Many-valued Logic Programming and Fixpoint Semantics for Higher-order Herbrand Models

Zoran Majkić

University of Maryland, College Park, USA
zoran@cs.umd.edu
http://www.cs.umd.edu/ $\sim$ zoran/

Abstract. In this paper we compare the two versions of knowledge invariant transformations of the original Many-valued logic programs: the strict Annotated logic programs and the 'meta' logic programs obtained by the ontological encapsulation [1]. We show that the first one has the higher-order Herbrand interpretations, while the last can be seen as the flattening of the first one. These two knowledge invariant 2-valued logic transformations of the 4-valued Belnap's bilattice-based logic, able to handle incompleteness and inconsistency of knowledge-base systems, are mutually inverse in Galois connection based on predicate compression and decompression (flattening). Consequently, we can use this Galois connection between them to establish their fixpoint semantics relationship. This results generalize the truth-knowledge fixpoint semantics for many-valued logic programming [2].

1 Introduction

So far, research in many-valued logic programming has proceeded along different directions: *Signed* logics [3, 4] and *Annotated* logic programming [5–7] which can be embedded into the first, *Bilattice-based* logics, [8, 9], and *Quantitative rule-sets*, [10, 11]. Earlier studies of these approaches quickly identified various distinctions between these frameworks. For example, one of the key insights behind bilattices was the interplay between the truth values assigned to sentences and the (non classic) notion of *implication* in the language under considerations. Thus, rules (implications) had weights (or truth values) associated with them as a whole. The problem was to study how truth values should be propagated "across" implications. Annotated logics, on the other hand, appeared to associate truth values with each component of an implication rather than the implication as a whole. Roughly, based on the way in which uncertainty is associated with facts and rules of a program, these frameworks can be classified into *implication based* (IB) and *annotation based* (AB).
In the IB approach a rule is of the form $A \leftarrow^{\alpha} B_1, .., B_n$, which says that the certainty associated with the implication is α. Computationally, given an assignment I of logical values to the B_is, the logical value of A is computed by taking the "conjunction" of logical values $I(B_i)$ and then somehow "propagating" it to the rule head A.
That is the standard case of a many-valued Logic Programming, as for example, the fuzzy or Bilattice-based logic where to each ground atom we can assign some value in the interval $[0, 1]$ or of the particular Bilattice respectively.
In the AB approach a rule is of the form $A : f(\beta_1, .., \beta_n) \leftarrow B_1 : \beta_1, ..., B_n : \beta_n$, which asserts "the certainty of the atom A is least (or is in) $f(\beta_1, .., \beta_n)$, whenever the certainty of the atom B_i is at least (or is in) β_i, $1 \leq i \leq n$", where f is an n-ary computable function and β_i is either constant or a variable ranging over many-valued logic values. The comparison in [12] shows:
1- while the way implication is treated on the AB approach is closer to the classical logic, the way rules are fired in the IB approach has definite intuitive appeal.
2- the AB approach is strictly more expressive than IB. The down side is that query processing in the AB approach is more complicated, e.g. the fixpoint operator is not continuous in general, while it is in the IB approaches.
3- the Fitting fixpoint semantics for logic programs, based exclusively on a bilattice-algebra operators, suffer two drawbacks: the lack of the notion of tautology (bilattice negation operator is an *epistemic* negation) leads to difficulties in defining proof procedures and to the need for additional complex truth-related

notions as ”formula closure”; there is an unpleasant asymmetry in the semantics of implication (which is strictly 2-valued) w.r.t. all other bilattice operators (which produce *any* truth value).
From the points above, it is believed that IB approach is easier to use and is more amenable for efficient implementations, but also annotated syntax (but with IB semantics) is useful to overcome two drawbacks above.
The drawbacks of the Fitting fixpoint semantics for logic programs are are definitely resolved by the introduction [1] of the intuitionistic semantic for a many-valued implication (relative pseudocomplement) transforming a Bilattice-based logic programs into the kind of the intuitionistic logic programs where the implication can be used in a body of programs also. An autoepistemic version of such intuitionistic bilattice-based logic programs is defined in [13] as the simple belief-revision solution for the management of the mutually inconsistent information in Data Integration Systems.
In what follows we will consider the possible embedding of a bilattice-based logic into the 2-valued logic:
1. In [7] it is shown how the Fitting's 3-valued bilattice logic can be embedded into an Annotated Logic Programming which is computationally very complex. In what follows we will use the syntactic annotation of the many-valued logic into the 2-valued logic, as shown above, where a rule is of the form $A : f(\beta_1, .., \beta_n) \leftarrow B_1 : \beta_1, ..., B_n : \beta_n$, asserts ”the certainty of the atom A is least (or is in) $f(\beta_1, .., \beta_n) = \beta_1 \wedge ... \wedge \beta_n$ is the result of the many-valued logic conjunction of logic values $\beta_i \in \mathcal{B}$ of a bilattice $\mathcal{B}$.
As we will see, such 2-valued embedding of a many-valued logic programs generates the higher-order Herbrand interpretations.
2. The ontological embedding [1] into the syntax of new encapsulated many-valued logic (in some sense ’meta’-logic for a many-valued bilattice logic) will be 2-valued and can be seen as the flattening of the many valued logic, where the many-valued logic value $\beta \in \mathcal{B}$ of an original ground atom $r(c_1, .., c_k)$ is deposited into the logic attribute of a new predicate r_F, obtained by an extension of the old predicate r, so that we obtain the ’flattened’ 2-valued ground atom $r_F(c_1, .., c_k, \beta)$.
These two *knowledge invariant* 2-valued logic transformations of the original many-valued logics are mutually inverse: we can consider annotations as contexts for the original atoms of the logic theory. Such context sensitive applications, with higher-order Herbrand models, can be transformed (that is, *flattened*) to logic theories with basic (ordinary) Herbrand interpretations, by enlarging the original predicates with new attributes which characterize the properties of the context: in this way the context becomes part of the language of the logic theory, that is, becomes visible.
The inverse of flattening is the predicate compression: for instance, there are numerous applications where we have to deal with the *database compressions*, that is, with a kind of database transformation where some number of attributes of its relations are *hidden*(compressed) but *not eliminated* as in the case of ordinary projections over relations: we still consider all inferential effects of all information contained in a database, but with respect to the reduced number of ’visible’ attributes. The hidden attributes can be seen as transformed from variables into parameters: in our case, from a logic variable into the logic annotation.
In this paper we will consider only the compression of the logic attribute of the flattened predicates: obtained compressed predicates are identical to predicates from the original many-valued logic program, but the value for their ground atoms is not value of a bilattice $\mathcal{B}$ but the *function* (higher-order value type) in $2^{\mathcal{B}}$.
Let H_F be the Herbrand base of an ontologically encapsulated Logic Program P^A and $Pred(H_F)$ be the set of all predicate symbols. The compression of a predicate $r_F(\mathbf{x}, \alpha), r_F \in Pred(H_F)$, where $\mathbf{x} = \{x_j \mid 1 \leq j \leq n\}$ is the set of ordinary attributes, and α is the logic attribute to be parameterized (transformed into an annotation), generates the *compressed* predicate $r(\mathbf{x})$ for the annotated logic program. We will denote by $W = \mathcal{B}$ the domain of the logic attribute α of the predicate $r_F(\mathbf{x}, \alpha)$.
Notice that the *compressed* predicate $r(\mathbf{x})$ is not a simple projection $\pi_{\mathbf{x}}$ of the original predicate over attributes in **x**, that is, $r(\mathbf{x}) \neq \pi_{\mathbf{x}} r_F(\mathbf{x}, \alpha)$, because it contains also all hidden parameters, and its interpretations are of higher-type: the logic values of its ground atoms are *functions* instead of classic logic values in $\mathbf{2} = \{0, 1\}$; these functions encapsulate the semantics of the hidden attributes.
Remark: in what follows, for simplicity, we will not make the clear formal distinction between predicate symbol r, its predicate form $r(t_1, t_2)$, where t_1, t_2 are terms (variables or functional forms) and a binary relation. We will use shortly ’predicate’ for a ’predicate form’.
The plan of this paper is the following: After a brief introduction to Many-valued bilattice-based logic, in

Section 2 we introduce the invariant knowledge transformation of predicates, based on the flattening and compression duality; we introduce the higher-order Herbrand interpretations and its correspondent flattening into the ordinary Herbrand interpretations. This duality holds also for the fixpoint semantics of these logic programs: given a fixpoint semantics for the first one we can derive the fixpoint semantics of the second, and vice versa.
In Section 3 we show how these concepts can be applied to the many-valued bilattice-based logic: we show that the annotated 2-valued transformation of the original logic program has the higher-order Herbrand interpretation, isomorphic to the original many-valued Herbrand interpretation, while the 2-valued ontological encapsulation (corresponds to flattening) of the original many-valued logic program has a standard Herbrand interpretation over a new Herbrand base, obtained by enlarging each original predicate by a new logic attribute.

1.1 Introduction to Many-valued epistemic logic based on a Bilattice

In [14], Belnap introduced a logic intended to deal in a useful way with inconsistent or incomplete information. It is the simplest example of a non-trivial bilattice and it illustrates many of the basic ideas concerning them. We denote the four values as $\{t, f, \top, \bot\}$, where t is *true*, f is *false*, $\top$ is inconsistent (both true and false) or *possible*, and $\bot$ is *unknown*. As Belnap observed, these values can be given two natural orders: *truth* order, $\leq_t$, and *knowledge* order, $\leq_k$, such that $f \leq_t \top \leq_t t$, $f \leq_t \bot \leq_t t$, and $\bot \leq_k f \leq_k \top$, $\bot \leq_k t \leq_k \top$. This two orderings define corresponding equivalences $=_t$ and $=_k$. Thus any two members α, β in a bilattice are equal, $\alpha = \beta$, if and only if (shortly 'iff') $\alpha =_t \beta$ and $\alpha =_k \beta$.
Meet and join operators under $\leq_t$ are denoted $\wedge$ and $\vee$; they are natural generalizations of the usual conjunction and disjunction notions. Meet and join under $\leq_k$ are denoted $\otimes$ (*consensus*, because it produces the most information that two truth values can agree on) and $\oplus$ (*gullibility*, it accepts anything it's told), such that hold:
$f \otimes t = \bot$, $f \oplus t = \top$, $\top \wedge \bot = f$ and $\top \vee \bot = t$.
There is a natural notion of truth negation, denoted $\sim$, (reverses the $\leq_t$ ordering, while preserving the $\leq_k$ ordering): switching f and t, leaving $\bot$ and $\top$, and corresponding knowledge negation, denoted $-$ (reverses the $\leq_k$ ordering, while preserving the $\leq_t$ ordering), switching $\bot$ and $\top$, leaving f and t. These two kind of negation commute: $- \sim x = \sim -x$ for every member x of a bilattice.
It turns out that the operations $\wedge, \vee$ and $\sim$, restricted to $\{f, t, \bot\}$ are exactly those of Kleene's strong 3-valued logic. A more general information about bilattice may be found in [15]: he also defines *exact* members of a bilattice, when $x = -x$ (they are 2-valued consistent), and *consistent* members, when $x \leq_k -x$ (they are 3-valued consistent), but a specific 4-valued consistence will be analyzed in the following paragraphs.
The Belnap's 4-valued bilattice is infinitary distributive. In the rest of this paper we denote by $\mathcal{B}_4$ a special case of the Belnap's bilattice.
One of the key insights behind bilattices [8, 9] was the interplay between the truth values assigned to sentences and the (non classic) notion of *implication*. The problem was to study how truth values should be propagated "across" implications. We proposed [1] the implication based approach, which extends the definition in [16] based on the following intuitionistic many valued implication :

$\rightarrow$	t	$\bot$	f	$\top$
t	t	$\bot$	f	$\top$
$\bot$	t	t	$\top$	$\top$
f	t	t	t	t
$\top$	t	$\bot$	$\bot$	t

The Herbrand base, H_B, is the set of all ground (i.e., variable free) atoms. A (ordinary) Herbrand interpretation is a many-valued mapping $I : H_B \rightarrow \mathcal{B}$. If P is a many-valued logic program with the Herbrand base H_B, then the ordering relations and operations in a bilattice $\mathcal{B}$ are propagated to the function space $\mathcal{B}^{H_B}$, that is the set of all Herbrand interpretations (functions), $I_B : H_P \rightarrow \mathcal{B}$. It is straightforward [15] that this makes a function space $\mathcal{B}^{H_B}$ itself a complete infinitary distributive bilattice.

2 Invariant Knowledge transformation: Flattening and compression duality

The higher-order Herbrand interpretations of logic programs (for example Databases), produce models where the true values for ground atoms are not truth constants but functions. In this section we will give the general definitions for such higher-order Herbrand interpretation types for logic programs and their models. We denote by $A \Rightarrow B$, or B^A, the set of all functions from A to B.

Definition 1. *(Higher-order Herbrand interpretation types) Let H^{com} be a Herbrand base. Then, the higher-order Herbrand interpretations are defined by $I_{com} : H^{com} \rightarrow T$, where T denotes the functional space $W_1 \Rightarrow (...(W_n \Rightarrow \mathbf{2})...)$, denoted also as $(...((2^{W_n})^{W_{n-1}})...)^{W_1}$, and W_i, $i \in [1,n]$, $n \geq 1$, the sets of parameters. In the case $n = 1$, $T = (W_1 \Rightarrow \mathbf{2})$, we will denote this interpretation by $I_{com} : H^{com} \rightarrow \mathbf{2}^{W_1}$.*

The interpretations $I_{com} : H^{com} \rightarrow \mathbf{2}^{\mathcal{W}}$ are higher-order types of Herbrand interpretations: the set of truth values for them are *functions* instead of constants. We pass from a flat truth structure for atoms in a Hebrand interpretations of original database $\mathcal{DB}$, to non flat functional space truth structure for atoms in the compressed Herbrand base H^{com}. The hidden parameters make "curve" the truth space for these atoms, as what happens when the real astronomic space curves in a presence of hidden (black) gravitational mass. Notice that in the Def. 3, the interpretation of compressed database $I_{com} : H^{com} \rightarrow \mathbf{2}^{\mathcal{W}}$ is the case when $n = 1$ (the simplest higher type $T = (\mathcal{W} \Rightarrow \mathbf{2})$ for higher Herbrand interpretations).
Now we will introduce the top-down transformation, called *flattening*, where the context (uncertain or approximated information), defined as the set of possible worlds $\mathcal{W}$, is fused into the Herbrand base by enlarging original predicates of old theory with new attributes taken from the context. In this way the hidden information of the context becomes the visible information and a visible part of the logic language.

Definition 2. *(Flattening) Each higher-order Herbrand interpretation $I_{com} : H^{com} \rightarrow T$, where T denotes the functional space $W_1 \Rightarrow (...(W_n \Rightarrow \mathbf{2})...)$, and $\mathcal{W} = W_1 \times ... \times W_n$ cartesian product, can be flattened into the Herbrand interpretation $I_F : H_F \rightarrow \mathbf{2}$, where $H_F = \{r_F(\mathbf{d}, \mathbf{w}) \mid r(\mathbf{d}) \in H^{com}$ and $\mathbf{w} \in \mathcal{W}\}$,
is the Herbrand base of predicates r_F, obtained as extension of original predicates r by parameters, such that for any $r_F(\mathbf{d}, \mathbf{w}) \in H_F$, $\mathbf{w} = (w_1, ..., w_n)$, holds that
$I_F(r_F(\mathbf{d}, \mathbf{w})) = I_{com}(r(\mathbf{d}))(w_1)...(w_n)$.*

We define as *parameterizable* database $\mathcal{DB}(\mathbf{y})$ any database such that *all* its relational schemas have the *common set* of attributes $\mathbf{y} = \{y_1, y_2, .., y_k\}$. In this, most simple case of compression, we can obtain an compressed database, denoted by $\mathcal{DB}_{com}(\mathbf{y})$, in the way that these common attributes become hidden attributes.

Definition 3. *(Global compression).
Let $I_F : H_F \rightarrow \mathbf{2}$ be the 2-valued Herbrand interpretation for a parameterizable database $\mathcal{DB}$ with the Herbrand base H_B and the model $M_H = \{r_F(\mathbf{d}, \mathbf{w}) \mid r_F(\mathbf{d}, \mathbf{w}) \in H_F$ and $I_F(r_F(\mathbf{d}, \mathbf{w})) = 1\}$. Then the interpretation for its compressed database $\mathcal{DB}_{com}(\mathbf{y})$ is defined by
$I_{com} : H^{com} \rightarrow \mathbf{2}^{\mathcal{W}}$, such that $I_{com} = [I_F \circ is]$,
where $\mathcal{W} = Dom_{y_1} \times ... \times Dom_{y_k}$ is the set of all parameter tuples, H^{com} is a Herbrand base for the compressed database $\mathcal{DB}_{com}(\mathbf{y})$, i.e.,
$H^{com} = \{r(\mathbf{d}) \mid \exists \mathbf{w}. r_F(\mathbf{d}, \mathbf{w}) \in H_F\}$,
$is : H^{com} \times \mathcal{W} \simeq H_F$ is a bijection, $[\_]$ is the curring (λ abstraction) for functions, and $\mathbf{2}^{\mathcal{W}}$ is the set of functions from $\mathcal{W}$ to $\mathbf{2} = \{0, 1\}$.*

No one of subsets $S \subseteq H^{com}$ can be *model* for $\mathcal{DB}_{com}$; that is, the models for compressed database *are not* ordinary Herbrand models but some kind of *higher-order type* of Herbrand models.

Definition 4. *In the case of the Global compression, in Definition [3], over variables in $\{w_1, .., w_n\}$, $n \geq 1$ with a domain $\mathcal{W}$ (n-tuples in the Herbrand Universe), we are able to define the following two partial orders*
1. For any two $I_F, I'_F \in \mathbf{2}^{H_F}$ we define the partial order $\leq_F$ as follows
$I_F \leq_F I'_F$ iff $\forall A \in H_F(I_F(A) \leq I'_F(A))$.
2. For any two $I_{com}, I'_{com} \in (\mathbf{2}^{\mathcal{W}})^{H^{com}}$ we define the partial order $\leq_{com}$ as follows
$I_{com} \leq_{com} I'_{com}$ iff $\forall A \in H^{com}, \mathbf{w} \in \mathcal{W}\ (I_{com}(A)(\mathbf{w}) \leq I'_{com}(A)(\mathbf{w}))$.

Obviously, the functional spaces $\mathbf{2}^{H_F}$ and $(\mathbf{2}^{\mathcal{W}})^{H^{com}}$ of ordinary and Higher-order Herbrand interpretations are complete lattices w.r.t. the partial orderings $\leq_F$ and $\leq_{com}$ respectively. We denote by $(\mathbf{2}^{H_F}, \leq_F)$ and $((\mathbf{2}^{\mathcal{W}})^{H^{com}}, \leq_{com})$ the flattened and compressed domains respectively.

Proposition 1 *Let $\alpha : \mathbf{2}^{H_F} \to (\mathbf{2}^{\mathcal{W}})^{H^{com}}$, and $\beta : (\mathbf{2}^{\mathcal{W}})^{H^{com}} \to \mathbf{2}^{H_F}$, be the two mappings such that*
- for any $I_F \in \mathbf{2}^{H_F}$ the $\alpha(I_F)$ is the mapping, such that for each $r(\boldsymbol{d}) \in H^{com}$ and $\mathbf{w} \in \mathcal{W}$, holds that $\alpha(I_F)(r(\boldsymbol{d}))(\boldsymbol{w}) = I_F(r_F(\boldsymbol{d}, \mathbf{w}))$;
- for any $I_{com} \in (\mathbf{2}^{\mathcal{W}})^{H^{com}}$ the $\beta(I_{com})$ is the mapping, such that for each $r_F(\boldsymbol{d}, \mathbf{w}) \in H_F$, holds that $\beta(I_{com})(r_F(\boldsymbol{d}, \mathbf{w})) = I_{com}(r(\boldsymbol{d}))(\boldsymbol{w})$;
Then the following three conditions define a Galois connection between the flattened and compressed domains:
1. $\alpha = [\_ \circ is]$ and $\beta = [\_]^{-1} \circ is^{-1}$ are monotonic functions.
2. $\forall I_{com} \in (\mathbf{2}^{\mathcal{W}})^{H^{com}}$ $(\alpha(\beta(I_{com})) \leq_{com} I_{com})$.
3. $\forall I_F \in \mathbf{2}^{H_F}$ $(\beta(\alpha(I_F)) \leq_F I_F)$.

Proof: $\alpha = [\_ \circ is]$ and $\beta = [\_]^{-1} \circ is^{-1}$ come from Definition 3 and 2 respectively. Let us prove that α is monotonic, and suppose that $I_F \leq_F I'_F$, than from point 3 in Definition 4 we have that for any $r_F(\mathbf{d}, \mathbf{w}) \in H_F$ holds $I_F(r_F(\mathbf{d}, \mathbf{w})) \leq I'_F(r_F(\mathbf{d}, \mathbf{w}))$, that $\alpha(I_F)(r(\mathbf{d}))(\mathbf{w}) \leq \alpha(I'_F)(r(\mathbf{d})))(\mathbf{w}$, or, for $r(\mathbf{d}) \in H_{com}, \mathbf{w} \in \mathcal{W}$ holds $\alpha(I_F)(r(\mathbf{d}))(\mathbf{w}) \leq \alpha(I'_F)(r(\mathbf{d})))(\mathbf{w}$, thus $\forall A \in H_{com}, \mathbf{w} \in \mathcal{W}$ $(\alpha(I_F)(A)(\mathbf{w}) \leq \alpha(I_F)(A))(\mathbf{w}))$, and, consequently, than from point 3 in Definition 4 we have that $\alpha(I_F) \leq_{com} \alpha(I'_F)$. The same holds for β. Points 2 and 3 hold because β is inverse of α. □

Galois connections give us the formal framework to prove the equivalency (the α and β transformations are inverse so that we have the particular case when the property 2 and 3 in the Galois conditions are equations) and mutual-inversion property between the flattened and Higher-order Herbrand interpretations: that is the reason that we informally defined, in Definition 2, the flattening as inverse of data compression.
Informally, the inverse monotonic homomorphisms in Galois connection,
$\alpha : (\mathbf{2}^{H_F}, \bigcup_F) \to ((\mathbf{2}^{\mathcal{W}})^{H^{com}}, \bigcup_{com})$, and $\beta : ((\mathbf{2}^{\mathcal{W}})^{H^{com}}, \bigcup_{com}) \to (\mathbf{2}^{H_F}, \bigcup_F)$,
which represent the mutually-inverse operations of predicate compression and flattening, are information-conservative transformations: by compression we only hide (but do not eliminate) a part of information, by flattening we render such hidden information visible (withot addition of new information). Thus, if we have some monotonic "immediate consequence operator" in one of these two join-semilattice algebras, then intuitively, must exist such one operator also in the other algebra. This intuition can be formalized as follows:

Proposition 2 *(Fixpoint adjunction)*
Let $T_P : \mathbf{2}^{H_F} \to \mathbf{2}^{H_F}$ be a monotonic "immediate consequence operator" in the space of Herbrand interpretations, then $\Phi_P = [T_P([\_]^{-1})] : (\mathbf{2}^{\mathcal{W}})^{H^{com}} \to (\mathbf{2}^{\mathcal{W}})^{H^{com}}$ is the monotonic operator in the higher-order Herbrand space $(\mathbf{2}^{\mathcal{W}})^{H^{com}}$.
Viceversa, let $\Phi_P : (\mathbf{2}^{\mathcal{W}})^{H^{com}} \to (\mathbf{2}^{\mathcal{W}})^{H^{com}}$ be the monotonic operator in the higher-order Herbrand space. Then $T_P = [\Phi_P([\_ \circ is])]^{-1} \circ is^{-1} : \mathbf{2}^{H_F} \to \mathbf{2}^{H_F}$ is the monotonic operator in the Herbrand space.

Proof: Form the Galois connection and Proposition 1 We have that $\Phi_P = \alpha \circ T_P \circ \beta = [\_ \circ is] \circ T_P \circ [\_]^{-1} \circ is^{-1} = [T_P \circ [\_]^{-1} \circ is^{-1} \circ is] = [T_P([\_]^{-1})]$, where α, β are monotonic operators. Thus if T_P is monotonic then also Φ_P is monotonic. Analogously, we have that $T_P = \beta \circ \Phi_P \circ \alpha = [\_]^{-1} \circ is^{-1} \circ \Phi_P \circ [\_ \circ is] = [\Phi_P([\_ \circ is])]^{-1} \circ is^{-1}$. □

3 Predicate Compression and Flattening in Many-valued Logic Programming

It is obvious that the immediate-consequence monotonic operator in the Herbrand join-semilattice is w.r.t. the truth ordering $\leq_F$; but its "dual" operator in the higher-order join-semilattice can have different from truth ordering. For instance, in the case of normal logic programs and its 3-valued fixpoint semantics WFS (Well-Founded Semantics for normal programs) based on the strong Kleene 3-valued logic, such semantics can not be formalized in the ordinary Herbrand join-semilattice algebra- that means that, in some way, the WFS must be embedded into the higher-order Herbrand join-semilattice with *knowledge* ordering $\leq_{com}$.

Interesting point, analyzed in less general framework, is that the flattening of such knowledge (higher-order) monotonic operator into ordinary Herbrand space of interpretations *must be* truth ordered [2]. Such flattening of the original many-valued bilattice based program into the 2-valued logic is denominated by *ontological-encapsulation*, where encapsulation of original many-valued logic program into the 2-valued "meta" logic program corresponds to the flattening process described here so that also the logic values in the Belnap's bilattice $\mathcal{B}$ from hidden become visible values in this "meta" logic program, and is developed in a number of papers [17, 1].
Here we will present a slightly modified version of this ontological encapsulation of a many-valued logic program.
We assume that the Herbrand universe is $\Gamma_U = \Gamma \bigcup \Omega$, where Γ is ordinary domain of database constants, and Ω is an infinite enumerable set of marked null values, $\Omega = \{\omega_0, \omega_1, ...\}$, and for a given logic program P composed by a set of predicate and function symbols, P_S, F_S respectively, we define a set of all terms, $\mathcal{T}_S$, and its subset of ground terms $\mathcal{T}_0$. Then the atoms are defined as:
$\mathcal{A}_S = \{p(c_1, .., c_n) \mid p \in P_S,\ n = arity(p)$ and $c_i \in \mathcal{T}_S\}$.
We introduce [17] the program encapsulation (flattening) transformation $\mathcal{E}$:

Definition 5. *(Flattening of Many-valued Logic Programs)*
Let P be a many-valued logic program with the set of predicate symbols P_S, Herbrand base H_B, and for any predicate $p \in P_S$ we introduce a mapping $\kappa_p : \mathcal{T}_0^{arity(p)} \rightarrow \mathcal{B}$. The translation $\mathcal{E}$ in the encapsulated syntax version P_F is as follows:
1. Each positive literal in P, we introduce a new predicate p_F as follows
$\mathcal{E}(p(x_1, .., x_n)) = p_F(x_1, .., x_n, \kappa_p(x_1, .., x_n))$;
2. Each negative literal in P, we introduce a new predicate p_F as follows
$\mathcal{E}(\sim p(x_1, .., x_n)) = p_F(x_1, .., x_n, \sim \kappa_p(x_1, .., x_n))$;
3. $\mathcal{E}(\phi \wedge \varphi) = \mathcal{E}(\phi) \wedge \mathcal{E}(\varphi)$; $\quad \mathcal{E}(\phi \vee \varphi) = \mathcal{E}(\phi) \vee \mathcal{E}(\varphi)$;
4. $\mathcal{E}(\phi \leftarrow \varphi) = \mathcal{E}(\phi) \leftarrow^A \mathcal{E}(\varphi)$ *, where $\leftarrow^A$ is a symbol for the implication at the encapsulated 2-valued 'meta' level. Thus, the obtained 'meta' program is equal to $P_F = \{\mathcal{E}(\phi) \mid \phi$ is a clause in $P\}$, with the 2-valued Herbrand base*
$H_F = \{\ p_F(c_1, .., c_n, \alpha) \mid p(c_1, .., c_n) \in H_B$ *and* $\alpha \in \mathcal{B}\}$.

This embedding of the many-valued logic program P into a 2-valued 'meta' logic program P_F is an *ontological* embedding: it views formulae of P as beliefs and interprets the negation $\sim p(x_1, .., x_n)$ in rather restricted sense - as belief in the falsehood of $p(x_1, .., x_n)$, rather as not believing that $p(x_1, .., x_n)$ is true (like in an ontological embedding for classical negation). Like Moore's autoepistemic operator, the encapsulation operator $\mathcal{E}$ (restricted to atoms), $\mathcal{E}\phi$ intends to capture the notion of, "I know that ϕ has a value $v_B(\phi)$ ", for a given valuation v_B of the many-valued logic program.
Notice, that with the transformation of the original many-valued logic program P into its encapsulated 'meta' version program P_F we obtain *a positive* logic program, thus with a unique minimal Herbrand model $I_F : H_F \rightarrow \mathbf{2}$.
Now we will show how many-valued logic can be transformed into 2-valued logic with higher-order Herbrand interpretations.
Let us consider an original many-valued logic clause (bold variables denote a tuple of variables)
$A(\mathbf{x}) \ \leftarrow \ B_1(\mathbf{x}_1), .., B_k(\mathbf{x}_k), \sim B_{k+1}(\mathbf{x}_{k+1}), ..., \sim B_k(\mathbf{x}_k)$.
Its transformation into an annotated clause will be the following
$A(\mathbf{x}) : \psi \ \leftarrow \ B_1(\mathbf{x}_1) : \beta_1, .., B_k(\mathbf{x}_k) : \beta_k,\ B_{k+1}(\mathbf{x}_{k+1}) : \beta_{k+1}, ..., B_k(\mathbf{x}_k) : \beta_n$
where ψ is a bilattice algebra term $(\beta_1 \wedge .. \wedge \beta_k \wedge \sim \beta_{k+1} \wedge ..\wedge \sim \beta_n)$. So that the interpretation (true or false) of each ground annotated atom $B_i(\mathbf{c}_i) : \beta_i$ is a mapping $I_A : H_B \times \mathcal{B} \rightarrow \mathbf{2}$, such that $I_A(B_i(\mathbf{c}_i), \beta_i)$ has the true or false value in $\mathbf{2}$, or equivalently (by λ-abstraction), $I_{com} = \lambda(I_A) : H_B \rightarrow \mathbf{2}^{\mathcal{B}}$, with $I_{com}(B_i(\mathbf{c}_i)) = h : \mathcal{B} \rightarrow \mathbf{2}$ a function from $\mathcal{B}$ into the set of truth values $\mathbf{2}$.
The image of this mapping is a subset, denoted by $\mathcal{B}_\delta \subset \mathbf{2}^{\mathcal{B}}$, such that its cardinality is equal to the cardinality of $\mathcal{B}$, that is $|\mathcal{B}_\delta| = |\mathcal{B}|$, defined as follows:

Definition 6. *(Higher-order Belnap's bilattice)*
We define the functional version of the Belnap's bilattice $\mathcal{B}_\delta$ as follows,
$\mathcal{B}_\delta = \{\delta_\kappa : \mathcal{B} \rightarrow \mathbf{2},\ \kappa \in \mathcal{B}$ *such that* $\delta_\kappa(\upsilon) = 1$ *if* $\upsilon = \kappa$; *0, otherwise* $\} \quad \subseteq \ \mathbf{2}^{\mathcal{W}}$,

with truth *order,* $\leq_t$*, and* knowledge *order,* $\leq_k$*, such that* $\delta_f \leq_t \delta_\top \leq_t \delta_t$*,* $\delta_f \leq_t \delta_\perp \leq_t \delta_t$*, and* $\delta_\perp \leq_k \delta_f \leq_k \delta_\top$*,* $\delta_\perp \leq_k \delta_t \leq_k \delta_\top$*.*

This functional Belnap's bilattice $\mathcal{B}_\delta$ is interchangeable with original Belnap's bilattice $\mathcal{B}$ because of the following isomorphism:

Proposition 3 *(Belnap's isomorphism) There exists the isomorphism* $\delta : \mathcal{B} \simeq \mathcal{B}_\delta$*, such that for any* $\upsilon \in \mathcal{B}$*,* $\delta(\upsilon) = \delta_\upsilon$*, and* $\delta^{-1}(\delta_\upsilon) = \upsilon$*.*

Let $I_B : H_B \rightarrow \mathcal{B}$, that is, $I_B \in \mathcal{B}^{H_B}$, be a many-valued interpretation for a normal logic program P with a Herbrand base $\mathcal{B}$. Such many-valued interpretation can be transformed into the many-valued interpretation $J : H_B \rightarrow \mathcal{B}_\delta \subseteq \mathbf{2}^{\mathcal{B}}$, where the original Belnap's 4-valued knowledge-ordered lattice $\mathcal{B}$ is replaced by the 4-valued lattice $\mathcal{B}_\delta$ by the $\delta : \mathcal{B} \simeq \mathcal{B}_\delta$ bijection, $\kappa \mapsto \delta_\kappa$, which *preserves the knowledge ordering*, so that for any ground atom $A \in H_B$, $J(A) = \delta_{I_B(A)}$.
We denote by $\preceq_k$ its extension to the functional space $(\mathbf{2}^{\mathcal{B}})^{H_B}$ such that:
$J \preceq_k J'$ iff $\forall A \in H_B \; (J(A) \leq_k J'(A))$.
The following proposition demonstrates that by such many-valued logic transformation we obtain the logic program with higher-order Herbrand models type.

Proposition 4 *Let* P *be a normal logic program with a Herbrand base* H_B*, over the lattice* $\mathcal{B}$ *with knowledge ordering* $\leq_k$ *and the Fitting's monotonic "immediate-consequence operator"* $\Psi_P : \mathcal{B}^{H_B} \rightarrow \mathcal{B}^{H_B}$*. Let denote by* Φ_P *the same monotonic operator obtained by replacing the original lattice* $\mathcal{B}$ *with equivalent to it lattice* $\mathcal{B}_\delta$*. Then this operator is monotonic operator over the join-semilattice* $((\mathbf{2}^{\mathcal{W}})^{H^{com}}, \bigcup_{com})$*, where* $\mathcal{W} = \mathcal{B}$ *and* $H^{com} = H_B$*.*

Proof: From the definitions above we have that $I_{com} = J$ and that the the extension of the knowledge ordering, $\preceq_k$ is equal to the partial ordering $\leq_{com}$, so, by replacing original lattice $\mathcal{B}$ by the equivalent "functional' lattice $\mathcal{B}_\delta$, we obtain the logic program with higher-order Herbrand models. □
Notice that the initial interpretation $I_{B_0} : H_B \rightarrow \mathcal{B}$ for many-valued fixpoint operator $\Psi_P : \mathcal{B}^{H_B} \rightarrow \mathcal{B}^{H_B}$ is a bottom knowledge element in the lattice of the many-valued interpretations $\mathcal{B}^{H_B}$ (such that for any ground atom $A \in H_B$, $I_{B_0}(A) = \perp$), corresponds to the initial higher-order interpretation $I_{com_0} : H_B \rightarrow \mathbf{2}^{\mathcal{B}}$, which is a bottom element in the join-semilattice $((\mathbf{2}^{\mathcal{B}})^{H_B}, \preceq_k) = ((\mathbf{2}^{\mathcal{W}})^{H^{com}}, \leq_{com})$, such that $I_{com_0}(A) = \delta_\perp$. So, the reducibility of the knowledge fixpoint semantics to the truth fixpoint semantics, described in [2] where is defined also the transformation of the original many-valued logic programs into 2-valued "meta" logic programs (at the level of Herbrand bases it equals to the flattening), is the direct consequence of the propositions 2 and 4.
In fact, let $I_B : H_B \rightarrow \mathcal{B}$ be the least fixpoint many-valued Herbrand model of a many-valued logic program P, and $I_{com} : H^{com} \rightarrow \mathbf{2}^{\mathcal{W}}$ be the least fixpoint in the joint- semilattice $((\mathbf{2}^{\mathcal{W}})^{H^{com}}, \bigcup_{com})$ obtained from the many-valued Herbrand model I_B by replacing the logic values in $\mathcal{B}$ with their correspondent values in $\mathcal{B}_\delta$.
Then, for the Herbrand interpretation $I_F = \beta(I_{com}) : H_F \rightarrow \mathbf{2}$, the *consistent* 2-valued flattened model is equal to:
$M_F = \{r_F(\mathbf{d}, \upsilon) \mid I_{com}(r(\mathbf{d})) = \delta_\upsilon\} = \{r_F(\mathbf{d}, \upsilon) \mid I_F(r_F(\mathbf{d}, \upsilon)) = 1\}$.
The *consistency* of the flattened model M_F means that cannot exist two ground atoms $r_F(\mathbf{d}, \upsilon\}$ and $r_F(\mathbf{d}, \upsilon_1\}$ in M_F such that $\upsilon \neq \upsilon_1$. Such consistency correspond to the fact that each ground atom $r(\mathbf{d}) \in H_B$ in the original many-valued program P can have only one value in the Belnap's bilattice $\mathcal{B}$.
Vice versa, given a consistent 2-valued 'meta' model M_F, we can define the compressed Herbrand interpretation $I_{com} : H^{com} \rightarrow \mathbf{2}^{\mathcal{W}}$, by:
$I_{com}(r(d)) = \delta_\upsilon$ iff $r_F(\mathbf{d}, \upsilon) \in M_F$
so that a many-valued Herbrand model $I_B : H_B \rightarrow \mathcal{B}$ can be obtained from I_{com} by replacing the logic values in $\mathcal{B}_\delta$ by correspondent values in $\mathcal{B}$.
This correspondence between the many-valued Herbrand model $I_B : H_B \rightarrow \mathcal{B}$ of the original many-valued logic program, with the two 2-valued knowledge invariant transformations, annotated program with a higher-order Herbrand model $I_{com} : H_B \rightarrow \mathbf{2}^{\mathcal{B}}$ and flattened (ontologically encapsulated) 'meta' logic program with a Herbrand model $I_F : H_F \rightarrow \mathbf{2}$ (over an enlarged Herbrand base H_F), can be briefly represented by the following table:

PROGRAM	Original program P	Annotated program P_A	'meta' program P_F (ontological encapsulation)
LOGIC	Many-valued logic	2-valued logic	2-valued logic
HERBRAND MODEL	Many-valued $I_B : H_B \to \mathcal{B}$	Higher-order $I_{com} : H_B \to \mathcal{B}_\delta \subset \mathbf{2}^{\mathcal{B}}$, (bijection $\mathcal{B} \simeq \mathcal{B}_\delta$)	Standard $I_F : H_F \to \mathbf{2}$, with $H_F = \{p_F(\mathbf{c}, \alpha) \vert p(\mathbf{c}) \in H_B, \alpha \in \mathcal{B}\}$
GALOIS		compression	flattening

so that for any ground atom $p(\mathbf{c}) \in H_B$ hold

$$I_{com}(p(\mathbf{c}))(I_B(p(\mathbf{c}))) = I_F(p(\mathbf{c}, I_B(p(\mathbf{c})))) = 1, \quad \text{and}$$
$$I_{com}(p(\mathbf{c}))(\alpha) = I_F(p(\mathbf{c}, \alpha)) = 0, \quad \text{if} \quad \alpha \neq I_B(p(\mathbf{c})),$$

which express the knowledge invariancy between program transformations.

4 Conclusion

In this paper we have shown how the implication-based Many-valued logic programs can be equivalently transformed into the Annotated logic programs with higher-order Herbrand interpretations. We have shown that the flattening of such higher-order Herbrand interpretations leads to the two-valued logic programs, identical to 'meta' logic programs obtained by an ontological encapsulation of the original Many-valued logic programs [17, 1].
Consequently, the Galois connection of the fixpoint semantics between higher-order Herbrand interpretations and their flattening into ordinary two-valued Herbrand interpretations, is the more general version of the knowledge-truth correspondence [2] between the fixpoint semantics of the original Many-valued logic programs and their ontologically encapsulated two-valued logic programs.

References

1. Z.Majkić, "Many-valued intuitionistic implication and inference closure in a bilattice based logic," *35th International Symposium on Multiple-Valued Logic (ISMVL 2005), May 18-21, Calgary, Canada*, 2005.
2. Z.Majkić, "Truth and knowlwdge fixpoint semantics for many-valued logic programming," *19th Workshop on (Constraint) Logic Programming (W(C)LP 2005), February 21-25, Ulm, Germany*, 2005.
3. G.Escalada Imaz and F.Manyá, "The satisfiability problem for multiple-valued Horn formulae," *In Proc. International Symposium on Multiple-Valued Logics (ISMVL), Boston, IEEE Press, Los Alamitos*, pp. 250–256, 1994.
4. B.Beckert, R.Hanhle, and F.Manyá, "Transformations between signed and classical clause logic," *In Proc. 29th Int.Symposium on Multiple-Valued Logics, Freiburg,Germany*, pp. 248–255, 1999.
5. H.A.Blair and V.S.Subrahmanian, "Paraconsistent logic programming," *Theoretical Computer Science,68*, pp. 135–154, 1989.
6. M.Kifer and E.L.Lozinskii, "A logic for reasoning with inconsistency," *Journal of Automated reasoning 9(2)*, pp. 179–215, 1992.
7. M.Kifer and V.S.Subrahmanian, "Theory of generalized annotated logic programming and its applications," *Journal of Logic Programming 12(4)*, pp. 335–368, 1992.
8. M.Ginsberg, "Multivalued logics: A uniform approach to reasoning in artificial intelligence," *Computational Intelligence, vol.4*, pp. 265–316, 1988.
9. M.C.Fitting, "Bilattices and the semantics of logic programming," *Journal of Logic Programming,11*, pp. 91–116, 1991.
10. M.H.van Emden, "Quantitative deduction and its fixpoint theory," *Journal of Logic Programming,4,1*, pp. 37–53, 1986.
11. S.Morishita, "A unified approach to semantics of multi-valued logic programs," *Tech. Report RT 5006, IBM Tokyo*, 1990.
12. V.S.Laksmanan and N.Shiri, "A parametric approach to deductive databases with uncertainty," *IEEE Transactions on Knowledge and Data Engineering,13(4)*, pp. 554–570, 2001.

13. Z.Majkić, “Autoepistemic logic programming for reasoning with inconsistency,” *International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR), September 7-9, 2005, Imperial College, London,UK*, 2005.
14. N.D.Belnap, “A useful four-valued logic,” *In J-M.Dunn and G.Epstein, editors, Modern Uses of Multiple-Valued Logic. D.Reidel*, 1977.
15. M.C.Fitting, “Billatices are nice things,” *Proceedings of Conference on Self-Reference, Copenhagen*, 2002.
16. T.Przymusinski, “Every logic program has a natural stratification and an iterated fixed point model,” *In Eighth ACM Symposium on Principles of Databases Systems*, pp. 11–21, 1989.
17. Z.Majkić, “Ontological encapsulation of many-valued logic,” *19th Italian Symposium of Computational Logic (CILC04),June 16-17, Parma, Italy*, 2004.