeed, we define $\mathcal{T}_P$ next.

Definition 2.1. *We define the mapping* $\mathcal{T}_P : \mathrm{HI}_{P,\mathbf{B}} \rightarrow \mathrm{HI}_{P,\mathbf{B}}$ *as follows:* $\mathcal{T}_P(\mathrm{HI})$ *denotes the set of all* $A : (\mu, \nu) \in B_P$ *such that either*

1. *There is a strictly ground instance of a clause* $A : (\mu, \nu) \leftarrow L_1 : (\mu_1, \nu_1), \ldots, L_n : (\mu_n, \nu_n)$ *in* P *such that there exist annotations* $(\mu'_1, \nu'_1), \ldots, (\mu'_n, \nu'_n)$ *satisfying* $\{L_1 : (\mu'_1, \nu'_1), \ldots, L_n : (\mu'_n, \nu'_n)\} \subseteq \mathrm{HI}$*, and one of the following conditions holds for each* (μ'_i, ν'_i)*:*
   (a) $(\mu'_i, \nu'_i) \geq_k (\mu_i, \nu_i)$,
   (b) $(\mu'_i, \nu'_i) \geq_k \oplus_{j \in J_i}(\mu_j, \nu_j)$*, where* J_i *is the finite set of those indices such that* $L_j = L_i$

or

2. *there are annotated strictly ground atoms* $A : (\mu_1^*, \nu_1^*), \ldots, A : (\mu_k^*, \nu_k^*) \in$ HI *such that* $\langle \mu, \nu \rangle \leq_k \langle \mu_1^*, \nu_1^* \rangle \oplus \ldots \oplus \langle \mu_k^*, \nu_k^* \rangle$.[1]

3 SLD-Resolution for BAPs

We propose a sound and complete proof procedure for BAPs.Like the resolution procedures given in [1] for lattice-based logics, the SLD-resolution for BAPs is enriched with additional rules reflecting the properties of the extended semantic operator for BAPs, and is an alternative to the constrained resolution for the general annotated logic programs of Kifer and Subrahmanian, see [2] and to resolutions for logics which are interpreted by linearly ordered sets and/or finite sets [2].

We adopt the following terminology. 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 3.1 (SLD-derivation). *Let* G_i *be the annotated goal* $\leftarrow A_1 : (\mu_1, \nu_1), \ldots, A_k : (\mu_k, \nu_k)$*, and let* $C, C_1^*, \ldots, C_l^*$ *be the annotated clauses* $A : (\mu, \nu) \leftarrow B_1 : (\mu_1', \nu_1'), \ldots, B_q : (\mu_q', \nu_q')$*,* $A_1^* : (\mu_1^*, \nu_1^*) \leftarrow body_1^*, \ldots, A_l^* : (\mu_l^*, \nu_l^*) \leftarrow body_l^*$*. Then the set of goals* $G_{i+1}^1, \ldots, G_{i+1}^m$ *is* derived *from* G_i *and* C *(and* $C_1^*, \ldots, C_l^*$*) using mgu*[2] $\theta\lambda$ *if the following conditions hold.*

1. $A_m : (\mu_m, \nu_m)$ *is an annotated atom, called the selected atom, in* G*.*
2. θ *is an* mgu *of* A_m *and* A*, and one of the following conditions holds:*
   (a) λ *is an* mgu *of* (μ_m, ν_m) *and* (μ, ν)*;*
   (b) $(\mu_m, \nu_m)\lambda$ *and* $(\mu, \nu)\lambda$ *are constants and* $(\mu, \nu)\lambda \geq_k (\mu_m, \nu_m)\lambda$*;*
   (c) *there are clauses* $C_1^*, \ldots, C_l^*$ *of the form* $A_1^* : (\mu_1^*, \nu_1^*) \leftarrow body_1^*, \ldots, A_l^* : (\mu_l^*, \nu_l^*) \leftarrow body_l^*$ *in* P*, such that* θ *is an* mgu *of* A*,* A_m *and* $A_1^*, \ldots, A_l^*$*,* λ *is an* mgu *of* (μ_m, ν_m)*,* (μ, ν) *and* $(\mu_1^*, \nu_1^*), \ldots, (\mu_l^*, \nu_l^*)$ *or* $(\mu_m, \nu_m)\lambda$*,* $(\mu, \nu)\lambda$ *and* $(\mu^*, \nu^*)\lambda, \ldots, (\mu_l^*, \nu_l^*)\lambda$ *are constants such that* $(\mu_m, \nu_m)\lambda \leq_k ((\mu, \nu)\lambda \oplus (\mu_1^*, \nu_1^*)\lambda \oplus \ldots \oplus (\mu_l^*, \nu_l^*)\lambda)$*.*
3. *in case* 2(*a*), 2(*b*), $G_{i+1} = (\leftarrow A_1 : (\mu_1, \nu_1), \ldots, A_{m-1} : (\mu_{m-1}, \nu_{m-1}), B_1 : (\mu_1', \nu_1'), \ldots, B_q : (\mu_q', \nu_q'), A_{m+1} : (\mu_{m+1}, \nu_{m+1}), \ldots, A_k : (\mu_k, \nu_k))\theta\lambda$*.*
4. *in case* 2(*c*), $G_{i+1} = (\leftarrow A_1 : (\mu_1, \nu_1), \ldots, A_{m-1} : (\mu_{m-1}, \nu_{m-1}), B_1 : (\mu_1', \nu_1'), \ldots, B_q : (\mu_q', \nu_q'), body_1^*, \ldots, body_l^*, A_{m+1} : (\mu_{m+1}, \nu_{m+1}), \ldots, A_k : (\mu_k, \nu_k))\theta\lambda$*.*
   In this case, G_{i+1} *is said to be derived from* G_i*,* C *and* $C_1^*, \ldots, C_l^*$ *using* $\theta\lambda$*.*

[1] Note that whenever $F : (\mu, \nu) \in HI$ and $(\mu', \nu') \leq_k (\mu, \nu)$, then $F : (\mu', \nu') \in HI$. Also, for each formula F, $F : (0, 0) \in HI$.

[2] Throughout this section, *mgu* stands for "most general unifier".

5. *The goals* $G^1_{i+1}, \ldots, G^m_{i+1}$ *can be obtained using the following rules: in case there are atomic formulae* $F_i : (\mu_i, \nu_i), F_{i+1} : (\mu_{i+1}, \nu_{i+1}), \ldots, F_j : (\mu_j, \nu_j)$ *in* G_i *such that* $F_i\theta = F_{i+1}\theta = \ldots = F_j\theta$*, form the next goal* $G^1_{i+1} = F_i\theta : ((\mu_i, \nu_i) \otimes (\mu_{i+1}, \nu_{i+1})), \ldots, F_j : (\mu_j, \nu_j)$*, then* $G^2_{i+1} = F_i : (\mu_i, \nu_i), F_i\theta : ((\mu_{i+1}, \nu_{i+1}) \otimes (\mu_{i+2}, \nu_{i+2})), \ldots, F_j : (\mu_j, \nu_j)$ *and so on for all possible combinations of these replacements. Form the set of goals* $G^1_{i+1}, \ldots, G^m_{i+1}$*, which is always finite and can be effectively enumerated by, for example, enumerating goals according to their leftmost replacements and then according to the number of replacements.*

6. *Whenever a goal* G^i_j *contains a formula of the form* $F : (0, 0)$*, then remove* $F : (0, 0)$ *from the goal and form the next goal* G^i_{j+1}*.*

Definition 3.2. *Suppose that* P *is a BAP and* G_0 *is a goal. An* SLD-derivation *of* $P \cup \{G_0\}$ *consists of a sequence* $G_0, G^i_1, G^j_2 \ldots$ *of BAP goals, a sequence* $C_1, C_2, \ldots$ *of BAP clauses and a sequence* $\theta_1\lambda_1, \theta_2\lambda_2, \ldots$ *of* mgu*s such that each* G^k_{i+1} *is derived from* G^j_i *and* C_{i+1} *using* $\theta_{i+1}\lambda_{i+1}$*.*

An SLD-refutation *of* $P \cup \{G_0\}$ *is a finite SLD-derivation of* $P \cup \{G\}$ *which has the empty clause* $\Box$ *as the last goal of the derivation. If* $G^i_n = \Box$*, we say that the refutation has length* n*.*

The *success set* of P is the set of all $A : (\mu, \nu) \in B_P$ such that $P \cup \{\sim A\}$ has an SLD-refutation.

Theorem 3.1 (Soundness and completeness of SLD-resolution). *The success set of* P *is equal to its least annotation Herbrand model. Alternatively, soundness and completeness can be stated as follows. Every computed answer for* $P \cup \{G\}$ *is a correct answer for* $P \cup \{G\}$*, and for every correct answer* $\theta\lambda$ *for* $P \cup \{G\}$*, there exist a computed answer* $\theta^*\lambda^*$ *for* $P \cup \{G\}$ *and substitutions* φ*,* ψ *such that* $\theta = \theta^*\varphi$ *and* $\lambda = \lambda^*\psi$*.*

References

[1] 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.

[2] M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of logic programming*, 12:335–367, 1991.

[3] E. Komendantskaya, A. K. Seda, and V. Komendantsky. On approximation of the semantic operators determined by bilattice-based logic programs. In *Proceedings of the Seventh International Workshop on First-Order Theorem Proving (FTP'05)*, pages 112–130, Koblenz, Germany, September 15–17 2005.

[4] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 2nd edition, 1987.