# A many-sorted semantics for many-valued annotated logic programs

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

**Abstract**

We show how many-valued annotated logic programs can be semantically and sintactically translated into many-sorted.

**Keywords:** Logic programs, bilattices, many-valued annotated logics, many-sorted logics.

## 1 Introduction

We construct a uniform algorithm which translates bilattice-based annotated logic programs (BAPs) into conventional many-sorted logic programs in the spirit of Manzano, see [4]. In fact, this algorithm can be uniformly extended to other well-known many-valued logic programs, for example logic progarms which are due to Fitting, Van Emden, Kifer and Lozinskii, Kifer and Subrahmanian, Lu, Murray and Rosental, and many others. The translation of many-valued logic programs into many-sorted sheds a light into semantical and sintactical properties of many-valued programs and allows to apply conventional SLD-resolution to them.

## 2 Bilattice-based annotated logic programs

We refer to [2] for a detailed description of BAPs. Here we give just a brief description. We pick the structure of bilattices as an interpretation for a first-order language. And define annotation terms of a form $(\mu, \nu)$ (and possibly $\vartheta^n((\mu_1, \nu_1), \ldots, (\mu_n, \nu_n))$) to run over bilattices. Then an *annotated formula* is defined inductively as follows: if $R$ is an $n$-ary predicate symbol, $t_1, \ldots, t_n$ are terms, and $(\mu, \nu)$ 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 connectives and quantifiers defined on bilattices.

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

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

*Research was partially funded by BCRI (UCC, Cork).

where $A : (\mu, \nu)$ denotes an annotated atom called the *head* of the clause, and 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 $\Sigma$.

## 3 Sorted logic programs MSLP$^*$

We refer to the paper of Manzano [4] for a detailed exposition of many-sorted logics. The abbreviation MSL was introduced in [4] to denote a many-sorted language of a signature $\Sigma$.

We use a fragment of original MSL, and will denote it as MSL$^*$ assuming a new signature $\Sigma^*$ as follows.

**Definition 1.** We define a *signature* $\Sigma^* = \langle \mathrm{SORT}, \mathrm{RANK} \rangle$, where $\mathrm{SORT}(\Sigma^*) = \mathrm{SORT} = \{0, 1, < 0, \overbrace{1, \ldots, 1}^{n} >, 3\}$ (for some $n \in \mathbb{N}$)(representing boolean, individual universes, a universe of $n$-ary relations on individuals and a universe of bilattice symbols).

We define $S_\omega(\mathrm{SORT})$ to be the set of all finite sequences of elements of SORT. RANK is a function whose values are in $[S_\omega(\mathrm{SORT})]$. We denote $\mathtt{Dom}(\mathrm{RANK})$ as $\mathrm{OPER.SYM}(\Sigma^*) = \mathrm{OPER.SYM}$ and call its elements operation symbols. In particular, $\neg, \vee, \wedge, \supset, =, \leq_k$[1] are in OPER.SYM and $\mathrm{RANK}(\neg) = \langle 0, 0 \rangle$, $\mathrm{RANK}(\vee) = \mathrm{RANK}(\wedge) = \mathrm{RANK}(\supset) = \langle 0, 0, 0 \rangle$; $\mathrm{RANK}(\doteq) = \mathrm{RANK}(\leq_k) = \langle 0, 3, 3 \rangle$. We define

$\mathrm{RANK}(f^n(x_1, \ldots, x_n)) = \langle 1, \overbrace{1, \ldots, 1}^{n} \rangle$, where $f^n$ is n-ary function over individual terms;

$\mathrm{RANK}(\vartheta^n((\mu_1, \nu_1), \ldots, (\mu_n, \nu_n)) = \langle 3, \overbrace{3, \ldots, 3}^{n} \rangle$, where $\vartheta^n$ is n-ary function over annotation terms;

$\mathrm{RANK}(R^n(x_1, \ldots, x_n)) = \langle 0, \overbrace{1, \ldots, 1}^{n} \rangle$, where $R^n$ is n-ary relation over individual terms. Additionally, $\mathrm{RANK}(I^n(x_1, \ldots, x_n)) = \langle 3, \overbrace{1, \ldots, 1}^{n} \rangle$, where $I^n$ is a function which assigns a bilattice value to $n$-tuple of individual terms. And, finally, for membership relations $\varepsilon_n$,

$$\mathrm{RANK}(\varepsilon_n) = \langle 0, \overbrace{1, \ldots, 1}^{n}, < 0, \overbrace{1, \ldots, 1}^{n} > \rangle.$$

**Definition 2.** We define a many-sorted *structure*

$$S = \langle A_1, A_2, A_3, f^{A_1}, f^{A_2}, f^{A_3} \rangle,$$

where $A_1$, $A_2$ and $A_3$ are universes for variables of sorts $< 1 >$, $< 0, \overbrace{1, \ldots, 1}^{n} >$ and $< 3 >$, $f^{A_1} \subseteq A_1^n$, $f^{A_2} \subseteq A_2^n$ and $f^{A_3} \subseteq A_3^n$.

[1]In our setting, $\doteq$ denotes equality, that is, $(x \doteq y) = \mathtt{T} \iff x = y$. The symbol $\leq_k$ denotes the relation $\leq$ with respect to $k$-ordering in a given bilattice.

We define a many-sorted formal language $\mathcal{L}$ to consist of symbols from OPER.SYM, quantifier $\exists$ and the set of variables $\mathcal{V} = V_i : i \in \mathrm{SORT} - \{0\}$.

Note that in our setting $\varepsilon x_1, \ldots, x_n X^n$ replaces the usual $X^n(x_1, \ldots, x_n)$, and expressions of the former type are formulae, but expressions of the latter type are not.[2]

We define many-sorted annotated logic programs MSLP$^*$ to consist of Horn clauses formed out of formulae of MSL$^*$ and written in a style of Lloyd [3].

## 3.1 Syntactical translation

We define a translation from BAPs into MSLP$^*$ in two stages: first we show the translation for single formulae and then for the whole clauses.

**Definition 3.** For atomic annotated formulae we define
$\mathrm{TRANSL}_{BAP \mapsto \mathrm{MSL}^*}(R(x_1, \ldots, x_n) : (\mu, \nu)) =$
$\forall I \forall x_1, \ldots, x_n \forall(\mu', \nu')((\varepsilon x_1, \ldots, x_n R^n \supset (I^n(x_1, \ldots x_n) \doteq (\mu', \nu')) \supset ((\mu, \nu) \leq_k (\mu', \nu')))$.

We will abbreviate $\mathrm{TRANSL}_{LAL \mapsto \mathrm{MSL}^*}$ as TRANSL$^*$.

Further, complex annotated formulae receive the following translation:

$$\mathrm{TRANSL}^*(F_1 \otimes F_2) = \mathrm{TRANSL}^*(F_1) \wedge \mathrm{TRANSL}^*(F_2).$$

$$\mathrm{TRANSL}^*(\Sigma x F) = \exists x \mathrm{TRANSL}^*(F).$$

Then the following holds:

**Lemma 1.** Let $F$ be an annotated formula of bilattice-based annotated language (BAL). Let $\Sigma^*$ and $A$ be a signature respectively a structure of MSL$^*$, and $||_I$ be an interpretation for the BAL, then the following holds:

$$|F|_I = \langle 1, 0 \rangle \text{ in BAL} \iff A(\mathrm{TRANSL}^*(F)) = T \text{ in MSL}^* .$$

The following translation can be given for bilattice-based annotated clauses. For each clause of the form $R^n(x_1, \ldots, x_n) : (\mu, \nu) \leftarrow L_1 : (\mu_1, \nu_1) \otimes \ldots \otimes L_n : (\mu_n, \nu_n)$ we obtain the following set of many-sorted clauses: $(\mu, \nu) \leq_k (\mu', \nu') \leftarrow \varepsilon x_1, \ldots, x_n R^n \wedge I(x_1, \ldots, x_n) \doteq (\mu', \nu') \wedge \mathrm{TRANSL}^{**}(L_1 : (\mu_1, \nu_1) \otimes \ldots \otimes L_n : (\mu_n, \nu_n))$;
$I(x_1, \ldots, x_n) \doteq (\mu', \nu') \leftarrow \varepsilon x_1, \ldots, x_n R^n \wedge \mathrm{TRANSL}^{**}(L_1 : (\mu_1, \nu_1) \otimes \ldots \otimes L_n : (\mu_n, \nu_n))$;
$\varepsilon x_1, \ldots, x_n R^n \leftarrow \mathrm{TRANSL}^{**}(L_1 : (\mu_1, \nu_1) \otimes \ldots \otimes L_n : (\mu_n, \nu_n))$. Further, TRANSL$^{**}$ acts on the connective $\otimes$ precisely as TRANSL$^*$ does. For each annotated literal $L_i : (\mu_i, \nu_i) = R_i^m(y_1, \ldots, y_m) : (\mu_i, \nu_i)$ the translation function TRANSL$^{**}$ gives the following result: $\mathrm{TRANSL}^{**}(R_i^m(y_1, \ldots, y_m) : (\mu_i, \nu_i)) = \varepsilon y_1, \ldots, y_m R_i^m \wedge (I_i(y_1, \ldots, y_m) \doteq (\mu_i', \nu_i')) \wedge ((\mu_i, \nu_i) \leq (\mu_i', \nu_i'))$.

Thus, each BAP $P$ can be translated into a many-sorted logic program TRANSL$^{**}(P)$. This program, in its turn, should be enriched with certain axioms reflecting properties of $\doteq$ and $\leq_k$[3] and we will denote this enriched program as $\mathcal{P}^{**}$.

[2] Keeping this convention in mind, we will abuse the notation when using predicates $\doteq$ and $\leq_k$ in order to make syntactical translation clearer.

[3] The axioms are $\forall(x \doteq x \leftarrow)$; $\forall(f(x_1, \ldots, x_y) \doteq f(y_1, \ldots, y_n) \leftarrow (x_1 \doteq y_1) \wedge \ldots \wedge (x_1 \doteq y_1))$; $\forall(x \leq_k x) \leftarrow$; $\forall(x \not\leq y) \leftarrow (y \leq_k x) \wedge (x \neq y)$, $\forall((x \leq_k z) \leftarrow (x \leq_k y) \wedge (y \leq_k z))$; $\forall((x \odot y) \leq_k (x' \odot y') \leftarrow (x \leq_k x') \wedge (y \leq_k y'))$, where $\odot$ stands for any of bilattice operations $\oplus, \otimes$; $\forall(x \leq_k (y \oplus z) \leftarrow (x \leq_k y))$; $\forall((x \otimes y) \leq_k z \leftarrow (x \leq_k z))$; $\forall(\langle 0, 0 \rangle \leq_k x \leftarrow)$.

Using Lemma 1 we can prove the following:

**Theorem 2.** For any bilattice-based annotated logic program $P$ and any annotated formula $F$, $(P \models F) \Rightarrow (\mathcal{P}^{**} \models_S \text{TRANSL}^{**}(F))$.

The Theorem 2 shows that BAPs (both languages and structures) can be fully translated into conventional many-sorted logic programs. It is remarkable that BAPs are translated into a fragment of many-sorted logic programs which include conventional second-order logic programs, see [4]. In particular, functions $I^n$ are variables and this fact makes possible unifications of these functions when applying method of resolution. And we briefly discuss it in the next section.

## 4 Two SLD-resolutions

In [1] we introduced a sound and complete SLD-resolution for BAPs. This resolution is essentially conventional SLD-resolution but reformulated and enriched with several additional rules in order to reflect some properties of bilattices. These extra rules make the resolution sound and complete, but at the same time they make it rather bulky. On the contrary, many-sorted translation $\mathcal{P}^{**}$ for each BAP $P$ allows us to apply conventional SLD-resolution and this may make computations clearer. Bilattice properties are reflected in additional axioms built into each $\mathcal{P}^{**}$, and although the axioms can be used in process of refutation, they do not change the very mechanisms of unification and refutation. The only complication in refutations for $\mathcal{P}^{*}*$ comparing with conventional SLD-resolution given in [3] is that function and predicate symbols get involved in the process of unification, and this happens because they all thought of as being quantified.

**Theorem 3.** For any bilattice-based annotated logic program $P$ and goal $G$ the following property holds: if $P \cup \{\leftarrow G\}$ has a refutation, than $\mathcal{P}^{**} \cup \text{TRANSL}^{**}(\leftarrow G)$ has a refutation.

## References

[1] E. Komendantskaya and A. K. Seda. Sound and complete sld-resolution for bilattice-based annotated logic programs. In *Proceedings of the International Conference INFORMATION-MFCSIT'06*, Cork, Ireland, August 1 – 5, 2006.

[2] 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.

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

[4] M. Manzano. Introduction to many-sorted logic. In K. Meinke and J. V. Tucker, editors, *Many-Sorted logic and its Applications.* John Wiley and Sons, UK, 1993.