Ontological encapsulation of many-valued logic

Zoran Majkić

Dipartimento di Informatica e Sistemistica, University of Roma "La Sapienza"
Via Salaria 113, I-00198 Rome, Italy
majkic@dis.uniroma1.it
http://www.dis.uniroma1.it/ ∼ majkic/

Abstract. Large databases obtained by the data integration of different source databases can be incomplete and inconsistent in many ways. The classical logic is not the appropriate formalism for reasoning about inconsistent databases. Certain local inconsistencies should not be allowed to significantly alter the intended meaning of such logic programs. The variety of semantical approaches that have been invented for logic programs is quite broad. In particular we are interested for many-valued logics with negation, based on bilattices. We present a 2-valued logic, based on an Ontological Encapsulation of Many-Valued Logic Programming, which overcome some drawbacks of the previous research approaches in many-valued logic programming. We defined a Model theory for Herbrand interpretations of ontologically encapsulated logic programs, based on a semantic reflection of the epistemic many-valued logic.

1 Introduction to Many-valued logic programming

Semantics of logic programs are generally based on a classical 2-valued logic by means of stable models, [1,2]. Under these circumstances not every program has a stable model. Three-valued, or partial model semantics had an extensive development for logic programs generally, [3,4]. Przymusinski extended the notion of stable model to allow 3-valued, or partial, stable models, [5], and showed every program has at least one partial stable model, and the well-founded model is the smallest among them, [6]. Once one has made the transition from classical to partial models allowing *incomplete* information, it is a small step to also allow models admitting *inconsistent* information. Doing so provides a natural framework for the semantic understanding of logic programs that are distributed over several sites, with possibly conflicting information coming from different places. As classical logic semantics decrees that inconsistent theories have no models, classical logic is not the appropriate formalism for reasoning about inconsistent databases: certain "localizable" inconsistences should not be allowed to significantly alter the intended meaning of such databases.
So far, research in many-valued logic programming has proceeded along different directions: *Signed* logics [7,8] and *Annotated* logic programming [9,10] which can be embedded into the first, *Bilattice-based* logics, [11,12], and *Quantitative rule-sets*, [13,14]. 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.

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 [15] 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 bilattice) - it is a sign that strict bilattice language is not enough expressive for logic programming, and we need some reacher (different) syntax for logical programming.

From the above points, 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 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 syntactically seen as a kind of very simple annotated syntax. Thus the implication (and classical negation also), not present in a bilattice algebra operators, will have a natural semantic interpretation in this enriched framework.

In [10] it is shown how the Fitting's 3-valued bilattice logic can be embedded into an Annotated Logic Programming which is computationaly very complex. The aim of this work is (1) to extend the Fittihg's fixpoint semantics to deal with inconsistencies also, and (2) to define the notion of a model for such many-valued logic programs by some kind of 'minimal' (more simple and less computationally expensive than APC) logic. In order to respond to these questions we (1) introduce *built-in predicates* in the heads of clauses, and (2) *encapsulate* the 'object' epistemic many-valued logic programs into 2-valued 'meta' ontological logic programs. We argue that such logic will be good framework for supporting the data integration systems with key and foreign

key integrity constraints with incomplete and inconsistent source databases, with less computation complexity for certain answers to conjunctive queries [16,17].
The plan of this paper is the following: Section 2 introduce the Belnap's bilattice concepts and the particular 4-valued version, $\mathcal{B}_4$, used in this paper. In Section 3 is presented an inference framework for a 4-valued bilattice based logic, particularly for derivation of *possible* facts (w.r.t. true and false facts as in 3-valued strong Kleene's logic) and is given a representation theorem for this 4-valued logic. In Section 4 is developed conceptual framework for encapsulation of this epistemic 'object' 4-valued logic into an ontological 'meta' 2-valued logic by mean of *semantic reflection*. Moreover, is given the definition for a 4-valued implication useful for inconsistent databases and an example where inconsistency is managed by clauses with built-in predicate in a head. Finally, Section 5 defines the syntax and the *model theoretic* Herbrand semantics for the ontological encapsulation of many-valued logic programs.

2 Many-valued epistemic logic based on a Bilattice

In [18], 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 $\{t, f, \mathsf{T}, \bot\}$, where t is *true*, f is *false*, T 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 $f \leq_t \mathsf{T} \leq_t t$, $f \leq_t \bot \leq_t t$, and $\bot \leq_k f \leq_k \mathsf{T}$, $\bot \leq_k t \leq_k \mathsf{T}$. 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:
$f \otimes t = \bot$, $f \oplus t = \mathsf{T}$, $\mathsf{T} \wedge \bot = f$ and $\mathsf{T} \vee \bot = t$.
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 T, and corresponding knowledge negation, denoted $-$ (reverses the $\leq_k$ ordering, while preserving the $\leq_t$ ordering), switching $\bot$ and T, 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. Any bilattice $\langle \mathcal{B}, \leq_t, \leq_k \rangle$ is:
1. *Interlaced*, if each of the operations $\wedge, \vee, \otimes$ and $\oplus$ is monotone with respect to both orderings (for instance, $x \leq_t y$ implies $x \otimes z \leq_t y \otimes z$, $x \leq_k y$ implies $x \wedge z \leq_k y \wedge z$).
2. *Infinitarily interlaced*, if it is complete and four infinitary meet and join operations are monotone with respect to both orderings.
3. *Distributive*, if all 12 distributive laws connecting $\wedge, \vee, \otimes$ and $\oplus$ are valid.
4. *Infinitarily distributive*, if it is complete and infinitary, as well as finitary, distributive laws are valid. (Note that a bilattice is *complete* if all meets and joins exist, w.r.t. both orderings. We denote infinitary meet and join w.r.t. $\leq_t$ by $\bigwedge$ and $\bigvee$, and by $\prod$ and $\sum$

for the $\leq_k$ ordering; for example, the distributive law for $\otimes$ and $\bigwedge$ may be given by $x \otimes \bigwedge_i y_i = \bigwedge_i (x \otimes y_i)$).
A more general information about bilattice may be found in [19]: 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. In this way we consider the *possible* value as weak true value and not as inconsistent (that is true and false together). We have more knowledge for ground atom with such value, w.r.t. the true ground atom, because we know also that if we assign the true value to such atom we may obtain an inconsistent database.

3 Representation theorem

Ginsberg [11] defined a world-based bilattices, considering a collection of worlds W, where by world we mean some possible way of things might be, and where $[U, V]$ is a pair of subsets of W which express truth of some sentence p, with $\leq_t, \leq_k$ truth and knowledge preorders relatively, as follows:
1. U is a set of worlds where p is true, V is a set of worlds where p is false, $P = U \bigcap V$ is a set where p is inconsistent (both true and false), and $W - (U \bigcup V)$ is a set where p is unknown.
2. $[U, V] \leq_t [U_1, V_1]$ iff $U \subseteq U_1$ and $V_1 \subseteq V$
3. $[U, V] \leq_k [U_1, V_1]$ iff $U \subseteq U_1$ and $V \subseteq V_1$
Such definition is well suited for the 3-valued Kleene logic, but for the 4-valued logic used to overcome "localizable" inconsistencies it is not useful, mainly for two following reasons:
1. The *inconsistent* (both true and false) top knowledge value in the Belnap's bilattice can't be assigned to sentences, otherwise we will obtain an inconsistent logic theory; because of that consistent logics in this interpretation can have only three remaining values. Thus we interpret it as *possible* value, which will be assigned to mutually inconsistent sentences, and we obtain possibility to have consistent 4-valued logic theories in order to overcome such inconsistencies.
2. Let denote by $T = U - P$, $F = V - P$, where P is a set of worlds where p has a possible logic value. Then we obtain that $[U, V] \leq_t [U_1, V_1]$ also when $T \supset T_1$, which is in contrast with our intuition. Consequently, we adopt a triple $[T, P, F]$ of mutually disjoint subsets of W to express truth of some sentence p ($W - T \bigcup P \bigcup F$ are worlds where p is unknown), with the following definition for their truth and knowledge orders:
2.1 $[T, P, F] \leq_t [T_1, P_1, F_1]$ iff $T \subseteq T_1$ and $F_1 \subseteq F$
2.2 $[T, P, F] \leq_k [T_1, P_1, F_1]$ iff $T \subseteq T_1$, $P \subseteq P_1$ and $F \subseteq F_1$.
Let us try now to render *more rational* these two intuitions described above. In order to obtain a new bilattice abstraction rationality, useful to manage logic programs with possible 'localizable' inconsistencies, we need to consider more deeply the *fundamental phenomena* in such one framework. In the process of derivation of new facts, for a given logic program, based on the 'immediate consequence operator', we have the following

three truth transformations for ground atoms in a Herbrand base of such program:
1. When ground atom pass from *unknown* to *true* logic value, without generating inconsistence. Let denote this action by $\uparrow_1{:}\bot \rightarrowtail t$. The preorder of this 2-valued sublattice of $\mathcal{B}$, $L_1 = \{\bot, t\}$, defined by the direction of this transformation, 'truth increasing', is $\leq_1 \equiv \leq_t$. The meet and join operators for this lattice are $\wedge, \vee$ respectively. It is also knowledge increasing.
2. When some ground atom, try to pass from unknown to true/false value, generating an inconsistency, then is applied the *inconsistency repairing*, that is the *true* value of the literal of this atom, in a body of a violated clause with built-in predicate, is replaced by *possible* value. Let denote this action by $\uparrow_2{:}\, t \rightarrowtail \intercal$. The preorder of this 2-valued sublattice of $\mathcal{B}$, $L_2 = \{t, \intercal\}$, defined by the direction of this transformation, 'knowledge increasing'. The meet and join operators for this lattice, w.r.t. this ordering are $\otimes, \oplus$ respectively. Notice that this transformation *does not change* the truth ordering because the ground atom pass from unknown to possible value.
3. When ground atom pass from *unknown* to *false* logic value, without generating inconsistence. Let denote this action by $\uparrow_3{:}\bot \rightarrowtail f$. The preorder of this 2-valued sublattice of $\mathcal{B}$, $L_3 = \{\bot, f\}$, defined by the direction of this transformation, 'falsehood increasing' (inverse of 'truth increasing'), is $\leq_3 \equiv \leq_t^{-1}$. The meet and join operators for this lattice are $\vee, \wedge$ respectively. It is also knowledge increasing.
Thus, any truth transformation in some multi-valued logic theory (program) can be seen as composition of these three orthogonal dimensional transformations,i.e. by triples (or *multi-actions*), $[a_1, a_2, a_3]$, acting on the idle (default) state $[\bot, t, \bot]$; for instance the multi-action $[\_, \_, \uparrow_3]$, composed by the singe action $\uparrow_3$, applied to the default state generates the "false" state $[\bot, t, f]$. The default state $[\bot, t, \bot]$ in this 3-dimensional space has role as unknown value for single-dimensional bilattice transformations, that is it is a "unknown" state. Consequently, we define this space of states by the cartesian product of single-dimensional lattices, $L_1 \times L_2 \times L_3$, composed by triples $[x, y, z]$, $x \in L_1 = \{\bot, t\}$, $y \in L_2 = \{t, \intercal\}$ and $z \in L_3 = \{\bot, f\}$.

Definition 1. *By $L_1 \odot L_2 \odot L_3$ we mean the bilattice $< L_1 \times L_2 \times L_3, \leq_t^B, \leq_k^B >$ where, given any $X = [x, y, z]$, and $X_1 = [x_1, y_1, z_1]$:*
1. Considering that the second transformation does not influence the truth ordering,
$X \leq_t^B X_1$ *if* $x \leq_1 x_1$ *and* $z \leq_3 z_1$ *, i.e., if* $x \leq_t x_1$ *and* $z \geq_t z_1$
2. Considering that all three transformations are knowledge increasing, we have
$X \leq_k^B X_1$ *if* $x \leq_k x_1$ *and* $y \leq_k y_1$ *and* $z \leq_k z_1$
3. $X \wedge_B X_1 =_{def} [(x \wedge_1 x_1,\ y \wedge_1 y_1), z \wedge_3 z_1] = [x \wedge x_1,\ y \wedge y_1,\ z \vee z_1]$
4. $X \vee_B X_1 =_{def} [x \vee_1 x_1, (y \vee_3 y_1,\ z \vee_3 z_1)] = [x \vee x_1,\ y \wedge y_1,\ z \wedge z_1]$
5. $X \otimes_B X_1 =_{def} [x \otimes x_1,\ y \otimes y_1,\ z \otimes z_1]$
6. $X \oplus_B X_1 =_{def} [x \oplus x_1,\ y \oplus y_1,\ z \oplus z_1]$

These three bilattice transformations can be formally defined by lattice homomorphisms.

Proposition 1 *The following three lattice homomorphisms defines the 3-dimensional truth transformations:*
1. Truth dimension, $\theta_1 = \_ \vee \bot{:}\ (\mathcal{B}, \wedge, \vee, \otimes, \oplus) \rightarrow (L_1, \wedge_1, \vee_1, \otimes, \oplus)$,
with $\wedge_1 = \wedge$, $\vee_1 = \vee$. *This is a strong positive transformation, which transforms*

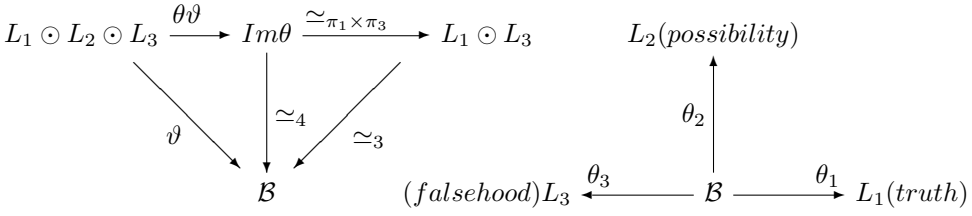
falsehood into unknown and possibility in truth.
2. Possibility dimension, $\theta_2 = \_\vee \sim \_ \vee \mathsf{T} : (\mathcal{B}, \otimes, \oplus) \rightarrow (L_2, \otimes, \oplus)$*. This is a weak knowledge transformation which transform unknown into possibility.*
3. Falsehood dimension, $\theta_3 = \_\wedge \bot$*:* $(\mathcal{B}, \vee, \wedge, \otimes, \oplus) \rightarrow (L_3, \wedge_3, \vee_3, \otimes, \oplus)$*, with* $\wedge_3 = \vee$*,* $\vee_3 = \wedge$*. This is a strong negative transformation, which transforms truth into unknown and possibility into falsehood.*
We define the following two mappings between Belnap's and its derived bilattice:
Dimensional partitioning: $\theta =< \theta_1, \theta_2, \theta_3 >: \mathcal{B} \rightarrow L_1 \odot L_2 \odot L_3$ *and*
Collapsing: $\vartheta : L_1 \odot L_2 \odot L_3 \rightarrow \mathcal{B}$*, such that* $\vartheta(x_1, x_2, x_3) =_{def} (x_1 \oplus x_3) \wedge x_2$*.*

These three lattice homomorphisms preserves the bilattice structure of $\mathcal{B}$ into the space of states $L_1 \odot L_2 \odot L_3$. That is we have that ($'\_'$ represents no action)
$\theta(\bot) = [\_, \_, \_]([\bot, t, \bot]) = [\bot, t, \bot]$, unknown state
$\theta(f) = [\_, \_, \uparrow_3]([\bot, t, \bot]) = [\bot, t, f]$, false state
$\theta(t) = [\uparrow_1, \_, \_]([\bot, t, \bot]) = [t, t, \bot]$, true state
$\theta(\mathsf{T}) = [\uparrow_1, \uparrow_2, \uparrow_3]([\bot, t, \bot]) = [t, \mathsf{T}, f]$, possible state.
Notice that the multi-action $[\uparrow_1, \uparrow_2, \uparrow_3]$ represents two cases for repairing inconsistencies: first, when unknown value of some ground atom tries to become true (action $\uparrow_1$) but makes inconsistency, thus is applied also action $\uparrow_2$ to transform it into possible value; second, when unknown value of some ground atom tries to become false (action $\uparrow_3$) but makes inconsistency, thus is applied also action $\uparrow_2$ to transform it into possible value. Notice that the isomorphism between the set of states and the set of multi-actions $\{[a_1, a_2, a_3] \mid a_1 \in \{\uparrow_1, \_\}, \ a_2 \in \{\uparrow_2, \_\}, \ a_3 \in \{\uparrow_3, \_\}\}$ defines the *semantics* to the bilattice $L_1 \odot L_2 \odot L_3$.

Proposition 2 *Let* $Im\theta \subseteq L_1 \odot L_2 \odot L_3$ *be the bilattice obtained by image of Dimensional partitioning. It has also unary operators:*
Negation, $\sim_B = \theta \sim \vartheta$ *, and conflation,* $-_B = \theta - \vartheta$*.*

It is easy to verify that $\vartheta \circ \theta = id_{\mathcal{B}}$ is an identity on $\mathcal{B}$, and that ϑ is surjective with $\theta \circ \vartheta = id_{Im\theta}$. The negation $\sim_B$ preserves knowledge and inverts truth ordering and $\sim_B \sim_B X = X$; the conflation $-_B$ preserves truth and inverts knowledge ordering and $-_B -_B X = X$; and holds the commutativity $\sim_B -_B = -_B \sim_B$. (for example, $\sim_B -_B = \theta \sim \vartheta\theta - \vartheta = \theta \sim id_{\mathcal{B}} - \vartheta = \theta \sim -\vartheta = \theta - \sim \vartheta = \theta - \vartheta\theta \sim \vartheta = -_B \sim_B$). So, we obtain that, for any $X = [x, y, z]$, hold $\sim_B X =_{def} [\sim z, \ y, \sim x]$ and $-_B X =_{def} [\theta_1(-z), \ \theta_2(-\vartheta(X)), \ \theta_3(-x)]$.

Theorem 1. *(Representation theorem) If* $\mathcal{B}$ *is a 4-valued distributive lattice then there are its distributive sublattices,* L_1, L_2, L_3*, such that* $\mathcal{B}$ *is isomorphic to the sublattice of* $L_1 \odot L_2 \odot L_3$ *defined by image of Dimensional partitioning* $Im\theta$ *. Moreover the following diagram (on the left) of bilattice homomorphisms commute*



where $\simeq_{\pi_1 \times \pi_3}$ is a projection isomorphism, $\simeq_3$ is the isomorphism (restriction of ϑ to the projection $L_1 \odot L_3$) of Fitting's representation Th. [20] valid for a 3-valued logics, and $\simeq_4$ is new 4-valued isomorphism (restriction of ϑ to $Im\theta$, and inverse to θ).
If $\mathcal{B}$ has negation and conflation operators that commute with each other, they are preserved by all isomorphisms of the right commutative triangle.

Proof. It is easy to verify that all arrows are homomorphisms (w.r.t. binary bilattice operators). The following table represents the correspondence of elements of these bilattices defined by homomorphisms:

$Multi-actions$	$L_1 \odot L_2 \odot L_3$	$Im\theta$	$L_1 \odot L_3$	$\mathcal{B}$
$[\_, \_, \_]$	$[\bot, t, \bot]$	$[\bot, t, \bot]$	$[\bot, \bot]$	$\bot$
$[\_, \uparrow_2, \_]$ $[\_, \_, \uparrow_3]$ $[\_, \uparrow_2, \uparrow_3]$	$[\bot, \mathsf{T}, \bot]$ $[\bot, t, f]$ $[\bot, \mathsf{T}, f]$	$[\bot, t, f]$	$[\bot, f]$	f
$[\uparrow_1, \_, \_]$	$[t, t, \bot]$	$[t, t, \bot]$	$[t, \bot]$	t
$[\uparrow_1, \uparrow_2, \_]$ $[\uparrow_1, \uparrow_2, \uparrow_3]$ $[\uparrow_1, \_, \uparrow_3]$	$[t, \mathsf{T}, \bot]$ $[t, \mathsf{T}, f]$ $[t, t, f]$	$[t, \mathsf{T}, f]$	$[t, f]$	T

Let prove, for example, that the isomorphism $\theta : \mathcal{B} \to Im\theta$ preserves negation and conflation: $\sim_B \theta(x) = \theta \sim \vartheta\theta(x) = \theta \sim id_B(x) = \theta(\sim x)$, and
$-_B\theta(x) = \theta - \vartheta\theta(x) = \theta - id_B(x) = \theta(-x)$.

4 Semantic reflection of the epistemic logic

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 atoms are defined as:
$\mathcal{A}_S = \{p(c_1, .., c_n) \mid p \in P_S,\ n = arity(p)\ and\ c_i \in \mathcal{T}_S\}$
The Herbrand base, H_P, is the set of all ground (i.e., variable free) atoms. A (ordinary) Herbrand interpretation is a many-valued mapping $I : H_P \to \mathcal{B}$. If P is a many-valued logic program with the Herbrand base H_P , then the ordering relations and operations in a bilattice $\mathcal{B}_4$ are propagated to the function space $\mathcal{B}_4^{H_P}$, that is the set of all Herbrand interpretations (functions), $I = v_B : H_P \to \mathcal{B}_4$, as follows:

Definition 2. *Ordering relations are defined on the Function space $\mathcal{B}_4^{H_P}$ pointwise, as follows: for any two Herbrand interpretations $v_B, w_B \in \mathcal{B}_4^{H_P}$*
1. $v_B \leq_t w_B$ if $v_B(A) \leq_t w_B(A)$ for all $A \in H_P$.
2. $v_B \leq_k w_B$ if $v_B(A) \leq_k w_B(A)$ for all $A \in H_P$.
3. $\sim v_B$ is the interpretation such that $(\sim v_B)(A) = \sim (v_B(A))$.
4. $-v_B$ is the interpretation such that $(-v_B)(A) = -(v_B(A))$.

It is straightforward [19] that this makes a function space $\mathcal{B}_4^{H_P}$ itself a complete infinitary distributive bilattice.

One of the key insights behind bilattices [11,12] 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. In [21] is proposed the following IB based approach to the 'object' 4-valued logic programming, which extends the definition given for a 3-valued logic programming [5]:

Definition 3. *Let $\mathcal{P}_B$ be the set of built-in predicates. The valuation, $v_B : H_P \rightarrow \mathcal{B}_4$, is extended to logic implication of a ground clause $p(\mathbf{c}) \leftarrow B$, where $B = B_1 \wedge .. \wedge B_n$, as follows:*

$$v_B(B \rightarrow p(\mathbf{c})) = t\,, \quad iff \qquad v_B(p(\mathbf{c})) \geq_t v_B(B) \; or \; (v_B(B) = \mathsf{T} \; and \; p \in \mathcal{P}_B)$$

Inconsistency acceptance: if $p \in \mathcal{P}_B$ is a built-in predicate, this clause is satisfied also when $v_B(p(\mathbf{c})) = f$ and $v_B(B) = \mathsf{T}$. This principle extends the previous definition of implication based only on truth ordering.
In order to obtain such many-valued definition, which generalize the 2-valued definition given above we will consider the conservative extensions of Lukasiewicz's and Kleene's strong 3-valued matrices (where third logic value $\perp$ is considered as unknown). So we obtain the following matrix, $f_{\leftarrow} : \mathcal{B} \times \mathcal{B} \rightarrow \mathcal{B}$, for implication ($\alpha = t$ and $\alpha = \perp$ for Lukasiewicz's and Kleene's case, respectively):

$\rightarrow$	t	$\perp$	T	f
t	t	$\perp$	T	f
$\perp$	t	α	T	$\perp$
T	t	t	t	t
f	t	t	t	t

For our purpose we assume the Lukasiewicz's extension, i.e. $\alpha = t$, in order to have a tautology $a \leftarrow a$ for any formula a, and also to guarantee the truth of a clause (implication) $p(\mathbf{c}) \leftarrow B$, whenever $v_B(p(\mathbf{c})) \geq_t v_B(B)$, as used in fixpoint semantics for 'immediate consequence operators'. Such conservative extensions are based on the following observation: the problem to study how the truth values should be propagated "across" implications can be restricted only to *true* implications (in fact we don't use implications when are not true, because the 'immediate consequence operator' derives new facts only for *true* clauses, i.e. when implication is true).
Example 1: The *built-in predicates* (ex, $=, \leq, \geq, ..$) may be used for integrity constraints: let $p(x, y)$ be a predicate and we define the key-constraint for attributes in x by $(y = z) \leftarrow p(x, y), p(x, z)$, where the atom $y = z$ is based on the built-in predicate $' ='$. Let consider a program : $\quad p(x, y) \leftarrow r(x, y)\,, \quad (y = z) \leftarrow p(x, y),\; p(x, z)$
where r is a source database relation with two tuples, $(a, b),\; (a, c)$, p is a virtual relation of this database with key constraint, and x, y, z are object variables. The built-in predicates have the same prefixed extension in *all* models of a logic program, and that their ground atoms are *true or false*. If we assume that, $r(a, b),\; r(a, c)$ are true, then such facts are mutually inconsistent for p because of key constraint ($b = c$ is false). Thus, only one of them may be true in any model of this logic program, for example $r(a, b)$. So, if we assign the 'possible' value T to $r(a, c)$ (or to both of them), we obtain that the clause $(b = c) \leftarrow p(a, b),\; p(a, c)$, thanks to the *inconsistency acceptance*, is satisfied.

Each *Herbrand interpretation* is a valuation. Valuations can be extended to maps from the set of all ground (variable free) formulas to $\mathcal{B}$ in the following way:

Definition 4. *Let $\mathcal{P}_S$ be the set of all predicate symbols ($\mathcal{P}_B \subseteq \mathcal{P}_S$ is a subset of built-in predicates), $\boldsymbol{e}$ the special (error) singleton, and $I : H_P \rightarrow \mathcal{B}$ be a many-valued Herbrand interpretation. A valuation I determines:*
1. A Generalized interpretation mapping $\mathcal{I} : \mathcal{P}_S \times \bigcup_{i\leq\omega} \mathcal{T}_0^i \rightarrow \mathcal{B}\bigcup\{\boldsymbol{e}\}$, such that for any $\boldsymbol{c} = (c_1, .., c_n) \in \mathcal{T}_0^n$, $\mathcal{I}(p, \boldsymbol{c}) = I(p(\boldsymbol{c}))$ iff $arity(p) = n$; $\boldsymbol{e}$ otherwise.
2. A unique valuation map, also denoted $v_{\mathcal{B}} : \mathcal{L} \rightarrow \mathcal{B}$, on the set of all ground formulas $\mathcal{L}$, according to the following conditions:
2.1. $v_{\mathcal{B}}(\sim X) = \sim v_{\mathcal{B}}(X)$
2.2. $v_{\mathcal{B}}(X \circledcirc Y) = v_{\mathcal{B}}(X) \circledcirc v_{\mathcal{B}}(Y)$, where $\circledcirc \in \{\wedge, \vee, \otimes, \oplus \leftarrow\}$
3. A truth assignment $u_B : \mathcal{L} \rightarrow \mathcal{B}$ will be called an extension of a truth assignment v_B if $u_B(\psi) \geq_k v_B(\psi)$ for all $\psi \in \mathcal{L}$. If u_B is an extension of v_B, we will write $u_B \geq_k v_B$.

The 'object' many-valued logic is based on four bilattice values which are *epistemic*. Sentences are to be marked with some of these bilattice logic values, according as to what the computer has been told; or, with only a slight metaphor, according to what it *believes or knows*. Of course these sentences *have* also Frege's ontological truth-values (true and false), independently of what the computer has been told: we want that the computer can use also these ontological 'meta' knowledge. Let, for example, the computer believes that the sentence p has a value T (possible); then the 'meta' sentence,"I (computer) believe that p has a possible value" is *ontologically true*. The many-valued encapsulation, defined as follows, is just the way to pass from the epistemic ('object') many-valued logic into ontological ('meta') 2-valued logic.
Such encapsulation is characterized by having capability for *semantic-reflection*: intuitively, for each predicate symbol we need some function which *reflects* its logic semantic over a domain Γ_U. Let introduce also the set of functional symbols κ_p over a domain Γ_U in our logical language in order to obtain an enriched logical language where we can encapsulate the 'object' (ordinary) many-valued logic programming. Such set of functional symbols will be derived from the following Bilattice-semantic mapping $\mathcal{K}$:

Definition 5. *A semantic-reflection is a mapping $\mathcal{K} : \mathcal{P}_S \rightarrow (\mathcal{B}\bigcup\{\boldsymbol{e}\})^{\bigcup_{i\leq\omega} \mathcal{T}_0^i}$, and we denote shortly $\kappa_p = \mathcal{K}(p) : \bigcup_{i\leq\omega} \mathcal{T}_0^i \rightarrow \mathcal{B}\bigcup\{\boldsymbol{e}\}$, $p \in \mathcal{P}_S$, such that for any $\boldsymbol{c} = (c_1, .., c_n) \in \mathcal{T}_0^n$, holds: $\kappa_p(\boldsymbol{c}) = \boldsymbol{e}$ iff $arity(p) \neq n$.*
If p is a built-in predicate, then a mapping κ_p is uniquely defined by: for any $\boldsymbol{c} \in \mathcal{T}_0^n$, $n = arity(p)$, holds that $\kappa_p(\boldsymbol{c}) = t$ if $p(\boldsymbol{c})$ is true; f otherwise.

5 Ontological encapsulation programming language

The many-valued ground atoms of a bilattice-based logical language $\mathcal{L}_{\mathcal{B}}$ can be transformed in 'encapsulated' atoms of a 2-valued logic in the following simple way: the original (many-valued) fact that the ground atom $A = p(c_1, .., c_n)$, of the n-ary predicate p, has an epistemic value $\alpha = \kappa_p(c_1, .., c_n)$ in $\mathcal{B}_4$, we transform in encapsulated atom $p^A(c_1, .., c_n, \alpha)$ with meaning "it is *true* that A has a value α". Indeed,

what we do is to *replace* the original n-ary predicate $p(x_1, .., x_n)$ with n+1-ary predicate $p^A(x_1, .., x_n, \alpha)$, with the added logic-attribute α. It is easy to verify that for any given many-valued valuation $v_{\mathcal{B}}$, every ground atom $p^A(c_1, .., c_n, \alpha)$ is ontologically true (when $\alpha = v_B(p(c_1, .., c_n))$) or false. Let EMV denote this new 2-valued encapsulation of many-valued logic *for logic programming*.

5.1 Syntax

We distinguish between what the reasoner believes in (at the *object* (epistemic many-valued sublanguage) level), and what is actually true or false in the real world (at the EMV ontological 'meta' level), thus, roughly, the 'meta' level is an (classic) encapsulation of the object level. Thus, we introduce the modal operator of encapsulation $\mathcal{E}$ as follows:

Definition 6. *Let P be an 'object' many-valued logic program with the set of predicate symbols P_S. The translation in the encapsulated syntax version in P^A is as follows:*
1. Each positive literal in P, $\mathcal{E}(p(x_1, .., x_n)) = p^A(x_1, .., x_n, \kappa_p(x_1, .., x_n))$;
2. Each negative literal in P, $\mathcal{E}(\sim p(x_1, .., x_n)) = p^A(x_1, .., x_n, \sim \kappa_p(x_1, .., x_n))$;
3. $\mathcal{E}(\phi \wedge \varphi) = \mathcal{E}(\phi) \wedge \mathcal{E}(\varphi)$;
4. $\mathcal{E}(\phi \vee \varphi) = \mathcal{E}(\phi) \vee \mathcal{E}(\varphi)$;
5. $\mathcal{E}(\phi \leftarrow \varphi) = \mathcal{E}(\phi) \leftarrow^A \mathcal{E}(\varphi)$ *, where $\leftarrow^A$ is a new syntax symbol for the implication at the encapsulated 2-valued 'meta' level.*
Thus, the obtained 'meta' program is equal to $P^A = \{\mathcal{E}(\phi) \mid \phi$ *is a clause in* $P\}$, *with the 2-valued Herbrand base* $H_P^A = \{\, p^A(c_1, .., c_n, \alpha) \mid p(c_1, .., c_n) \in H_P$ *and* $\alpha \in \mathcal{B}\}$

This embedding of the many-valued 'object' logic program P into a 2-valued 'meta' logic program P^A is an *ontological* embedding: views formulae of P as beliefs and interprets 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 for Moore's autoepistemic operator, for the encapsulation modal operator $\mathcal{E}$, $\mathcal{E}\phi$ is intended to capture the notion of, "I know that ϕ has a value $v_B(\phi)$ ", for a given valuation v_B of the 'object' logic program.
Let $\mathcal{L}$ be the set of all ground well-formed formulae defined by this Herbrand base H_P and bilattice operations (included many-valued implication $\leftarrow$ also), with $\mathcal{B} \subseteq \mathcal{L}$. We define the set of all well-formed encapsulated formulae by:
$\mathcal{L}^A =_{def} \{\mathcal{E}(\psi) \mid \psi \in \mathcal{L}\}$, so that $H_P^A \subseteq \mathcal{L}^A$, thus, we can extend operator $\mathcal{E}$ to all formulas in $\mathcal{L}$ (also to bilattice logic values, such that $\mathcal{E} : \mathcal{B} \rightarrow 2$), so, we obtain

Proposition 3 *The encapsulation operator $\mathcal{E}$ is :*
1. Nondeductive modal operator, such that, for any $\alpha \in \mathcal{B}$, $\mathcal{E}(\alpha) = t$ *if* $\alpha = t$; f *otherwise. It cannot be written in terms of the bilattice operations $\wedge, \vee, \otimes, \oplus$ and $\sim$.*
2. Homomorphism between the 'object' algebra $(\mathcal{L}, \wedge, \vee, \leftarrow)$ with carrier set of (positive and negative) literals, and 'meta' algebra $(\mathcal{L}^A, \wedge^A, \vee^A, \leftarrow^A)$, where $\wedge^A, \vee^A$ are 2-valued reductions of bilattice meet and join, respectively, denoted by $\wedge, \vee$ also.

5.2 Semantics

The modal operator $\mathcal{E}$ is more selective than Moore's modal operator M (which returs the truth also when its argument has a possible value). In fact $M(\alpha) = \mathcal{E}(\alpha \vee \bot)$.
Notice, that with the transformation of the original 'object' logic program P into its annotated 'meta' version program P^A we obtain *always positive* consistent logic program.
A Herbrand interpretation of P^A is a 2-valued mapping $I^A : H_P^A \rightarrow \mathbf{2}$. We denote by $\mathbf{2}^{H_P^A}$ the set of all a-interpretations (functions) from H_P^A into **2**, and by $\mathcal{B}^{H_P}$ the set of all *consistent* Herbrand many-valued interpretations, from H_P to the bilattice $\mathcal{B}$. The meaning of the *encapsulation* of this 'object' logic program P into this 'meta' logic program P^A is fixed into the kind of interpretation to give to such new introduced functional symbols $\kappa_p = \mathcal{K}(p)$: in fact *we want* [21] that they reflect (encapsulate) the semantics of the 'object' level logic program P.

Definition 7. *(Satisfaction) The **encapsulation** of an epistemic 'object' logic program P into an 'meta' program P^A means that, for any* consistent *many-valued Herbrand interpretation $I \in \mathcal{B}^{H_P}$ and its extension $v_B : \mathcal{L} \rightarrow \mathcal{B}$, the function symbols $\kappa_p = \mathcal{K}(p),\ p \in P_S$ reflects this semantics (is compatible to it), i.e.*
for any tuple $\mathbf{c} \in \mathcal{T}_0^{arity(p)}$, $\kappa_p(\mathbf{c}) = I(p(\mathbf{c}))$.
So, we obtain a mapping, $\Theta : \mathcal{B}^{H_P} \rightarrow \mathbf{2}^{H_P^A}$, such that $I^A = \Theta(I) \in \mathbf{2}^{H_P^A}$ with: for any ground atom $p(\mathbf{c})$, $I^A(\mathcal{E}(p(\mathbf{c}))) = t$, if $\kappa_p(\mathbf{c}) = I(p(\mathbf{c}))$; f otherwise.
Let g be a variable assignment which assigns values from Γ_U to object variables. We extent it to atoms with variables, so that $g(\mathcal{E}(p(x_1, .., x_n))) = \mathcal{E}(p(g(x_1), .., g(x_n)))$, and to all formulas in the usual way: ψ/g denotes a ground formula obtained from ψ by assignment g, then
1. $I^A \models_g \mathcal{E}(p(x_1, .., x_n))$ iff $\kappa_p((g(x_1), .., g(x_n))) = I(p(g(x_1), .., g(x_n)))$.
$I^A \models_g \mathcal{E}(\sim p(x_1, .., x_n))$ iff $\sim \kappa_p((g(x_1), .., g(x_n))) = I(p(g(x_1), .., g(x_n)))$.
2. $I^A \models_g \mathcal{E}(\phi \wedge \psi)$ iff $I^A \models_g \mathcal{E}(\phi)$ and $I^A \models_g \mathcal{E}(\psi)$.
3. $I^A \models_g \mathcal{E}(\phi \vee \psi)$ iff $I^A \models_g \mathcal{E}(\phi)$ or $I^A \models_g \mathcal{E}(\psi)$.
4. $I^A \models_g \mathcal{E}(\phi \leftarrow \psi)$ iff $v_B(\phi/g \leftarrow \psi/g)$ is true.

Notice that in this semantics the 'meta' implication $\leftarrow^A$, in $\mathcal{E}(\phi) \leftarrow^A \mathcal{E}(\psi) = \mathcal{E}(\phi \leftarrow \psi)$, is based on the 'object' epistemic many-valued implication $\leftarrow$ (which *is not classical*, i.e., $\phi \leftarrow \psi \neq \phi \vee \sim \psi$) and determines how the logical value of a body of clause "propagates" to its head.

Theorem 1 *The semantics of encapsulation $\mathcal{E}$ is obtained by identifying the semantic-reflection with the λ-abstraction of Generalized Herbrand interpretation, $\mathcal{K} = \lambda\mathcal{I}$, so that the semantics of many-valued logic programs can be determined by $\mathcal{I}$ (at 'object' level) or, equivalently, by its reflection $\mathcal{K}$ (at encapsulated or 'meta' level).*

Proof. From $\mathcal{K} = \lambda\mathcal{I}$ we obtain that for any $p(\mathbf{c}) \in H_P$ holds $I(p(\mathbf{c})) = \mathcal{I}(p, \mathbf{c}) = \lambda\mathcal{I}(p)(\mathbf{c}) = \mathcal{K}(p)(\mathbf{c}) = \kappa_p(\mathbf{c})$, what is the semantic of encapsulation.

We can consider the λ-abstraction of Generalized Herbrand interpretation as an epistemic semantics, because, given a Herbrand (epistemic) interpretation $I : H_P \rightarrow \mathcal{B}$,

then for any predicate symbol p and constant $\mathbf{c} \in \mathcal{T}_0^{arity(p)}$, holds $\lambda\mathcal{I}(p)(\mathbf{c}) = I(p(\mathbf{c}))$. Then the semantic of encapsulation may be defined as follows:

" ontological semantic-reflection $\equiv$ epistemic semantics", that is, $\mathcal{K} = \lambda\mathcal{I}$.

Recently, in [22], this semantics is used to give a coalgebraic semantics for logic programs. Notice that at 'meta' (ontological) level (differently from $\wedge, \vee$, which are classic 2-value boolean operators), the semantics for 'meta' implication operator, $I^A \models_g \mathcal{E}(\phi \leftarrow \psi)$, is not defined on $I^A \models_g \mathcal{E}(\phi)$ and $I^A \models_g \mathcal{E}(\psi)$. For example, let $I^A \models_g \mathcal{E}(p(\mathbf{c}))$ and $I^A \models_g \mathcal{E}(q(\mathbf{d}))$, with $\kappa_p(\mathbf{c}) = f$ and $\kappa_q(\mathbf{d}) = t$: then $p(\mathbf{c}) \leftarrow q(\mathbf{d})$ is false and, consequently, does not hold $I^A \models \mathcal{E}(p(\mathbf{c}) \leftarrow q(\mathbf{d}))$.

Proposition 4 *$I^A \models_g \mathcal{E}(\phi) \leftarrow^A \mathcal{E}(\psi)$ implies $I^A \models_g \mathcal{E}(\phi)$ and $I^A \models_g \mathcal{E}(\psi)$, but not viceversa. The truth of $\mathcal{E}(\phi/g)$ and $\mathcal{E}(\psi/g)$ are necessary but not sufficient conditions for the truth of $\mathcal{E}(\phi/g) \leftarrow^A \mathcal{E}(\psi/g)$.*

More over $\leftarrow^A$ has a *constructivistic* viewpoint (notice that the implication $\leftarrow^A$ is satisfied when the body and the head of such clause are *true*, while in the 'object' logic program such clause may be satisfied when their body and the head *are not true* also). Thus, by encapsulation of a many-valued 'object' logic program into a 2-valued 'meta' logic program we obtain a constructive logic program: in each clause we derive from the true facts in its body other new true facts.
Following the standard definitions, we say that an interpretation I^A, of a program P^A, is a *model* of a P^A if and only if every clause of P^A is satisfied in I^A. In this way we define a *model theoretic* semantics for encapsulated logic programs.
A set of formulas S, of encapsulated logic EMV, *logically entails* a formula ϕ, denoted $S \models \phi$, if and only if every model of S is also a model of ϕ.

6 Conclusion

We have presented a programming logic capable of handling inconsistent beliefs and based on the 4-valued Belnap's bilattice, which has clear model theory. In the process of the encapsulation we distinguish two levels: the 'object' many-valued level of ordinary logic programs with epistemic negation based on a bilattice operators, and the encapsulated or 'meta' logic programs. In this approach, 'inconsistent' logic program (which minimal stable models contain at least an 'inconsistent' ground atom) at object level is classic consistent logic program at 'meta' level also. In such abstraction we obtained a kind of a minimal Constructivistic Logic where fixpoint 'immediate consequence' operator is always continuous, and which is *computationally equivalent* to the standard Fitting's fixpoint semantics. Following this approach we are able to define a unique many-valued Herbrand model for databases with inconsistencies based on the fixpoint of a monotonic (w.r.t. knowledge ordering) immediate consequence operator, and the inference closure for many-valued logic programming also.
This research is partially supported by the project NoE INTEROP-IST-508011 and the project SEWASIE-IST-2001-3425. The autor wishes to thank Tiziana Catarci and Maurizio Lenzerini for their support.

References

1. M.Gelfond and V.Lifshitz, "The stable model semantics for logic programming," *In Proc. of the Fifth Logic Programming Symposium, Cambridge, MA. MIT Press*, pp. 1070–1080, 1988.
2. K.Fine, "The justification of negation as failure," *In Logic, Methodology and Philosophy of Science VIII, Amsterdam, North-Holland*, pp. 263–301, 1989.
3. M.C.Fitting, "A kripke/kleene semantics for logic programs," *Journal of Logic Programming 2*, pp. 295–312, 1985.
4. K.Kunen, "Negation in logic programming," *Journal of Logic Programming 4*, pp. 289–308, 1987.
5. 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.
6. T.Przymusinski, "Well-founded semantics coincides with thre-valued stable-semantics," *Fundamenta Informaticae 13*, pp. 445–463, 1990.
7. 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.
8. 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.
9. M.Kifer and E.L.Lozinskii, "A logic for reasoning with inconsistency," *Journal of Automated reasoning 9(2)*, pp. 179–215, 1992.
10. 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.
11. M.Ginsberg, "Multivalued logics: A uniform approach to reasoning in artificial intelligence," *Computational Intelligence, vol.4*, pp. 265–316, 1988.
12. M.C.Fitting, "Billatices and the semantics of logic programming," *Journal of Logic Programming,11*, pp. 91–116, 1991.
13. M.H.van Emden, "Quantitative deduction and its fixpoint theory," *Journal of Logic Programming,4,1*, pp. 37–53, 1986.
14. S.Morishita, "A unified approach to semantics of multi-valued logic programs," *Tech. Report RT 5006, IBM Tokyo*, 1990.
15. 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.
16. A.Calì, D.Calvanese, G.De Giacomo, and M.Lenzerini, "Data integration under integrity constraints," in *Proc. of the 14th Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, 2002, pp. 262–279.
17. Z. Majkić, "Fixpoint semantic for query answering in data integration systems," *AGP03 - 8.th Joint Conference on Declarative Programming, Reggio Calabria*, pp. 135–146, 2003.
18. 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.
19. M.C.Fitting, "Billatices are nice things," *Proceedings of Conference on Self-Reference, Copenhagen*, 2002.
20. M.C.Fitting, "Kleene's three valued logics and their children," *Fundamenta Informaticae*, vol. 26, pp. 113–131, 1994.
21. Z. Majkić, "Two-valued encapsulation of many-valued logic programming," *Technical Report, University 'La Sapienza', Roma, in http://www.dis.uniroma1.it/ $\sim$ majkic/.*, 2003.
22. Z. Majkić, "Coalgebraic semantics for logic programming," *18th Worshop on (Constraint) Logic Programming, WLP 2004, March 04-06, Berlin, Germany*, 2004.