Four-Valued Semantics for Deductive Databases

Dominique Laurent, Nicolas Spyratos

Four-Valued Semantics for Deductive Databases

Dominique Laurent · Nicolas Spyratos

Abstract In this paper, we introduce a novel approach to deductive databases meant to take into account the needs of current applications in the area of data integration. To this end, we extend the formalism of standard deductive databases to the context of Four-valued logic so as to account for *unknown*, *inconsistent*, *true* or *false* information under the open world assumption. In our approach, a database is a pair (E, R) where E is the extension and R the set of rules. The extension is a set of pairs of the form $\langle \varphi, \mathtt{v} \rangle$ where φ is a fact and $\mathtt{v}$ is a value that can be true, inconsistent or false - but not unknown (that is, unknown facts are not stored in the database). The rules follow the form of standard Datalog^{neg} rules but, contrary to standard rules, their head may be a negative atom.
Our main contributions are as follows: (*i*) we give an expression of first-degree entailment in terms of other connectors and exhibit a functionally complete set of basic connectors not involving first-degree entailment, (*ii*) we define a new operator for handling our new type of rules and show that this operator is monotonic and continuous, thus providing an effective way for defining and computing database semantics, and (*iii*) we argue that our framework allows for the definition of a new type of updates that can be used in most standard data integration applications.

Keywords Open World Assumption . Multi-valued logic .
Inconsistent database . Deductive database . Update Semantics

Dominique Laurent
ETIS Laboratory - ENSEA, CY Cergy Paris University, CNRS
F-95000 Cergy-Pontoise, France
dominique.laurent@u-cergy.fr

Nicolas Spyratos
LISN Laboratory - University Paris-Saclay, CNRS
F-91405 Orsay, France
nicolas.spyratos@lri.fr

Acknowledgment: Work conducted while the second author was visiting at FORTH Institute of Computer Science, Crete, Greece (https://www.ics.forth.gr/)

1 Introduction

In this paper, we present a novel approach meant to take into account the needs of many current applications, specifically in the domain of data integration. Our purpose is to extend the concept of deductive databases [1,2] to the context of Four-valued logic [3], a formalism known to be suitable for data integration, as it allows to deal with *unknown*, *inconsistent*, *true* or *false* information. We begin by illustrating our approach through an example used as our running example throughout the paper.

Running Example. Our example concerns the storage of bags of rice grains, considering two important factors that (among others) influence the design and development of optimum storage, namely color and humidity of the rice grains [4].

We assume that each bag is tested for the color and humidity of its rice grains in two different sites, first just before leaving the rice farm and then just before entering the warehouse. The outcomes of these tests can be: humid or not humid (with respect to a humidity threshold); and white or not white (with respect to a color threshold). Based on these outputs, the following actions are taken:

- If the grains are not humid and white then store the bags in the warehouse.
- If the grains are humid then do not store the bags but cure the grains.
- If the grains are not white then do not store the bags but analyze further.

We assume that the tests are conducted by sensors: two sensors at the rice farm, one for humidity, denoted H_1, and one for color denoted W_1; and two sensors at the warehouse denoted H_2 and W_2. We also assume that, during a test, if the sensor is functioning then it returns a Boolean value (true or false), otherwise it returns no value. Under these assumptions, one of the following cases can appear for the sensors testing humidity (and similarly for the sensors testing color):

1. The two sensors return the same value.
2. The two sensors return *different* values.
3. Only one of the two sensors returns a value.
4. Neither of the two sensors returns a value.

In this setting, let $Humid(ID)$, denote the humidity state or 'value' of a bag with identifier ID. Then the question is: what value should we assign to $Humid(ID)$ in each of the four cases above? In our formalism, we answer this question by 'integrating' the outputs of H_1 and H_2 as follows (and similarly for the outputs of W_1 and W_2):

1. $Humid(ID)$ is set to the common value returned by the sensors.
2. $Humid(ID)$ is set to *inconsistent*, to mean that the sensors returned different values.
3. $Humid(ID)$ is set to the value returned by the sensor which returned a value.
4. $Humid(ID)$ is set to *unknown*, to mean that neither of the two sensors returned a value.

As our example shows, we clearly need more than the standard truth values `True` and `False`, to express the cases 2 and 4 above. It will be seen that the Four-valued logic introduced in [3] provides the right formalism as it provides the additional truth values needed and also appropriate connectors to work with these additional

truth values. For instance, using a connector denoted by $\oplus$ we can express all four cases above in a single expression: $Humid(ID) = H_1(ID) \oplus H_2(ID)$.

The database is a pair (E, R) where E collects the sensor outputs and where R is a set of rules describing how to integrate these outputs and how to treat the bags based on the integrated values. Formally, the elements of E are pairs of the form $\langle \varphi, \mathtt{v} \rangle$ to represent the output of one sensor about a bag recognized by its identifier. In such pair φ is a fact regarding the humidity or the color of a bag and $\mathtt{v}$ is its associated truth value. The rules expressing the integration of the sensor outputs and the conditions regarding the storage of the bags are as follows:

$\rho_1 : Humid(x) \leftarrow H_1(x) \oplus H_2(x)$
$\rho_2 : White(x) \leftarrow W_1(x) \oplus W_2(x)$
$\rho_3 : Store(x) \leftarrow \neg Humid, (x) \wedge White(x)$
$\rho_4 : \neg Store(x) \leftarrow Humid(x)$
$\rho_5 : Cure(x) \leftarrow Humid(x)$
$\rho_6 : \neg Store(x) \leftarrow \neg White(x)$
$\rho_7 : New\_test(x) \leftarrow \neg White(x)$

Although the rules above roughly look like standard Datalog rules with negation, the following basic differences have to be noticed:

1. The body of a rule is not restricted to be a conjunction of literals; in fact we allow all available connectors to occur in the body of a rule.
2. The head of a rule is not restricted to be an atom: negative literals are allowed, at the cost of generating contradictory facts.
3. Contradictions are allowed in database semantics and treated as such, in the context of the Four-valued semantics introduced in [3].

To illustrate how our approach deals with such rules, we first give a rough overview of the basic notions used in our approach. First, in Four-valued logic, four truth values are considered, namely `t`, `b`, `n` and `f`, standing respectively for *true*, *inconsistent*, *unknown*[1] and *false*.

In this context the pieces of information to be stored in the database extension are pairs of the form $\langle \varphi, \mathtt{v} \rangle$ where φ is a fact (i.e. an atom with no variable) and $\mathtt{v}$ is one of the four truth values just mentioned. By such a pair, which we call *valuated pair* or v-pair for short, we mean that *'φ has truth value* `v`*'*. Moreover, we make the intuitively appealing convention that unknown facts are not stored, meaning that the database extension can *not* contain a v-pair of the form $\langle \varphi, \mathtt{n} \rangle$. We emphasize that, contrary to most database approaches in which only *true* pieces of information are stored, our approach allows to store *true*, *false* or even *inconsistent* pieces of information.

Continuing with our example, assume there are three rice bags with identifiers 101, 202 and 303 for which the following sensor outputs and corresponding v-pairs are stored in the database:

Regarding bag 101*:* H_1 and H_2 both return `False`; this results in storing the two v-pairs $\langle H_1(101), \mathtt{f} \rangle$ and $\langle H_2(101), \mathtt{f} \rangle$ in the database extension. W_1 returns `True` but W_2 returns no value; this results in storing the pair $\langle W_1(101), \mathtt{t} \rangle$ in the database extension.

Regarding bag 202*:* H_2 returns `True` and H_1 returns no value; this results in storing the v-pair $\langle H_2(202), \mathtt{t} \rangle$ in the database extension. W_1 returns `False` while W_2 returns `true`; this results in storing the two pairs $\langle W_1(202), \mathtt{f} \rangle$ and $\langle W_2(202), \mathtt{t} \rangle$ in the database extension.

[1] The intuition explaining the notation `b` and `n` will be clarified later in this paper.

Regarding bag 303*:* H_1 and H_2 both return no value, W_1 returns `False` and W_2 returns no value; this results in storing the pair $\langle W_1(303), \mathtt{f}\rangle$ in the database extension.

Roughly speaking, given a set S of v-pairs, applying a rule ρ is achieved as follows: for every instantiation of ρ denoted $inst(\rho)$, the truth value of the body of $inst(\rho)$ is computed against S, and if this truth value is `t` or `b` then this truth value is assigned to the head of the $inst(\rho)$. Moreover, as more than one rule head may involve the same fact, in case of conflicting assignment, we apply the integration statements as done for the sensors. We illustrate this processing below.

1. At the first step, the only rules that apply are ρ_1 and ρ_2.
   - Based on the v-pairs $\langle H_1(101), \mathtt{f}\rangle$ and $\langle H_2(101), \mathtt{f}\rangle$, ρ_1 generates the v-pair $\langle Humid(101), \mathtt{f}\rangle$ stating that the grains in bag 101 are not humid.
     As for identifier 202, since the output of H_1 is missing, we consider the (non-stored) v-pair $\langle H_1(202), \mathtt{n}\rangle$, which combined by $\oplus$ with the stored v-pair $\langle H_2(202), \mathtt{t}\rangle$ generates $\langle Humid(202), \mathtt{t}\rangle$ stating that the grains in bag 202 are humid.
     As for identifier 303, since both H_1 and H_2 no value, ρ_1 generates no v-pair involving $Humid(303)$, meaning that the humidity of the grains in the bag 303 is unknown.
   - As for $White(101)$, since W_2 returns no value, ρ_2 generates the v-pair $\langle White(101), \mathtt{t}\rangle$ stating that the grains in bag 101 are white.
     As for $White(202)$, we notice that W_1 and W_2 disagree. In this case, ρ_2 generates the v-pair $\langle White(202), \mathtt{b}\rangle$, meaning that the fact $White(202)$ is inconsistent, thus that the color of the grains in bag 202 cannot be decided.
     As for $White(303)$, since W_2 returns no value, ρ_2 generates the v-pair $\langle White(303), \mathtt{f}\rangle$, meaning that the grains in bag 303 cannot be considered white.
2. The next step is based on the v-pairs earlier generated, namely: $\langle Humid(101), \mathtt{f}\rangle$, $\langle Humid(202), \mathtt{t}\rangle$, $\langle White(101), \mathtt{t}\rangle$, $\langle White(202), \mathtt{b}\rangle$ and $\langle White(303), \mathtt{f}\rangle$. The rules $\rho_3 \ldots \rho_7$ apply as follows:
   - Based on $\langle Humid(101), \mathtt{f}\rangle$ and $\langle White(101), \mathtt{t}\rangle$, ρ_3 generates the v-pair $\langle Store(101), \mathtt{t}\rangle$. Considering $\langle Humid(202), \mathtt{t}\rangle$ and $\langle White(202), \mathtt{b}\rangle$, since the conjunction of the body is false, ρ_3 does not apply. Since $Humid(303)$ is unknown and $White(303)$ is false, the conjunction of the body is false, entailing that ρ_3 does not apply.
   - Since $Humid(101)$ is not true, ρ_4 does not apply. Since $Humid(202)$ is true, ρ_4 generates $\langle Store(202), \mathtt{f}\rangle$. Since $Humid(303)$ is unknown, ρ_4 does not apply.
   - As above, since $Humid(101)$ is not true, ρ_5 does not apply, but ρ_5 generates $\langle Cure(202), \mathtt{t}\rangle$ because $Humid(202)$ is true.
   - Similarly, since $White(101)$ is not false, ρ_6 and ρ_7 do not apply. Since $White(202)$ is *inconsistent*, ρ_6 and ρ_7 generate respectively $\langle Store(202), \mathtt{b}\rangle$ and $\langle New\_test(202), \mathtt{b}\rangle$. Moreover, since $White(303)$ is false, ρ_6 and ρ_7 generate respectively $\langle Store(303), \mathtt{f}\rangle$ and $\langle New\_test(303), \mathtt{t}\rangle$.

   After applying the rules, conflicting v-pairs involving $Store(202)$ appear, because $Store(202)$ has been found *false* by ρ_4 and *inconsistent* by ρ_6. In this case, we integrate these different truth values in much the same way as we did for

the sensor outputs, stating that $Store(202)$ should be inconsistent. Therefore, the v-pair $\langle Store(202), \mathtt{f}\rangle$ is removed from the result of this step.
3. As no further v-pair can be generated by the rules based on the v-pairs generated in the previous steps, the processing stops and returns the set of all these v-pairs, which added to the database extension constitutes what we call the *database semantics*.

The obtained database semantics is therefore the set of the following v-pairs:

$\langle H_1(101), \mathtt{f}\rangle$, $\langle H_2(101), \mathtt{f}\rangle$, $\langle H_2(202), \mathtt{t}\rangle$,
$\langle W_1(101), \mathtt{t}\rangle$, $\langle W_1(202), \mathtt{f}\rangle$, $\langle W_2(202), \mathtt{t}\rangle$, $\langle W_1(303), \mathtt{f}\rangle$,
$\langle Humid(101), \mathtt{f}\rangle$, $\langle Humid(202), \mathtt{t}\rangle$,
$\langle White(101), \mathtt{t}\rangle$, $\langle White(202), \mathtt{b}\rangle$, $\langle White(303), \mathtt{f}\rangle$,
$\langle Store(101), \mathtt{t}\rangle$, $\langle Store(202), \mathtt{b}\rangle$, $\langle Store(303), \mathtt{f}\rangle$,
$\langle Cure(202), \mathtt{t}\rangle$, $\langle New\_test(202), \mathtt{b}\rangle$, $\langle New\_test(303), \mathtt{t}\rangle$.

It is shown in this paper that the computation just described in an informal way is sound and its relationship with other related approaches is investigated. Moreover, some basic properties of the underlying Four-valued logic are stated, and among them this example raises the following question: could the rules ρ_4 and ρ_6 be replaced by the single rule $\rho_{46} : \neg Store(x) \leftarrow Humid(x) \vee \neg White(x)$? Whereas this question is answered positively in standard approaches to Datalog databases ([1, 2]) and in the Four-valued approach of [5], we argue that this replacement raises some issues. □

This work is an extension of that in [6] where rule bodies are restricted to be conjunctions. The main contributions of this paper are as follows:

1. We show that FDE (*First Degree Entailment*) implication, one of the standard implications in Four-valued logic, can be expressed in terms of the usual connectors.
2. We exhibit a functionally complete set of basic connectors *not* involving FDE implication, contrary to the results in [7].
3. We generalize the rules by allowing negative literals in their heads and connectors other than negation, conjunction and disjunction in their bodies.
4. We define a new immediate consequence operator for handling such rules, and we show that this operator is monotonic and continuous, thus providing an effective way for defining and computing database semantics.
5. We argue that our context allows for the definition of a new type of updates that can be used in data integration applications. Notice that to the best of our knowledge, the problem of database updating in a Four-valued logic framework has never been addressed in the literature.

The paper is organized as follows: In Section 2 we review the formalism related to Four-valued logic and we address the first two issues mentioned above. Section 3 is devoted to the definitions of the syntax and the semantics of databases in the context of Four-valued logic. In Section 4, we define two types of updates, one standard and another one related to data integration. Then, in Section 5 we review some of the approaches related to our work that can be found in the literature. Section 6 provides an overview of our approach and suggests research issues that we are currently investigating or that we intend to investigate in the next future.

2 Background: Four-Valued Logic

2.1 Basics of Four-Valued Logic

Four-valued logic was introduced by Belnap in [3], who argued that this formalism could be of interest when integrating data from various data sources. To this end, denoting by $\mathtt{t}$, $\mathtt{b}$, $\mathtt{n}$ and $\mathtt{f}$ the four truth values, the usual connectives $\neg$, $\vee$ and $\wedge$ have been defined as shown in Figure 1. An important feature of this Four-valued logic is that it allows to compare truth values according to two partial orderings, known as *truth ordering* and *knowledge ordering*, respectively denoted by $\preceq_t$ and $\preceq_k$ and defined by:

$$\mathtt{n} \preceq_k \mathtt{t} \preceq_k \mathtt{b} \ ; \ \mathtt{n} \preceq_k \mathtt{f} \preceq_k \mathtt{b} \qquad \text{and} \qquad \mathtt{f} \preceq_t \mathtt{n} \preceq_t \mathtt{t} \ ; \ \mathtt{f} \preceq_t \mathtt{b} \preceq_t \mathtt{t}.$$

To explain the choice of $\mathtt{b}$ and $\mathtt{n}$ as notation for *inconsistent* and *unknown*, let $\mathcal{V} = \{\mathtt{True}, \mathtt{False}\}$ be the set of the usual truth values. The four truth values in Four-valued logic can then be thought of as corresponding to the elements in the power set of $\mathcal{V}$, by associating respectively $\emptyset$, $\{\mathtt{False}\}$, $\{\mathtt{True}\}$, $\{\mathtt{True}, \mathtt{False}\}$ with $\mathtt{n}$, $\mathtt{f}$, $\mathtt{t}$, $\mathtt{b}$. Then the notation $\mathtt{n}$ and $\mathtt{b}$ can be read respectively as *none* and *both*. Notice also that, under this association, the ordering $\preceq_k$, the connectors $\oplus$ and $\otimes$ are respectively nothing but the restriction to the power set of $\mathcal{V}$ of set theoretic inclusion, union and intersection.

As in standard two-valued logic, conjunction (respectively disjunction) corresponds to minimum (respectively maximum) truth value, when considering the truth ordering. It has also been shown in [3,5] that the set $\{\mathtt{t}, \mathtt{b}, \mathtt{n}, \mathtt{f}\}$ equipped with these two orderings has a distributive bi-lattice structure, where the minimum and maximum with respect to $\preceq_k$ are denoted by $\otimes$ and $\oplus$, respectively.

Not surprisingly, it should be emphasized that in this Four-valued logic some basic properties holding in standard logic do not hold. For example, Figure 1 shows that formulas of the form $\Phi \vee \neg\Phi$ are not always true, independently from the truth value of Φ. More importantly, it has been argued in [7–9] that defining the implication $\Phi_1 \Rightarrow \Phi_2$ by $\neg\Phi_1 \vee \Phi_2$, is problematic.

To see this, we consider as in [3,7–9], that $\mathtt{t}$ and $\mathtt{b}$ are the two *designated truth values*, because as mentioned above, these truth values are the only ones corresponding to sets containing $\mathtt{True}$. As a consequence, a formula Φ is said to be *valid* if its truth value is designated, *i.e.,* either $\mathtt{t}$ or $\mathtt{b}$.

As argued in [7–9], $\Rightarrow$ does not satisfy the deduction theorem, because the formula Φ defined by $(\Phi_1 \wedge (\Phi_1 \Rightarrow \Phi_2)) \Rightarrow \Phi_2$ is *not valid for every truth value assignment*. Indeed based on Figure 2, for every assignment v such that $v(\Phi_1) = \mathtt{n}$ and $v(\Phi_2) = \mathtt{f}$, we have $v(\Phi_1 \Rightarrow \Phi_2) = \mathtt{n}$ and thus, $v(\Phi) = \mathtt{n}$. As a consequence, we discard $\Rightarrow$ as the implication providing semantics to our rules.

Among the various implications introduced in the literature, *First Degree Entailment* implication, or FDE implication, denoted hereafter by $\rightarrow$ ([7,8]) is the most popular. We also mention another implication introduced in [9] and denoted hereafter by $\hookrightarrow$. Each of these implications is associated with another implication, denoted by $\overset{*}{\rightarrow}$ and $\overset{*}{\hookrightarrow}$ whose role is explained next. The truth tables of all these implications are shown in Figure 2.

Recall from [7] (Corollary 9) that $\rightarrow$, is defined 'from scratch' in the sense that it cannot be expressed using the other standard connectives $\neg$, $\vee$ and $\wedge$. As we shall see shortly we can provide an expression of $\rightarrow$ involving standard connectors

φ	$\neg\varphi$
t	f
b	b
n	n
f	t

φ	$\not\sim\varphi$
t	b
b	t
n	f
f	n

φ	$\sim\varphi$
t	f
b	n
n	b
f	t

$\vee$	t	b	n	f
t	t	t	t	t
b	t	b	t	b
n	t	t	n	n
f	t	b	n	f

$\wedge$	t	b	n	f
t	t	b	n	f
b	b	b	f	f
n	n	f	n	f
f	f	f	f	f

$\oplus$	t	b	n	f
t	t	b	t	b
b	b	b	b	b
n	t	b	n	f
f	b	b	f	f

$\otimes$	t	b	n	f
t	t	t	n	n
b	t	b	n	f
n	n	n	n	n
f	n	f	n	f

Fig. 1 Truth tables of basic connectors

in the formalism of [9]. It is also important to notice that as shown in [9], $\Phi_1 \hookrightarrow \Phi_2$ is defined by $\sim \Phi_1 \vee \Phi_2$, where $\sim$ is a complement operator whose truth table is shown in Figure 1.

Moreover, since $\Phi_1 \rightarrow \Phi_2$ and $\neg\Phi_2 \rightarrow \neg\Phi_1$ are not equivalent, the implication $\Phi_1 \overset{*}{\rightarrow} \Phi_2$ is introduced in [7,8] as a shorthand for $(\Phi_1 \rightarrow \Phi_2) \wedge (\neg\Phi_2 \rightarrow \neg\Phi_1)$. As a similar situation holds regarding $\hookrightarrow$, $\Phi_1 \overset{*}{\hookrightarrow} \Phi_2$ is defined in [9] as $(\Phi_1 \hookrightarrow \Phi_2) \wedge (\neg\Phi_2 \hookrightarrow \neg\Phi_1)$.

In an attempt to compare these implications, we notice that, contrary to $\Rightarrow$, the formula Φ defined by $(\Phi_1 \wedge (\Phi_1 \rightsquigarrow \Phi_2)) \rightsquigarrow \Phi_2$ is valid when replacing $\rightsquigarrow$ with one of the implications $\rightarrow$, $\hookrightarrow$, $\overset{*}{\rightarrow}$ or $\overset{*}{\hookrightarrow}$. It is also interesting to see that when merging the truth values t and b (respectively f and n) into a single value, say TRUE (respectively FALSE), the corresponding truth tables of $\rightarrow$ and $\hookrightarrow$ are that of the standard implication, while this is not the case for $\Rightarrow$, $\overset{*}{\rightarrow}$ and $\overset{*}{\hookrightarrow}$. This explains why we discard these three implications. However, the choice between $\rightarrow$ and $\hookrightarrow$ is not easy for the following reasons:

- In [7,8], it is argued that, similarly to two-valued implication, $\rightarrow$ satisfies the property that $v(\Phi_1 \rightarrow \Phi_2) = v(\Phi_2)$ whenever $v(\Phi_1)$ is designated. However, $\rightarrow$ does not satisfy the properties of $\hookrightarrow$ given below.
- Although $\hookrightarrow$ does not satisfy the above property, it is argued in [9] that, similarly to two-valued implication, $\hookrightarrow$ satisfies the property that $v(\Phi_1) \preceq_t v(\Phi_2)$ if and only if $v(\Phi_1 \hookrightarrow \Phi_2) = \mathtt{t}$.

We draw attention on that none of these two implications satisfies all intuitively appealing properties that standard two-valued implication satisfies, among which contraposition is an example.

Looking at the truth tables of the two implications $\rightarrow$ and $\hookrightarrow$, when the left hand side is valid in S, it is necessary that the right hand side be also valid in order to make the implication valid. More precisely, if Φ_1 is valid, the implications $\Phi_1 \rightarrow \Phi_2$ and $\Phi_1 \hookrightarrow \Phi_2$ are valid in S for any truth assignment v such that:

– $v(\Phi_1) = \mathtt{t}$ and $v(\Phi_2) = \mathtt{t}$ or $v(\Phi_2) = \mathtt{b}$,

$\Rightarrow$	t	b	n	f
t	t	b	n	f
b	t	b	t	b
n	t	t	n	n
f	t	t	t	t

$\rightarrow$	t	b	n	f
t	t	b	n	f
b	t	b	n	f
n	t	t	t	t
f	t	t	t	t

$\hookrightarrow$	t	b	n	f
t	t	b	n	f
b	t	t	n	n
n	t	b	t	b
f	t	t	t	t

$\overset{*}{\rightarrow}$	t	b	n	f
t	t	f	n	f
b	t	b	n	f
n	t	n	t	n
f	t	t	t	t

$\overset{*}{\hookrightarrow}$	t	b	n	f
t	t	f	f	f
b	t	t	f	f
n	t	f	t	f
f	t	t	t	t

Fig. 2 Truth tables of implications

ϕ	$\mathbf{T}\phi$
t	t
b	f
n	f
f	f

ϕ	$\mathbf{B}\phi$
t	f
b	t
n	f
f	f

ϕ	$\mathbf{N}\phi$
t	f
b	f
n	t
f	f

ϕ	$\mathbf{F}\phi$
t	f
b	f
n	f
f	t

ϕ	$\circ\phi$
t	f
b	f
n	t
f	t

Fig. 3 More truth tables

– $v(\Phi_1) = \mathtt{b}$ and $v(\Phi_2) = \mathtt{t}$ or $v(\Phi_2) = \mathtt{b}$.

As a consequence, if it happens that Φ_1 is valid while Φ_2 is not, the implication can be made valid by changing the truth value of Φ_2 in two ways: making it either true or inconsistent. As will be seen later, we choose to set $v_S(\Phi_2)$ as equal to $v_S(\Phi_1)$. This choice is motivated by the fact that it is the only one satisfying $v(\Phi_1) \preceq_k v(\Phi_2)$ and $v(\Phi_1) \preceq_t v(\Phi_2)$.

To see how to express FDE implication $\rightarrow$ in terms of the basic connectors $\neg$, $\vee$, $\wedge$, $\not\sim$, $\oplus$ and $\otimes$ of [9], we recall that $\sim$ is defined for every formula ϕ by:

$$\sim \phi = \neg \not\sim \neg \not\sim \phi = \not\sim \neg \not\sim \neg\phi.$$

Moreover, the additional connectors **T**, **B**, **N** and **F**, whose truth tables are shown in Figure 3, allow to 'characterize' each truth value in terms of only the standard ones, namely `t` and `f`. Roughly speaking, given a truth value `v`, the corresponding connector which we denote by **V**, is defined for every formula ϕ by the fact that $\mathbf{V}\phi$ is true if ϕ has the truth value `v` and false otherwise.

In what follows, *equivalent* formulas ϕ_1 and ϕ_2 are defined as formulas having the same truth tables, which is denoted by $\phi_1 \equiv \phi_2$. Using this notation, it is shown in [9] that for each of these connectors, the following equivalences hold:

$$\mathbf{T}\phi \equiv \phi \wedge \sim \neg\phi \ ; \ \mathbf{B}\phi \equiv \not\sim \phi \wedge \not\sim \neg\phi \ ; \ \mathbf{N}\phi \equiv \sim \not\sim \phi \wedge \neg \not\sim \phi \ ; \ \mathbf{F}\phi \equiv \sim \phi \wedge \neg\phi.$$

We now consider an additional connector denoted by $\circ$, and defined as follows:

$$\circ\phi = \mathbf{N}(\phi) \vee \mathbf{F}(\phi).$$

This new connector 'characterizes' the non validity of a formula ϕ in terms of the truth values `t` and `f`. In other words, as shown in Figure 3, $\circ\phi$ is true if ϕ is not valid and false otherwise.

An important point is that this new connector allows for an intuitively appealing expression of the FDE implication ([7,8]) $\rightarrow$. It is indeed easy to show based on the truth tables of Figure 2 and Figure 3, that for all formulas ϕ_1 and ϕ_2, the following equivalence holds:

$$\phi_1 \to \phi_2 \equiv \circ\phi_1 \vee \phi_2.$$

Since $\circ\phi$ can be read as *true if ϕ is not valid and false otherwise*, the equivalence above suggests that $\phi_1 \to \phi_2$ can be read as *either ϕ_1 is not valid or ϕ_2 is valid*. We emphasize that this is pretty much like implication in standard FOL that is read as *either not ϕ_1 is true or ϕ_2 is true*.

Based on these remarks and on truth tables in Figures 1–3, the following proposition holds. The first item in this proposition is the subject of some comments in the next section.

Proposition 1 *Given formulas ϕ_1, ϕ_2 and ϕ_3, the following equivalences hold:*

- $(\phi_1 \vee \phi_2) \to \phi_3 \equiv (\phi_1 \oplus \phi_2) \to \phi_3 \equiv (\phi_1 \to \phi_3) \wedge (\phi_2 \to \phi_3)$
- $(\phi_1 \wedge \phi_2) \to \phi_3 \equiv (\phi_1 \otimes \phi_2) \to \phi_3 \equiv (\phi_1 \to \phi_3) \vee (\phi_2 \to \phi_3)$.

2.2 About Functional Completeness

Functional completeness in our context can be stated as follows: Given a function W from $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}^k$ to $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}$ where k is a positive integer, can W be 'expressed' as a formula $\Phi_W(P_1, P_2, \ldots, P_k)$ involving k propositional variables $P_1, P_2, \ldots, P_k$? More formally, given W, the problem is to prove that there exists a formula Φ_W such that for $\mathtt{V} = (\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_k)$ in $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}^k$, if v is a valuation such that for $i = 1, 2, \ldots, k$, $v(P_i) = \mathtt{v}_i$, then $v(\Phi_W(\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_k)) = W(\mathtt{V})$.

This question has been answered positively in [7] where the proposed formula Φ_W involves the connectors $\neg$, $\wedge$ and $\to$ and the constants $\mathtt{b}$ and $\mathtt{n}$. The authors give also some other variants of this result by proposing various sets of connectors, all of which containing the implication $\to$.

Given that $\phi_1 \to \phi_2$ can be expressed as $\circ\phi_1 \vee \phi_2$, functional completeness can also be shown based on the connectors introduced in [9], that is $\neg$, $\not\sim$, $\vee$, $\wedge$, $\oplus$ and $\otimes$, but not $\to$. We prove this result in two ways: one based on [7], and one more direct, using the connectors defined in [9].

Proof based on [7]. In [7], it is shown that the language $L^* = \{\neg, \wedge, \to, \mathtt{n}, \mathtt{b}\}$ is functionally complete, meaning that for every $k \geq 0$ and every function W from $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}^k$ to $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}$ there exists a formula Φ^*_W in L^* involving k propositional variables $P_1, P_2, \ldots, P_k$ such that, for $\mathtt{V} = (\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_k)$ in $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}^k$, if v is a valuation such that for $i = 1, 2, \ldots, k$, $v(P_i) = \mathtt{v}_i$, then $v(\Phi^*_W(\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_k)) = W(\mathtt{V})$.

Thus, given W from $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}^k$ to $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}$, by replacing in Φ^*_W every occurrence of $\phi_1 \to \phi_2$ by $\circ\phi_1 \vee \phi_2$ we obtain a formula Φ°_W that, using the definitions of $\circ$ and of the connectors **N** and **F**, can be expressed by using the basic connectors $\neg$, $\wedge$, $\vee$, $\oplus$, $\otimes$, $\not\sim$ and the four truth values.

Direct proof based on [9]. Based on the connectors **T**, **B**, **N** and **F** introduced in [9], every $\mathtt{V} = (\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_k)$ in $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}^k$ is associated with a formula $\phi_{\mathtt{V}}(P_1, P_2, \ldots, P_k)$ defined as follows:

$$\phi_{\mathtt{V}}(P_1, P_2, \ldots, P_k) = \textstyle\bigwedge_{i=1}^{i=k} \phi_i(P_i)$$

where, for $i = 1, 2, \ldots, k$, $\phi_i(P_i) = \mathbf{T}P_i$ if $\mathtt{v}_i = \mathtt{t}$, $\phi_i(P_i) = \mathbf{B}P_i$ if $\mathtt{v}_i = \mathtt{b}$, $\phi_i(P_i) = \mathbf{N}P_i$ if $\mathtt{v}_i = \mathtt{n}$ and $\phi_i(P_i) = \mathbf{F}P_i$ if $\mathtt{v}_i = \mathtt{f}$.

It is thus easy to see that $v(\phi_{\mathtt{V}}(P_1, P_2, \ldots, P_k)) = \mathtt{t}$ if for $i = 1, 2, \ldots, k$, $v(P_i) = \mathtt{v}_i$ and $v(\phi_{\mathtt{V}}(P_1, P_2, \ldots, P_k)) = \mathtt{f}$ otherwise.

Now, given a function W from $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}^k$ to $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}$, we consider the partition induced by W on $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}^k$, defined by $\{W^{-1}(\mathtt{t}),\ W^{-1}(\mathtt{b}),\ W^{-1}(\mathtt{n}),\ W^{-1}(\mathtt{f})\}$. For every truth value $\mathtt{v}$ in $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}$, the corresponding element $W^{-1}(\mathtt{v})$ of this partition, which is a subset of $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}^k$, is associated with a formula $\Phi_\mathtt{v}$ defined by:

$$\Phi_\mathtt{v} = \bigvee_{\mathtt{V}\in W^{-1}(\mathtt{v})} \phi_\mathtt{V}.$$

It can be seen that for every $\mathtt{v}$ in $\{\mathtt{t},\mathtt{b},\mathtt{n},\mathtt{f}\}$, $v(\Phi_\mathtt{v}) = \mathtt{t}$ if $(v(P_1), v(P_2), \ldots, v(P_k))$ is in $W^{-1}(\mathtt{v})$, and $v(\Phi_\mathtt{v}) = \mathtt{f}$ otherwise. The targetted formula Φ_W is defined by:

$$\Phi_W = ((\Phi_\mathtt{t} \vee \neg\Phi_\mathtt{f}) \otimes \sim\!\!\not\sim \Phi_\mathtt{n}) \oplus \not\sim \Phi_\mathtt{b}.$$

The proof that Φ_W is indeed the expected formula is done by successively considering the four possible truth values. For $\mathtt{V} = (\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_k)$, consider the following cases:

- $\mathtt{V} \in W^{-1}(\mathtt{t})$: In this case, we have that $W(\mathtt{V}) = \mathtt{t}$. On the other hand, if v is such that for $i = 1, 2, \ldots, k$, $v(P_i) = \mathtt{v}_i$, $v(\Phi_\mathtt{t}) = \mathtt{t}$, $v(\Phi_\mathtt{b}) = \mathtt{f}$, $v(\Phi_\mathtt{n}) = \mathtt{f}$ and $v(\Phi_\mathtt{f}) = \mathtt{f}$, $v(\Phi_W)$ evaluates as $v(\Phi_W) = ((\mathtt{t} \vee \neg\mathtt{f}) \otimes \sim\!\!\not\sim \mathtt{f}) \oplus \not\sim \mathtt{f} = \mathtt{t}$. Thus, $W(\mathtt{V}) = v(\Phi_W) = \mathtt{t}$.
- $\mathtt{V} \in W^{-1}(\mathtt{b})$: In this case, we have that $W(\mathtt{V}) = \mathtt{b}$. On the other hand, if v is such that for $i = 1, 2, \ldots, k$, $v(P_i) = \mathtt{v}_i$, $v(\Phi_\mathtt{t}) = \mathtt{f}$, $v(\Phi_\mathtt{b}) = \mathtt{t}$, $v(\Phi_\mathtt{n}) = \mathtt{f}$ and $v(\Phi_\mathtt{f}) = \mathtt{f}$, $v(\Phi_W)$ evaluates as $v(\Phi_W) = ((\mathtt{f} \vee \neg\mathtt{f}) \otimes \sim\!\!\not\sim \mathtt{f}) \oplus \not\sim \mathtt{t} = \mathtt{b}$. Thus, $W(\mathtt{V}) = v(\Phi_W) = \mathtt{b}$.
- $\mathtt{V} \in W^{-1}(\mathtt{n})$: In this case, we have that $W(\mathtt{V}) = \mathtt{n}$. On the other hand, if v is such that for $i = 1, 2, \ldots, k$, $v(P_i) = \mathtt{v}_i$, $v(\Phi_\mathtt{t}) = \mathtt{f}$, $v(\Phi_\mathtt{b}) = \mathtt{f}$, $v(\Phi_\mathtt{n}) = \mathtt{t}$ and $v(\Phi_\mathtt{f}) = \mathtt{f}$, $v(\Phi_W)$ evaluates as $v(\Phi_W) = ((\mathtt{f} \vee \neg\mathtt{f}) \otimes \sim\!\!\not\sim \mathtt{t}) \oplus \not\sim \mathtt{f} = \mathtt{n}$. Thus, $W(\mathtt{V}) = v(\Phi_W) = \mathtt{n}$.
- $\mathtt{V} \in W^{-1}(\mathtt{f})$: In this case, we have that $W(\mathtt{V}) = \mathtt{f}$. On the other hand, if v is such that for $i = 1, 2, \ldots, k$, $v(P_i) = \mathtt{v}_i$, $v(\Phi_\mathtt{t}) = \mathtt{f}$, $v(\Phi_\mathtt{b}) = \mathtt{f}$, $v(\Phi_\mathtt{n}) = \mathtt{f}$ and $v(\Phi_\mathtt{f}) = \mathtt{t}$, $v(\Phi_W)$ evaluates as $v(\Phi_W) = ((\mathtt{f} \vee \neg\mathtt{t}) \otimes \sim\!\!\not\sim \mathtt{f}) \oplus \not\sim \mathtt{f} = \mathtt{f}$. Thus, $W(\mathtt{V}) = v(\Phi_W) = \mathtt{f}$.

As a consequence, we obtain that $W(\mathtt{V}) = v(\Phi_W)$ thus that the formula Φ_W has the same truth values as the truth values defined by the function W.

3 Four-Valued Logic and Databases

3.1 Database Syntax

As usual when dealing with deductive databases, the considered alphabet is made of constants, variables and predicate symbols with a fixed arity. We thus assume a fixed set of contants, called *universe* and denoted by $\mathcal{U}$. It should be noticed that $\mathcal{U}$ may be infinite.

As in traditional approaches, a term t is either a constant from $\mathcal{U}$ or a variable, an *atomic formula* or an atom is a formula of the form $P(t_1, t_2, \ldots, t_k)$ where P is a k-ary predicate and for every $i = 1, 2, \ldots, k$, t_i is a term. A formula is said to be *ground* if it contains no variables. A *fact* is a ground atom, that is an atom in which all terms are constants. Moreover, a *literal* is either an atom or the negation of an atom. In the former case the literal is said to be *positive* and in the latter case it is said to be *negative*. The *Herbrand Base* associated with $\mathcal{U}$ is the set of

all facts that can be built up using the constants in $\mathcal{U}$ and the predicates. Clearly, if $\mathcal{U}$ is infinite, then so is $\mathcal{HB}$.

In the traditional two-valued setting under the CWA (Closed World Assumption [10]), the database extension and the database semantics are sets of facts, meant to be true, and the facts not in the database semantics are set to be false. In our context of Four-valued logic under the OWA (Open World Assumption), the database extension and the database semantics may contain facts that are either true, inconsistent or false, assuming that non stored facts are unknown. To account for this situation, we consider sets of pairs of the form $\langle \varphi, \mathtt{v} \rangle$ where φ is a fact in $\mathcal{HB}$ and where $\mathtt{v}$ is one of the values $\mathtt{t}$, $\mathtt{b}$ or $\mathtt{f}$, while facts whose truth value is $\mathtt{n}$ are not stored. Moreover, such a set S is said to be *consistent* if for all distinct pairs $\langle \varphi_1, \mathtt{v}_1 \rangle$ and $\langle \varphi_2, \mathtt{v}_2 \rangle$ in S, $\varphi_1 \neq \varphi_2$. Consequently a consistent set S is seen as a valuation v_S defined for every φ in $\mathcal{U}$ by:

$$v_S(\varphi) = \mathtt{v}, \text{ if } S \text{ contains a pair } \langle \varphi, \mathtt{v} \rangle \ ; \ v_S(\varphi) = \mathtt{n}, \text{ otherwise.}$$

Consistent sets of pairs are called *v-sets*, standing for *valuated* sets.

Given a v-set S and a ground formula Φ, Φ is said to be *valid in* S if $v_S(\Phi)$ is designated. For example, $P(a) \rightarrow Q(b)$ is valid in $S_1 = \{\langle P(a), \mathtt{t} \rangle, \langle Q(b), \mathtt{b} \rangle\}$ because $v_{S_1}(P(a) \rightarrow Q(b)) = \mathtt{b}$, but $P(a) \rightarrow Q(b)$ is not valid in $S_2 = \{\langle P(a), \mathtt{t} \rangle\}$ because $v_{S_2}(P(a) \rightarrow Q(b)) = \mathtt{n}$.

The two orderings $\preceq_k$ and $\preceq_t$ are extended to v-sets over the same base $\mathcal{HB}$ in a point-wise manner as follows.

Definition 1 For all v-sets S_1 and S_2 over $\mathcal{U}$, $S_1 \preceq_k S_2$, respectively $S_1 \preceq_t S_2$, holds if for every φ in $\mathcal{U}$, $v_{S_1}(\varphi) \preceq_k v_{S_2}(\varphi)$, respectively $v_{S_1}(\varphi) \preceq_t v_{S_2}(\varphi)$, holds.

For example for $\mathcal{HB} = \{P(a), P(b), P(c)\}$, $S_1 = \{\langle P(a), \mathtt{t} \rangle\}$ and $S_2 = \{\langle P(a), \mathtt{b} \rangle, \langle P(b), \mathtt{f} \rangle\}$, we have $v_{S_1}(P(b)) = v_{S_1}(P(c)) = v_{S_2}(P(c)) = \mathtt{n}$. Thus:

- $v_{S_1}(P(a)) \preceq_k v_{S_2}(P(a))$, $v_{S_1}(P(b)) \preceq_k v_{S_2}(P(b))$ and $v_{S_1}(P(c)) \preceq_k v_{S_2}(P(c))$, implying that $S_1 \preceq_k S_2$ holds.
- $v_{S_2}(P(a)) \preceq_t v_{S_1}(P(a))$, $v_{S_2}(P(b)) \preceq_t v_{S_1}(P(b))$ and $v_{S_2}(P(c)) \preceq_t v_{S_1}(P(c))$, implying that $S_2 \preceq_t S_1$ holds.
- $\emptyset \preceq_k S_2$, because for every φ, $v_\emptyset(\varphi) = \mathtt{n}$, the least value with respect to $\preceq_k$.
- $\emptyset$ and S_2 are not comparable with respect to $\preceq_t$, because $v_\emptyset(P(a)) = \mathtt{n}$ and $v_{S_2}(P(a)) = \mathtt{b}$ are not comparable with respect to $\preceq_t$.

The extension of $\preceq_k$ generalizes set inclusion in the sense that if $S_1 \subseteq S_2$, then we have $S_1 \preceq_k S_2$. Notice that, as the last item above shows, the truth ordering $\preceq_t$ does not satisfy this property, because $\emptyset \subseteq S_2$ holds while $\emptyset \preceq_t S_2$ does not.

In our context, as in approaches to Datalog databases ([1,11]), a database consists of an *extension* and a *set of rules*, formally defined as follows.

Definition 2 A database Δ is a pair $\Delta = (E, R)$ where E and R are respectively called the *extension* and the *rule set* of Δ. If $\Delta = (E, R)$, then:

- E is a v-set.
- R is a set of rules of the form $\rho : h(X) \leftarrow B(X, Y)$ where the variables in X are free in $h(X)$ and $B(X, Y)$ and the variables in Y are free in $B(X, Y)$, and
  1. $B(X, Y)$ is a well formed formula involving the connectors $\neg$, $\vee$, $\wedge$, $\oplus$ and $\otimes$. $B(X, Y)$ is called the *body* of ρ, denoted by $body(\rho)$.
  2. $h(X)$ is a positive or negative literal, called the *head* of ρ, denoted by $head(\rho)$.

It should be clear that the rules as defined above generalize standard Datalog^{neg} rules ([11]). On the other hand, the definition above also generalizes rules as defined in [6] where the bodies of the rules are restricted to be conjunctions only. Moreover, in our approach and contrary to [5,11], rules may generate contradictory facts. It is important to notice that our approach is closely related to the generalized rules as introduced in [5], with the following notable differences:

1. In our approach, negative literals are allowed in the rule heads, which is not the case in [5].
2. In our approach, several rules may have the same predicate involved in their head, which is not the case in [5]. This important point will be discussed later.
3. In our approach, quantifiers are not allowed, whereas in [5] four quantifiers are allowed ($\forall$ and $\exists$ associated with $\preceq_t$ and Π and Σ associated with $\preceq_k$).

3.2 Database Semantics

As usual, rules are seen as implications, either $\rightarrow$ or $\hookrightarrow$ that must be valid in the database semantics. Notice in this respect that Figure 2 shows that for all formulas ϕ_1 and ϕ_2, $\phi_1 \rightarrow \phi_2$ is valid if and only if so is $\phi_1 \hookrightarrow \phi_2$. This explains why in [6], our approach has been shown to be 'compatible' with either implication. Here, we focus on FDE implication $\rightarrow$, thus forgetting the implication $\hookrightarrow$ of [9].

Similarly to the standard Datalog approach, a model of a database $\Delta = (E, R)$ could be defined as a v-set M containing E and in which all rules in R are valid. However, such a definition would raise important problems:

1. *A database might have no model.* To see this, consider $\Delta = (E, R)$ where $R = \{Q(b) \leftarrow P(a)\}$ and where $E = \{\langle P(a), \mathtt{t}\rangle, \langle Q(b), \mathtt{f}\rangle\}$. Then in any model M, $v_M(P(a) \rightarrow Q(b)) = \mathtt{f}$ because M must contain the two pairs of E. Notice that this cannot happen in standard Datalog since the storage of false facts is not allowed.
2. *A database might have more than one minimal model, with respect to set inclusion.* This case is illustrated above where $S'_1 = \{\langle P(a), \mathtt{t}\rangle, \langle Q(b), \mathtt{t}\rangle\}$ are $S'_2 = \{\langle P(a), \mathtt{t}\rangle, \langle Q(b), \mathtt{b}\rangle\}$ two minimal v-sets containing $\{\langle P(a), \mathtt{t}\rangle\}$ in which $Q(b) \leftarrow P(a)$ is valid. This situation does not happen in standard Datalog because the minimal model is known to be unique.

Whereas the second issue raised above will be further investigated later, the first issue is solved in our approach by giving the priority to the database extension over the rules. To do so, we prevent from applying a rule in R when it leads to some conflict with a v-pair in E.

In order to implement this policy, given a database $\Delta = (E, R)$ over universe $\mathcal{U}$, we denote by $inst(E, R)$ the set of all instantiations ρ of rules in R such that $head(\rho)$ does not occur in E. Moreover, given a rule $\rho : head(\rho) \leftarrow body(\rho)$ we denote by $\rho^{\rightarrow}$ the formula $body(\rho) \rightarrow head(\rho)$. The definition of a model of Δ then follows.

Definition 3 Let $\Delta = (E, R)$ be a database. A v-set M is a *model* of Δ if the following holds:

1. $E \subseteq M$, i.e., M must contain the database extension, and
2. every ρ of $inst(E, R)$ is valid in M, that is, $v_M(\rho^{\rightarrow})$ is designated.

To illustrate Definition 3, consider the following simple examples:

- $\Delta = (E, R)$ with $E = \{\langle P(a), \mathtt{t}\rangle, \langle Q(b), \mathtt{f}\rangle\}$ and $R = \{Q(b) \leftarrow P(a)\}$. E is a model of Δ as $inst(E, R) = \emptyset$. It is easy to see that E is the only minimal model with respect to set inclusion.
- $\Delta = (E, R)$ with $E = \{\langle P(a), \mathtt{t}\rangle\}$ and $R = \{Q(b) \leftarrow P(a)\}$ $S_1 = \{\langle P(a), \mathtt{t}\rangle, \langle Q(b), \mathtt{t}\rangle\}$ and $S_2 = \{\langle P(a), \mathtt{t}\rangle, \langle Q(b), \mathtt{b}\rangle\}$ are two models of Δ. Moreover, it can be seen that these two models are minimal with respect to set inclusion.

Given a database Δ, an immediate consequence operator is defined below. It will then be seen that this allows for computing a particular model of Δ, which we call the *semantics* of Δ.

Definition 4 Let $\Delta = (E, R)$ be a database. The *semantic immediate consequence operator* associated with Δ, denoted by Σ_Δ, is defined for every v-set S by the following steps:

1. Define first $\Gamma^E_\Delta(S)$ as follows:

$$\begin{aligned}\Gamma^E_\Delta(S) = S \cup \{\langle h, \mathtt{t}\rangle \mid (\exists \rho \in inst(E,R))(h = head(\rho) \wedge v_S(body(\rho)) = \mathtt{t})\}\\ \cup \{\langle h, \mathtt{b}\rangle \mid (\exists \rho \in inst(E,R))(h = head(\rho) \wedge v_S(body(\rho)) = \mathtt{b})\}\\ \cup \{\langle h, \mathtt{f}\rangle \mid (\exists \rho \in inst(E,R))(\neg h = head(\rho) \wedge v_S(body(\rho)) = \mathtt{t})\}\\ \cup \{\langle h, \mathtt{b}\rangle \mid (\exists \rho \in inst(E,R))(\neg h = head(\rho) \wedge v_S(body(\rho)) = \mathtt{b})\}\end{aligned}$$

2. Then, define $\Sigma_\Delta(S)$ by: $\Sigma_\Delta(S) = \{\langle \varphi, \mathtt{v}_\oplus(\varphi)\rangle \mid \varphi \text{ occurs in } \Gamma^E_\Delta(S)\}$, where $\mathtt{v}_\oplus(\varphi) = \bigoplus\{\mathtt{v} \mid \langle \varphi, \mathtt{v}\rangle \in \Gamma^E_\Delta(S)\}$.

Definition 4 should be seen as fitting our view on rule semantics based of FDE implication, whose validity has been expressed earlier as $\phi_1 \rightarrow \phi_2$ *is valid if and only if whenever* ϕ_1 *is valid, so is* ϕ_2. This point of view is similar to that in Datalog databases (where 'valid' means 'true'), but different from the one in [5], where the truth value of the head of the rule is equated to that of the body, *whatever the truth value of the body*, even when it is $\mathtt{f}$. The following lemma shows basic properties of the operator Σ_Δ.

Lemma 1 *For every database* $\Delta = (E, R)$, Σ_Δ *is monotonic and continuous with respect to* $\preceq_k$.

Proof We first notice that the connectors involved in rule bodies are monotonic, that is, for all formulas ϕ_1 and ϕ_2 involving $\neg$, $\vee$, $\wedge$, $\oplus$ or $\otimes$, if S_1 and S_2 are two v-sets such that $S_1 \preceq_k S_2$ then $v_{S_1}(\phi_1) \preceq_k v_{S_2}(\phi_2)$ (this can be checked for each operator based on the truth tables in Figure 1).

For every φ in $\mathcal{HB}$ and every $i = 1, 2$, let $D^+_i(\varphi)$ (respectively $D^-_i(\varphi)$) denote the set of all rules ρ in $inst(E, R)$ such that $v_{S_i}(body(\rho))$ is distinguished in S_i and $head(\rho) = \varphi$ (respectively $head(\rho) = \neg\varphi$). Then, $v_{\Sigma_\Delta(S_i)}(\varphi)$ can be defined as follows:

$$v_{\Sigma_\Delta(S_i)}(\varphi) = v_{S_i}(\varphi) \oplus \bigoplus_{\rho \in D^+_i(\varphi)} v_{S_i}(body(\rho)) \oplus \bigoplus_{\rho \in D^-_i(\varphi)} \neg v_{S_i}(body(\rho))$$

By monotonicity of $\oplus$ and $\neg$, we obtain that , if $S_1 \preceq_k S_2$, then for every φ in $\mathcal{HB}$, $v_{\Sigma_\Delta(S_1)}(\varphi) \preceq_k v_{\Sigma_\Delta(S_2)}(\varphi)$, thus entailing the monotonicity of Σ_Δ with respect to $\preceq_k$. The proof that Σ_Δ is continuous with respect to $\preceq_k$, is as in [5] (see the proof of Theorem 16) and thus omitted here.

As a consequence of Lemma 1, given $\Delta = (E, R)$, let $\left(\Sigma^i\right)_{i\geq 0}$ the sequence defined by

$\Sigma^0 = E$, and for every $n \geq 1$, $\Sigma^n = \Sigma_\Delta(\Sigma^{n-1})$

has a limit which is the unique least-fixed point of Σ_Δ that is reached for some ordinal at most ω. This limit, denoted by Σ^*_Δ, is called the *semantics of* Δ and the valuation $v_{\Sigma^*_\Delta}$ is denoted by v_Δ.

Example 1 We illustrate the computation of the semantics in the context of our running example, where $\Delta = (E, R)$ is defined by:

- $E = \{\langle H_1(101), \mathtt{f}\rangle, \langle H_2(101), \mathtt{f}\rangle, \langle W_1(101), \mathtt{t}\rangle, \langle H_2(202), \mathtt{t}\rangle, \langle W_1(202), \mathtt{f}\rangle, \langle W_2(202), \mathtt{t}\rangle, \langle W_1(303), \mathtt{f}\rangle\}$
- $R = \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, \rho_7\}$, where
  $\rho_1 : Humid(x) \leftarrow H_1(x) \oplus H_2(x)$
  $\rho_2 : White(x) \leftarrow W_1(x) \oplus W_2(x)$
  $\rho_3 : Store(x) \leftarrow \neg Humid(x) \wedge White(x)$
  $\rho_4 : \neg Store(x) \leftarrow Humid(x)$
  $\rho_5 : Cure(x) \leftarrow Humid(x)$
  $\rho_6 : \neg Store(x) \leftarrow \neg White(x)$
  $\rho_7 : New\_test(x) \leftarrow \neg White(x)$

We first note that in case, $inst(E, R) = R$ because no predicate occurring in E appears in the heads of the rules of R. On the other hand, variables have only three possible instantiations, namely 101, 202 and 303. The computation of Σ^*_Δ is as follows, starting with $\Sigma^0 = E$:

1. $\Sigma^1 = \Sigma_\Delta(\Sigma^0)$. The rule ρ_1 generates $\langle Humid(101), \mathtt{f}\rangle$ and $\langle Humid(202), \mathtt{t}\rangle$, and ρ_2 generates $\langle White(101), \mathtt{t}\rangle$, $\langle White(202), \mathtt{b}\rangle$, and $\langle White(303), \mathtt{f}\rangle$.
   Since $\Sigma_\Delta(\Sigma^0) = \Gamma^E_\Delta(\Sigma^0)$ we obtain that $\Sigma^1 = E \cup \{\langle Humid(101), \mathtt{f}\rangle, \langle Humid(202), \mathtt{t}\rangle, \langle White(101), \mathtt{t}\rangle, \langle White(202), \mathtt{b}\rangle, \langle White(303), \mathtt{f}\rangle\}$.
2. $\Sigma^2 = \Sigma_\Delta(\Sigma^1)$. The computation involves the 5 rules $\rho_3 \ldots \rho_7$ as follows:
   − ρ_3 generates $\langle Store(101), \mathtt{t}\rangle$, because $\neg Humid(101) \wedge White(101)$ has truth value $\mathtt{t}$. The other instances of ρ_3 do not apply because the body is not valid.
   − ρ_4 and ρ_5 generate respectively $\langle Store(202), \mathtt{f}\rangle$ and $\langle Cure(202), \mathtt{t}\rangle$ because $Humid(202)$ has truth value $\mathtt{t}$. The other instances of ρ_4 and of ρ_5 do not apply because the body is not valid.
   − ρ_6 and ρ_7 generate respectively $\langle Store(202), \mathtt{b}\rangle$ and $\langle New\_test(202), \mathtt{b}\rangle$ since $White(202)$ has truth value $\mathtt{b}$, remembering that $\neg\mathtt{b} = \mathtt{b}$.
   − ρ_6 and ρ_7 generate respectively $\langle Store(303), \mathtt{f}\rangle$ and $\langle New\_test(303), \mathtt{t}\rangle$ since $White(303)$ has truth value $\mathtt{t}$.
   − As $\Gamma^E_\Delta(\Sigma^1)$ contains $\langle Store(202), \mathtt{b}\rangle$ and $\langle Store(202), \mathtt{t}\rangle$, the computation of Σ^2 consists in integrating these v-pairs into $\langle Store(202), \mathtt{b}\rangle$, remembering that $\mathtt{b} \oplus \mathtt{t} = \mathtt{b}$. We thus obtain that
   $\Sigma^2 = \Sigma^1 \cup \{\langle Store(101), \mathtt{t}\rangle, \langle Store(202), \mathtt{b}\rangle, \langle Store(303), \mathtt{f}\rangle, \langle Cure(202), \mathtt{t}\rangle, \langle New\_test(202), \mathtt{b}\rangle, \langle New\_test(303), \mathtt{t}\rangle\}$.
3. Since no rule applies on Σ^2 to produce new v-pairs, the computation stops returning $\Sigma^*_\Delta = \Sigma^2$.

We draw attention on that Σ^*_Δ is a model of Δ because $E \subseteq \Sigma^*_\Delta$ and all instantiations of the rules in R are valid. For example the instantiation of x in ρ_4 and ρ_6 by 202 is valid in Σ^*_Δ because:

- $v_\Delta(\rho_4^{\rightarrow}) = \mathtt{b}$, since $v_\Delta(Humid(202)) = \mathtt{t}$ and $v_\Delta(Store(202)) = v_\Delta(\neg Store(202)) = \mathtt{b}$
- $v_\Delta(\rho_6^{\rightarrow}) = \mathtt{b}$, since $v_\Delta(\neg White(202)) = v_\Delta(\neg Store(202)) = \mathtt{b}$. □

The following proposition, shows that Σ^*_Δ is a model of Δ.

Proposition 2 *Given a database $\Delta = (E, R)$, Σ^*_Δ is a minimal model of Δ, with respect to set inclusion.*

Proof We show that Σ^*_Δ is a model of Δ by contraposition, assuming that Σ^*_Δ is not a model of Δ. First, we have $E \subseteq \Sigma^0$ and then, as $E \preceq_k \Sigma^*_\Delta$ holds by monotonicity and as no instantiated rule can change the truth value of the facts involved in E, we have $E \subseteq \Sigma^*_\Delta$. Thus, assuming that Σ^*_Δ is not a model of Δ implies that at least one rule ρ of $inst(E, R)$ is not valid in Σ^*_Δ. In this case, $head(\rho)$ is not valid, while $body(\rho)$ is valid. Then, denoting $head(\rho)$ by φ (respectively $\neg\varphi$), we have $v_\Delta(\varphi) = \mathtt{n}$ or $v_\Delta(\varphi) = \mathtt{f}$ (respectively $v_\Delta(\varphi) = \mathtt{t}$) along with $v_\Delta(body(\rho))$ equal to $\mathtt{t}$ or $\mathtt{b}$. Consequently $\Sigma_\Delta(\Sigma^*_\Delta) \neq \Sigma^*_\Delta$, which is not possible by Definition 4. This part of the proof is thus complete.

To show the minimality of Σ^*_Δ, we show that for every nonempty subset σ of Σ^*_Δ, $S = \Sigma^*_\Delta \setminus \sigma$ cannot be a model of Δ. To this end, assuming that S is a model of Δ, let k be the least integer such that $\Sigma^{k-1} \cap \sigma = \emptyset$ and $\Sigma^k \cap \sigma \neq \emptyset$. We notice that k exists such that $k > 0$ because, since S is a model of Δ, it holds that $E \subseteq S$ and so, since $\Sigma^0 = E$, we have $\Sigma^0 \cap \sigma = \emptyset$.

Let $\langle \varphi, \mathtt{v} \rangle$ be in $\Sigma^k \cap \sigma$ but not in Σ^{k-1}. In this case, $v_S(\varphi) = \mathtt{n}$ and as above, there exists one rule ρ in $inst(E, R)$ such that $head(\rho)$ is either φ or $\neg\varphi$ and in Σ^{k-1}, $head(\rho)$ is not valid, while $body(\rho)$ is valid. Since $\Sigma^{k-1} \subseteq S$, we have $\Sigma^{k-1} \preceq_k S$ and so, by monotonicity of the connectors involved in $body(\rho)$, $v_{\Sigma^{k-1}}(body(\rho)) \preceq_k v_S(body(\rho))$. As $body(\rho)$ is valid in Σ^{k-1}, so is it in S. Since $head(\rho)$ is not valid in S, ρ is not valid in S either. S being assumed to be a model of ρ, we obtain a contradiction, which completes the proof.

It has been shown in [6] that, even with conjunctive rules, Σ^*_Δ is not the only minimal model with respect to set inclusion, nor is it a minimal or a maximal model, with respect to any of the orderings $\preceq_k$ and $\preceq_t$. However, we also recall from [6] that, with conjunctive rules whose heads are positive literals (*i.e.,* for Dalatog^{neg} rules) all minimal models with respect to set inclusion share the same false facts and the same valid facts.

At this point, we would like to come back to Proposition 1, and make an important observation regarding the two closely related notions of implication and rule. We recall that the first item in that proposition is the following:

– $(\phi_1 \vee \phi_2) \rightarrow \phi_3 \equiv (\phi_1 \oplus \phi_2) \rightarrow \phi_3 \equiv (\phi_1 \rightarrow \phi_3) \wedge (\phi_2 \rightarrow \phi_3)$.

Now, consider the three implications as sets of instantiated rules:

$R_1 = \{\varphi \leftarrow \phi_1 \vee \phi_2\}$, $R_2 = \{\varphi \leftarrow \phi_1 \oplus \phi_2\}$ and $R_3 = \{\varphi \leftarrow \phi_1,\ \varphi \leftarrow \phi_2\}$

The important observation here is that when computing the corresponding semantics the results are *different*. In other words, the three sets of rules lead to different semantics, although the associated implications are *equivalent*.

We illustrate this important observation through the following example, in which we also compare our approach with that in [5].

Example 2 Let $\Delta_1^{\mathtt{v}_1,\mathtt{v}_2} = (E^{\mathtt{v}_1,\mathtt{v}_2}, R_1)$, $\Delta_2^{\mathtt{v}_1,\mathtt{v}_2} = (E^{\mathtt{v}_1,\mathtt{v}_2}, R_2)$, $\Delta_3^{\mathtt{v}_1,\mathtt{v}_2} = (E^{\mathtt{v}_1,\mathtt{v}_2}, R_3)$ be three families of databases where $\mathtt{v}_1$ and $\mathtt{v}_2$ are truth values in $\{\mathtt{t}, \mathtt{b}, \mathtt{n}, \mathtt{f}\}$, $E^{\mathtt{v}_1,\mathtt{v}_2}$ is either $\{\langle P(a), \mathtt{v}_1\rangle, \langle Q(a), \mathtt{v}_2\rangle\}$ when $\mathtt{v}_1 \neq \mathtt{n}$ and $\mathtt{v}_1 \neq \mathtt{n}$, or $\{\langle P(a), \mathtt{v}_1\rangle\}$

$\Delta_1^{\mathtt{v}_1,\mathtt{v}_2}$	t	b	n	f
t	t	t	t	t
b	t	b	t	b
n	t	t	n	n
f	t	b	n	n

$\Delta_2^{\mathtt{v}_1,\mathtt{v}_2}$	t	b	n	f
t	t	b	t	b
b	b	b	b	b
n	t	b	n	n
f	b	b	n	n

$\Delta_3^{\mathtt{v}_1,\mathtt{v}_2}$	t	b	n	f
t	t	b	t	t
b	b	b	b	b
n	t	b	n	n
f	t	b	n	n

Fig. 4 Computing the truth values of $S(a)$

when $\mathtt{v}_1 \neq \mathtt{n}$ and $\mathtt{v}_2 = \mathtt{n}$, or $\{\langle Q(a), \mathtt{v}_2\rangle\}$ when $\mathtt{v}_1 = \mathtt{n}$ and $\mathtt{v}_2 \neq \mathtt{n}$, or $\emptyset$ when $\mathtt{v}_1 = \mathtt{v}_2 = \mathtt{n}$, and

- $R_1 = \{S(a) \leftarrow P(a) \vee Q(a)\}$,
- $R_2 = \{S(a) \leftarrow P(a) \oplus Q(a)\}$,
- $R_3 = \{S(a) \leftarrow P(a),\ S(a) \leftarrow Q(a)\}$.

We are thus considering $3 \times 16 = 48$ databases whose semantics are defined by $E^{\mathtt{v}_1,\mathtt{v}_2} \cup \{\langle S(a), \mathtt{v}_i^{12}\rangle\}$ where for $i = 1, 2, 3$, $\mathtt{v}_i^{12}$ is the truth value obtained by applying $\Sigma_{\Delta_i^{\mathtt{v}_1,\mathtt{v}_2}}$ to the rule(s) in R_i and the v-pairs in $E^{\mathtt{v}_1,\mathtt{v}_2}$. The arrays displayed in Figure 4 show these truth values based on $\mathtt{v}_1$ (the rows of the arrays) and $\mathtt{v}_2$ (the columns of the arrays). From left to right, the arrays correspond respectively to the three sets of rules R_1, R_2 and R_3.

For example, the value $\mathtt{t}$ in row '$\mathtt{b}$' and column '$\mathtt{n}$' of the array labelled $\Delta_1^{\mathtt{v}_1,\mathtt{v}_2}$ in Figure 4, means that $\langle S(a), \mathtt{t}\rangle$ belongs to the semantics of $\Delta_1^{\mathtt{b},\mathtt{n}} = (\{\langle P(a), \mathtt{b}\rangle\}, R_1)$, where $Q(a)$ has truth value $\mathtt{n}$.

It should be stressed that since all these arrays are pairwise distinct, all three sets R_1, R_2 and R_3 produce different semantics in some cases. As examples it can be seen from Figure 4 that:

- for $\mathtt{v}_1 = \mathtt{t}$ and $\mathtt{v}_2 = \mathtt{b}$, $S(a)$ is true in $\Delta_1^{\mathtt{t},\mathtt{b}}$ and in $\Delta_3^{\mathtt{t},\mathtt{b}}$, but false in $\Delta_2^{\mathtt{t},\mathtt{b}}$,
- for $\mathtt{v}_1 = \mathtt{b}$ and $\mathtt{v}_2 = \mathtt{n}$, $S(a)$ is true in $\Delta_1^{\mathtt{b},\mathtt{n}}$ and false in $\Delta_2^{\mathtt{b},\mathtt{n}}$ and in $\Delta_3^{\mathtt{b},\mathtt{n}}$.

As a consequence, this implies that contrary to standard Datalog approaches, replacing the rule in R_1 by the two rules in R_3 has an impact on the database semantics in certain cases, although R_1 and R_3 yield the equivalent formulas as shown in Proposition 1. Therefore, the claim in [5] whereby *'There is a standard way in Prolog to combine two program clauses for the same relation symbol, using equality. Similar ideas carry over to languages based on a wide variety of bilattices...'* does not hold in our approach. This also shows that rule based semantics do not always exactly 'coincide' with the semantics of implication. Consequently, the claim above is debatable even in the approach of [5], because no comparison is possible, as it makes no sense in [5] that more than one rule head involves the same predicate.

Referring to our running example, the previous statements show that replacing the rules $\rho_4 : \neg Store(x) \leftarrow Humid(x)$ and $\rho_6 : \neg Store(x) \leftarrow \neg White(x)$ by the rule $\rho_{46} : \neg Store(x) \leftarrow Humid(x) \vee \neg White(x)$ would lead to different semantics. Indeed, when considering ρ_4 and ρ_6, the fact that $Humid(202)$ and $White(202)$ have respective truth values $\mathtt{t}$ and $\mathtt{b}$, implies that $Store(202)$ has truth value $\mathtt{b}$. On the other hand, Figure 4 shows that when considering ρ_{46}, the same truth values for $Humid(202)$ and $White(202)$ imply that $Store(202)$ has truth value $\mathtt{f}$. □

3.3 Safe Rules

An important issue in rule based databases is that a database can have *infinite* semantics when $\mathcal{HB}$ is infinite. This point is indeed problematic because in such cases, answers to some queries can be infinite, which is not acceptable in practice.

As a simple case, consider $\Delta = (E, R)$ where $E = \{\langle S(a), \mathtt{t}\rangle\}$ and $R = \{P(x,y) \leftarrow Q(x,y) \vee S(x)\}$. Based on the truth table of $\vee$ shown in Figure 1, for all α and β in $\mathcal{U}$, $Q(\alpha,\beta) \vee S(\alpha)$ is true if so is $S(\alpha)$. Hence, $\Sigma^*_\Delta = \{\langle S(a), \mathtt{t}\rangle\} \cup \{\langle P(a,\beta), \mathtt{t}\rangle \mid \beta \in \mathcal{U}\}$, which is infinite when $\mathcal{U}$ is infinite.

To cope with this difficulty, we define the notion of *safe* rules, inspired by the case of Datalog^{neg} databases. To see how the approaches are related regarding this issue, let $\mathcal{D} = (\{S(a)\}, \{P(x,y) \leftarrow \neg Q(x,y) \wedge S(x)\})$ be a Datalog^{neg}, whose semantics is $\{S(a)\} \cup \{P(a,\beta) \mid \beta \in \mathcal{U}\}$. This result is somehow similar to that for Δ above, and the rule in $\mathcal{D}$ is clearly not safe since the variable y in $\neg Q(x,y)$ occurs in no positive literal in the body of the rule.

To formalize and characterize safe rules in our context, we need some preliminaries as detailed next. First, we adapt the notion of *active domain* in relational databases [2] to our approach as follows. Given a universe $\mathcal{U}$, its associated Herbrand base $\mathcal{HB}$ and a database $\Delta = (E, R)$ over $\mathcal{HB}$, we call the *active domain of* Δ, denoted by $\mathcal{A}(\Delta)$, the subset of $\mathcal{U}$ containing all the constants occurring in Δ. Then the *active Herbrand base of* Δ, denoted by $\mathcal{AB}(\Delta)$ is the set of all facts in $\mathcal{HB}$ that only involve constants in $\mathcal{A}(\Delta)$. Notice that $\mathcal{A}(\Delta)$ and $\mathcal{AB}(\Delta)$ are finite sets, even if $\mathcal{U}$ is infinite, because E and R are assumed to be finite. The notion of safe rule is defined as follows.

Definition 5 Given a Herbrand base $\mathcal{HB}$, a rule ρ is said to be *safe* if for every database $\Delta = (E, \{\rho\})$ where E is an arbitrary finite v-set involving facts in $\mathcal{HB}$, Σ^*_Δ is a subset of $\mathcal{AB}(\Delta)$.

We first notice that, according to Definition 5, allowing variables in the head of a rule not occurring in the body would generate non safe rules, and this explains why in Definition 2, we have restricted all variables occurring in the heads of the rules to also occur in the bodies. Indeed, let $\rho : P(x,y) \leftarrow Q(x)$ and $\Delta = (\{\langle Q(a), \mathtt{t}\rangle\}, \{\rho\})$. Then, we have $\mathcal{AB}(\Delta) = \{P(a,a), Q(a)\}$ and $\Sigma^*_\Delta = \{\langle Q(a), \mathtt{t}\rangle\} \cup \{\langle P(a,\beta), \mathtt{t}\rangle \mid \beta \in \mathcal{U}\}$, showing that ρ is not safe according to Definition 5. Other examples not relaxing the restriction in Definition 2 are presented next.

Example 3 The rule $\rho : P(x) \leftarrow P_1(x) \oplus P_2(x,y)$ is safe, according to Definition 5. Indeed, if *inst* is an instantiation of x and y such that $inst(body(\rho))$ is valid in E, then at least one of the instantiated atoms $P_1(\alpha_1)$ or $P_2(\alpha_2, \beta_2)$ is valid in E. Hence, these atoms can not generate a v-pair $P(\gamma)$ where γ is different than α_1 and α_2.

Notice that the above reasoning does not hold for $\rho' : P'(x,y) \leftarrow P_1(x) \vee P_2(x,y)$ because for $\Delta = (\{\langle P_1(a), \mathtt{t}\rangle\}, \{\rho'\})$, we have
$\Sigma^*_\Delta = \{\langle P_1(a), \mathtt{t}\rangle\} \cup \{\langle P'(a,\beta), \mathtt{t}\rangle \mid \beta \in \mathcal{U}\}$,
showing that ρ is not safe according to Definition 5. □

In order to syntactically characterize safe rules, we adapt the usual notion of *disjunctive normal form* of a formula to the context of Four-valued logic. To this end, we recall from [5,9] the following standard properties of the connectors of the Four-valued logic:

– $\neg(\phi_1 \vee \phi_2) \equiv \neg\phi_1 \wedge \neg\phi_2$; $\neg(\phi_1 \wedge \phi_2) \equiv \neg\phi_1 \vee \neg\phi_2$
– $\neg(\phi_1 \oplus \phi_2) \equiv \neg\phi_1 \oplus \neg\phi_2$; $\neg(\phi_1 \otimes \phi_2) \equiv \neg\phi_1 \otimes \neg\phi_2$
– Distributivity: for all distinct binary connectors $\star$ and $\bullet$ in $\{\vee, \wedge, \oplus, \otimes\}$
$\phi_1 \star (\phi_2 \bullet \phi_3) \equiv (\phi_1 \star \phi_2) \bullet (\phi_1 \star \phi_3)$.

Using these properties, any quantifier free formula Φ can be transformed into its equivalent $\vee\oplus$-*normal form* according to the following steps:

1. $\vee$-*transformation:* $\Phi \equiv \Phi_1 \vee \Phi_2 \vee \ldots \vee \Phi_n$ where for every i in $\{1, 2, \ldots, n\}$, Φ_i does not involve the connector $\vee$.
2. $\oplus$-*tranformation:* For every i in $\{1, 2, \ldots, n\}$, Φ_i is transformed into its equivalent $\oplus$-normal form $\phi_i^1 \oplus \phi_i^2 \oplus \ldots \oplus \phi_i^{p_i}$ where for j in $\{1, 2, \ldots, p_i\}$, ϕ_i^j does not involve the connector $\oplus$.
3. $\vee\oplus$-*transformation:* Combining these previous two steps, we obtain:
$\Phi \equiv (\phi_1^1 \oplus \phi_1^2 \oplus \ldots \oplus \phi_1^{p_1}) \vee (\phi_2^1 \oplus \phi_2^2 \oplus \ldots \oplus \phi_2^{p_2}) \vee \ldots \vee (\phi_n^1 \oplus \phi_n^2 \oplus \ldots \oplus \phi_n^{p_n})$, where for every i in $\{1, 2, \ldots, n\}$ and every j in $\{1, 2, \ldots, p_i\}$, $\vee$ and $\oplus$ do not occcur in ϕ_i^j.
4. $\wedge\otimes$-*transformation:* As for every i in $\{1, 2, \ldots, n\}$ and every j in $\{1, 2, \ldots, p_i\}$, the only connectors occurring in ϕ_i^j are $\neg$, $\wedge$ and $\otimes$, the following equivalent form of ϕ_i^j can be computed by applying transformations similar to those above:
$\phi_i^j \equiv (\lambda_1^1 \otimes \lambda_1^2 \otimes \ldots \otimes \lambda_1^{r_1}) \wedge (\lambda_2^1 \otimes \lambda_2^2 \otimes \ldots \otimes \lambda_2^{r_2}) \wedge \ldots \wedge (\lambda_q^1 \otimes \lambda_q^2 \otimes \ldots \otimes \lambda_q^{r_q})$, where for every i in $\{1, 2, \ldots, q\}$ and every j in $\{1, 2, \ldots, r_i\}$, λ_i^j is a literal, that is of the form φ or $\neg\varphi$ where φ is in $\mathcal{HB}$.

Combining these transformations yields a formula equivalent to Φ, called the $\vee\oplus$-*normal form of* Φ. Based on the truth tables of Figure 1, given a formula Φ involving no variable, for every v-set S, Φ is valid in S if and only if there exist i_0 in $\{1, 2, \ldots, n\}$ and j_0 in $\{1, 2, \ldots, p_{i_0}\}$ such that $\phi_{i_0}^{j_0}$ is valid in S. Furthermore, assuming that $\phi_{i_0}^{j_0}$ is written as shown in the last item above, $\phi_{i_0}^{j_0}$ is valid in S if and only if every literal λ occurring in the $\wedge\otimes$-transformation of $\phi_{i_0}^{j_0}$ is valid in S, that is $v_S(\lambda)$ is `t` or `b` if $\lambda = \varphi$, and $v_S(\lambda)$ is `f` or `b` if $\lambda = \neg\varphi$.

As a consequence, Φ is valid in S if and only in the $\vee\oplus$-normal of Φ, there exists a $\vee$- and $\oplus$-free sub-formula $\phi_{i_0}^{j_0}$ for which all involved literals are valid in S, and thus occur in S with an appropriate truth value. Based on this important remark, the following proposition can be stated.

Proposition 3 *Let $\rho : h(X) \leftarrow B(X, Y)$ be a rule such that $B(X, Y)$ is written in its $\vee\oplus$-normal form using the same notation as above. ρ is safe if and only if for every i in $\{1, 2, \ldots, n\}$ and every j in $\{1, 2, \ldots, p_i\}$, the sub-formula ϕ_i^j involves at least all variables in X.*

Proof Assume first that there exist i_0 in $\{1, 2, \ldots, n\}$ and j_0 in $\{1, 2, \ldots, p_{i_0}\}$ such that $\phi_{i_0}^{j_0}$ does not involve all variables in X. We write X as X_1X_2 to mean that the variables in X_1 occur in $\phi_{i_0}^{j_0}$ whereas those in X_2 do not. Let *inst* be an instantiation of the variables in X_1 and $\Delta = (E, \{\rho\})$ where E is the set of all v-pairs $\langle\varphi, \mathtt{b}\rangle$ such that φ occurs in $inst(\phi_{i_0}^{j_0})$. Then $inst(\phi_{i_0}^{j_0})$ is valid in E and so, for every extension $inst^*$ of *inst* to the variables in X_2 or in Y, $inst^*(body(\rho))$ is valid in E. Hence, $inst^*(h(X_1X_2))$ belongs to the semantics of Δ, meaning that ρ is not safe.

Conversely, if for every i in $\{1, 2, \ldots, n\}$ and every j in $\{1, 2, \ldots, p_i\}$, the sub-formula ϕ_i^j involves at least all variables in X, whatever the valid sub-formula

$\phi_{i_0}^{j_0}$ in $B(X,Y)$, the instantiation of the variables in $\phi_{i_0}^{j_0}$ assigns a value to every variable in X implying that the fact involved in $inst(h(X))$ is in $\mathcal{AB}(\Delta)$. Thus, ρ is safe, and the proof is complete.

4 Updates

We first would like to emphasize that our approach to updates follows the same policy as in our previous work on database updating [12,13], whereby priority is given to the latest updates with respect to the current database semantics. This means that updates are *always* taken into account and that their effect can not be overridden when computing the semantics. In this approach, such update persistency holds because instantiated rules whose heads involve a fact occurring in E, are not applied. This is made possible by restricting instantiated rules to belong to $inst(E,R)$.

4.1 Standard Update Semantics

Notice that, contrary to the traditional 2-valued models, in our approach, facts are stored associated with a truth value. We emphasize in this respect that, in standard 2-valued approaches under CWA, inserting (respectively deleting) φ should be understood as *take into account that* φ *becomes* true *(respectively* false*) in the database*. On the other hand, in our Four-valued approach, an update should rather be seen as a *change in the truth value* of a given fact. Formally, updates are defined as follows.

Definition 6 Let $\Delta = (E,R)$ be a database and $\nu = \langle \varphi, \mathtt{v} \rangle$ a v-pair. The result of the *update defined by* ν *in* Δ is the database $\Delta_\nu = (E_\nu, R)$ where E_ν is defined as follows:

- If $\nu = \langle \varphi, \mathtt{n} \rangle$ then $E_\nu = E \setminus \{\langle \varphi, v_E(\varphi)\rangle\}$
- Otherwise, $E_\nu = (E \setminus \{\langle \varphi, v_E(\varphi)\rangle\}) \cup \{\nu\}$.

In terms of truth value, an intuitive way to state Definition 6 is the following:

- If $\mathtt{v} = \mathtt{n}$, the update requires to set the truth value of φ to unknown, which amounts to remove from E any v-pair involving φ, if any. This corresponds to deletions in standard approaches.
- Otherwise, if $\mathtt{v} \neq \mathtt{n}$, the update consists in replacing the v-pair in E involving φ, if any, by the v-pair involved in the update, that is ν.

Example 4 In the context of our running example, due to $\langle Store(202), \mathtt{b} \rangle$ in the database semantics, it is likely that the bag has to be tested again. Assuming that in this case the sensors output the following: $\langle H_1(202), \mathtt{t} \rangle$, $\langle H_2(202), \mathtt{t} \rangle$ and $\langle W_1(202), \mathtt{t} \rangle$, these new v-pairs are inserted and the conflicting ones are deleted, thus resulting in the following updated database extension:

$E' = \{\langle H_1(101), \mathtt{f} \rangle, \langle H_2(101), \mathtt{f} \rangle, \langle W_1(101), \mathtt{t} \rangle, \langle H_1(202), \mathtt{t} \rangle, \langle H_2(202), \mathtt{t} \rangle,$
$\langle W_1(202), \mathtt{t} \rangle, \langle W_2(202), \mathtt{t} \rangle, \langle W_1(303), \mathtt{f} \rangle\}$. □

4.2 Other Possible Update Semantics

In the context of data integration, traditional updates are not always appropriate. Indeed, suppose that $\langle \varphi, \mathtt{t} \rangle$ has to be *integrated* in a given database $\Delta = (E, R)$ according to the following policy:

- If E contains no v-pair involving φ (*i.e.,* φ is unknown in Δ), then the integration of $\langle \varphi, \mathtt{t} \rangle$ is processed by inserting the v-pair in E.
- If E contains the v-pair $\langle \varphi, \mathtt{t} \rangle$, then the integration of $\langle \varphi, \mathtt{t} \rangle$ requires no change.
- If E contains the v-pair $\langle \varphi, \mathtt{f} \rangle$, then the integration of $\langle \varphi, \mathtt{t} \rangle$ implies that φ becomes *inconsistent* in Δ, meaning that $\langle \varphi, \mathtt{t} \rangle$ should be changed to $\langle \varphi, \mathtt{b} \rangle$.
- If E contains the v-pair $\langle \varphi, \mathtt{b} \rangle$, then the integration of $\langle \varphi, \mathtt{t} \rangle$ implies that φ remains *inconsistent* in Δ, meaning that no change is required.

The last two cases do *not* correspond to standard updates, because in the updated database, the truth value of φ is not the one specified in the update. In fact, the truth value of φ in the updated database is defined by $v_E(\varphi) \oplus \mathtt{t}$. Generalizing this remark, we define *integrative updates* as follows.

Definition 7 Let $\Delta = (E, R)$ be a database, $\nu = \langle \varphi, \mathtt{v} \rangle$ a v-pair and $\diamond$ a well formed binary expression involving the connectors $\neg$, $\vee$, $\wedge$, $\oplus$ or $\otimes$. The *integrative update* on Δ defined by $(\nu, \diamond)$ results in the database $\Delta' = (E', R)$ where E' is defined by:

- If $(\mathtt{v} \diamond v_E(\varphi)) = \mathtt{n}$, $E' = E \setminus \{\langle \varphi, v_E(\varphi) \rangle\}$
- Otherwise, $E' = (E \setminus \{\langle \varphi, v_E(\varphi) \rangle\}) \cup \{\langle \varphi, (\mathtt{v} \diamond v_E(\varphi)) \rangle\}$

We illustrate and comment Definition 7 below.

1. As suggested earlier, standard data integration is expressed by defining $\diamond$ as $\varphi_1 \diamond \varphi_2 = \varphi_1 \oplus \varphi_2$.
2. Considering the connector $\otimes$ instead of $\oplus$ suggests another kind of data integration: instead of cumulating the knowledge as done with $\oplus$, the result of integration can be seen as the '*common* knowledge'. For example, when it comes to integrate $\langle \varphi, \mathtt{t} \rangle$ in the presence of $\langle \varphi, \mathtt{f} \rangle$, the result is $\langle \varphi, \mathtt{n} \rangle$, meaning that φ becomes unknown. Moreover, the integration of $\langle \varphi, \mathtt{t} \rangle$ in the presence of $\langle \varphi, \mathtt{b} \rangle$, results in keeping the former v-pair while eliminating the latter. This way of integrating can be seen as a mean to eliminate cases of inconsistency.
3. However, it could not be suitable to eliminate inconsistency, but on the contrary to preserve it. Namely, in the case above, *i.e.,* when integrating $\langle \varphi, \mathtt{t} \rangle$ in the presence of $\langle \varphi, \mathtt{b} \rangle$, it might be expected that $\langle \varphi, \mathtt{b} \rangle$ be kept. As shown in the right most table of Figure 5, our approach allows to take this case into account by defining a connector $\odot$ as follows:
$$\varphi_1 \odot \varphi_2 = (\varphi_1 \otimes \varphi_2) \oplus (\varphi_1 \otimes \neg\varphi_1) \oplus (\varphi_2 \otimes \neg\varphi_2).$$

It should be emphasized from Definition 7 that it is unlikely that *any* expression $\diamond$ makes sense for defining an integration policy. We however notice that the last item above shows that some sophisticated expressions might be relevant.

Example 5 In the context of our running example, we assume that the sensor H_2 has been replaced with a new one of another type that allows for the additional answer `b` when the degree of humidity has not been determined properly. Notice that this type of output should be distinguished from the absence of answer that

$\varphi_1 \otimes \varphi_2$	t	b	n	f
t	t	t	n	n
b	t	b	n	f
n	n	n	n	n
f	n	f	n	f

$\oplus$

$\varphi_1 \otimes \neg\varphi_1$	
t	n
b	b
n	n
f	n

$\oplus$

$\varphi_2 \otimes \neg\varphi_2$	
t	n
b	b
n	n
f	n

$\varphi_1 \odot \varphi_2$	t	b	n	f
t	t	b	n	n
b	b	b	b	b
n	n	b	n	n
f	n	b	n	f

Fig. 5 Computing the truth table of the expression $\odot$

is understood as a failure. However, since the sensor is new, its output has to be carefully taken into account. This can be modeled by *integrating* the output of the new sensor with the current content of the database (*i.e.,* the output from the old sensor).

As explained above this integration can be done in many different ways, some of which being illustrated below, starting form the database extension E' of Example 4, containing the v-pairs $\langle H_2(101), \mathtt{f}\rangle$ and $\langle H_2(202), \mathtt{t}\rangle$. We also assume that the values returned by the new sensor are: $\langle H_2(101), \mathtt{f}\rangle$, $\langle H_2(202), \mathtt{b}\rangle$ and $\langle H_2(303), \mathtt{t}\rangle$.

Integrating the new values with the existing ones in the standard way using $\oplus$ would yield: $\langle H_2(101), \mathtt{f}\rangle$, $\langle H_2(202), \mathtt{b}\rangle$ and $\langle H_2(303), \mathtt{t}\rangle$, meaning that the new values replace the current ones. However, a more conservative way of integrating the new values is to consider the connector $\otimes$ instead of $\oplus$, which would yield the following: $\langle H_2(101), \mathtt{f}\rangle$ and $\langle H_2(202), \mathtt{t}\rangle$, meaning that the inconsistency returned by the new sensor is not taken into account and that $H_2(303)$ remains unknown.

Although this result could be seen as more 'conservative' than the first one in case of disagreement, it might seem counter-intuitive that the inconsistency is not taken into account. Considering the operator $\odot$ would produce $\langle H_2(101), \mathtt{f}\rangle$ and $\langle H_2(202), \mathtt{b}\rangle$, meaning that the inconsistency is now taken into account and that $H_2(303)$ remains unknown. □

We argue that integrative updates generalize standard updates, because any standard update can be expressed as an integrative update. Indeed, given a database Δ and a fact φ, the following holds:

- The update defined by $\langle \varphi, \mathtt{t}\rangle$ is expressed by the integrative update $(\langle \varphi, \mathtt{t}\rangle, \vee)$.
- The update defined by $\langle \varphi, \mathtt{b}\rangle$ is expressed by the integrative update $(\langle \varphi, \mathtt{b}\rangle, \oplus)$.
- The update defined by $\langle \varphi, \mathtt{n}\rangle$ is expressed by the integrative update $(\langle \varphi, \mathtt{n}\rangle, \otimes)$.
- The update defined by $\langle \varphi, \mathtt{f}\rangle$ is expressed by the integrative update $(\langle \varphi, \mathtt{f}\rangle, \wedge)$.

5 Related Work

Comparing our approach with all related work in the literature is simply not possible due to the huge amount of papers on these topics that have been published during the past four or five decades... In what follows, we mainly focus on the most

related approaches dealing with (*i*) logic and databases, (*ii*) inconsistent databases, (*iii*) multi-valued logic.

Logic and Databases. We first refer to [1,2,14] for surveys of standard approaches to Datalog databases, while in [11] the problem of negation is overviewed in more details. It is important to recall that in all these work, CWA is assumed, thus leading to difficulties in handling falsity, a problem that does not arise in our framework, which assumes OWA instead of CWA.

Changing from CWA to OWA is not new [15] and the need has appeared due to the emergence of data integration on the web. This is so because in this framework, when a piece of information has not been retrieved in the answer to a query, this cannot be seen as that this piece of information is *false*, but rather that this piece of information has not been searched properly. It is thus more appropriate that this piece of information be assigned the truth value *unknown*.

On the other hand, the examples in this paper suggest that when integrating information from several sources, contradictions may occur, thus motivating for the introduction of *inconsistent* as a truth value. This point of view has also been considered in [16] but in a logical framework that differs from ours. Indeed, in [16], the underlying four valued logic is not the one in [3], although the considered implication looks similar to FDE implication. Morevover, in [16] the authors consider two negations in the context of CWA and propose an alternating strategy for computing the database semantics, inspired from the strategy in [17] with well-founded semantics.

The work in [5] is much closer to our approach than that in [16] because the underlying logic in [5] is that in [3]. However, the reader is referred to the previous sections regarding some main differences between the approach in [5] and ours. Among these differences, we mention the form of the rules and the semantic operator that in [5] makes rule heads false when so is the body, whereas in our approach, the truth value is not changed. Related work following this policy of head assignment to false can be found in [18,19] where, in the context of relational databases, reasoning with four truth values is modeled as reasoning *twice* under two truth values: once to deduce true information and once to deduce false information (inconsistency being information obtained in the two ways of reasoning). However, the context of the work in [18,19] differs from ours and that in [5] because in [18,19], implications express equality-generating or tuple-generating-constraints instead of rules.

It is also important to recall that the issue of deductive database updating was first addressed in [20], and then by many other authors among which we cite [13], which was the first approach suggesting to store false facts and to give priority to most recent updates. The present work builds upon these basic ideas in a much wider context.

Inconsistent Databases. Regarding related work on inconsistent databases, we propose a radically different approach. Indeed, the purpose of previous work dealing with contradictions in databases, is either to define and investigate 'repairs' so as to make the database consistent ([21,22]), and/or to identify a set of queries whose answer is independent from any contradiction ([23]). Instead, we propose an approach in which inconsistent information can be stored or deduced through rules, and our purpose is not to eliminate or avoid contradictions.

Indeed, our semantics allows for handling inconsistent information as such, thus reflecting real world applications in which true, false, inconsistent and unknown information have to be dealt with, as is the case when data integration is involved. In doing so, we follow the position in [24], in that inconsistent information should not be avoided, but treated as such by taking appropriate actions when necessary. The issue of taking actions lies beyond the scope of this paper, because our rules cannot express an information such as '*If* φ *is inconsistent then* ϕ'. Indeed in our formalism such a rule would be expressed as $\phi \leftarrow \mathbf{B}\varphi$, which is not allowed, but which is the subject of our current research.

The approach in [25] addresses the issue of data inconsistency due to data integration according to a specific scenario. In [25], the authors consider that the information consists of facts that a central server collects from autonomous sources and then tries to combine, using rules that follow the syntax and the semantics of [5], and a set of *hypotheses* H, representing the server's own estimates. In this setting, the authors show how to compute what they call the *support of* H, defined as the maximal part of H that does not contradict the facts in the database semantics. This notion of support has then been shown to provide hypothesis-based semantics for the class of programs defined in [5], and in the case of Datalog^{neg} programs, these semantics have been shown to extend well-founded semantics of [17] and Kripke Kleen semantics of [26].

Multi-valued Logic. The Four-valued logic that we consider in this work has been introduced in [3] and then has motivated many research efforts in the community of research in non standard logic. Again, our aim is not to review all these work, and we refer to [27] for a nice review of this topic. Here, we focus on those work that are the most closely related to ours and that have already been cited in many places. In [7] the issue of the functional completeness has been addressed among others and their result has of course inspired our concern on this issue, related to FDE implication. On the other hand, the bi-lattice structure of this logic has been widely studied in [5], where the concept of logic programs in this framework was first introduced. We recall that the semantics of the rules in [5] is different from ours in that in [5], the head is set to false when the body is false, whereas in our approach, the truth value of the head is not changed in this case. We argue in this respect that our approach follows standard approaches in that implications whose body is not valid are valid, implying that truth values of the head have not to be changed.

More recently, in [9], an implication slightly different than FDE implication (that we have formerly denoted by $\hookrightarrow$) has ben proposed, and a strong relationship between this logic and rough set theory has been established. We recall that it has been shown in [6] that our approach works with this implication as well, although FDE implication has been chosen in the present paper.

6 Conclusion

In this paper we have introduced a novel approach to deductive databases dealing with contradictory information. We stress again that this work is motivated by the facts that (i) many contradictions occur in the real world and these contradictions must be dealt with as such, and (ii) data integration is a field where such contradictions are common. To cope with this issue we consider a deductive database

approach based on the Four-valued logic initially introduced in [3]. Our database semantics follows FDE implication and has been shown slightly different from that of [5]. We also recall that in this paper, rules whose head is a *negative* literal are allowed and we have shown that contradicting rules could be safely taken into account in our context. Another important contribution of this work is to propose a new kind of update that allows to 'combine' the expected truth value of a fact with its current truth value in the database. This updating policy is of particular interest when it comes to *integrate* new pieces of information in a given database.

Based on the results reported in this paper, we are investigating the following issues. First, as rules can contradict each other (a situation which frequently happens in real life), it is important to characterize the exact situations when these contradictions happen and if so, which actions have to be taken, as suggested in [24]. We are investigating this important issue by extending the form of the rules to allow in their body additional connectors introduced [9] (such as connector **B** recalled in Section 2). Another important extension of this work is the investigation of an *algebraic* language that would allow for the definition of a *generic* framework and the expression of *constraints* on data such as functional dependencies or tuple generating dependencies. Last but not least, based on such an algebra, we strongly believe that the Four-valued framework provides an elegant and efficient tool for defining a new query language devoted to *data integration* rather than to data querying or updating. The notion of *integrative updates* as defined in Section 4, will be the starting point of this future work.

References

1. Ceri S, Gottlob G, Tanca L. Logic Programming and Databases. Surveys in computer science. Springer, 1990. ISBN 3-540-51728-6. URL `http://www.worldcat.org/oclc/20595273`.
2. Garcia-Molina H, Ullman JD, Widom J. Database systems - the complete book (2. ed.). Pearson Education, 2009. ISBN 978-0-13-187325-4.
3. Belnap ND. A Useful Four-Valued Logic. In: Dunn JM, Epstein G (eds.), Modern Uses of Multiple-Valued Logic. Springer Netherlands, Dordrecht. ISBN 978-94-010-1161-7, 1977 pp. 5–37. doi:10.1007/978-94-010-1161-7. URL `https://doi.org/10.1007/978-94-010-1161-7`.
4. Batay YL. Maintaining Grain Quality During Storage and Transport. In: Cereal Grains, Assessing and Managing Quality, Second Edition. Woodhead Publishing Series in Food Science, Technology and Nutrition, 2017 pp. 571–590.
5. Fitting M. Bilattices and the Semantics of Logic Programming. *J. Log. Program.*, 1991. **11**(1&2):91–116. doi:10.1016/0743-1066(91)90014-G. URL `https://doi.org/10.1016/0743-1066(91)90014-G`.
6. Laurent D. 4-Valued Semantics Under the OWA: A Deductive Database Approach. In: Flouris G, Laurent D, Plexousakis D, Spyratos N, Tanaka Y (eds.), Information Search, Integration, and Personalization - 13th International Workshop, ISIP 2019, Heraklion, Greece, May 9-10, 2019, Revised Selected Papers, volume 1197 of *Communications in Computer and Information Science*. Springer, 2019 pp. 101–116. doi:10.1007/978-3-030-44900-1_7. URL `https://doi.org/10.1007/978-3-030-44900-1\_7`.
7. Arieli O, Avron A. The Value of the Four Values. *Artif. Intell.*, 1998. **102**(1):97–141. doi:10.1016/S0004-3702(98)00032-0. URL `https://doi.org/10.1016/S0004-3702(98)00032-0`.
8. Hazen AP, Pelletier FJ. K3, L3, LP, RM3, A3, FDE: How to Make Many-Valued Logics Work for You. *CoRR*, 2017. **abs/1711.05816**. 1711.05816, URL `http://arxiv.org/abs/1711.05816`.
9. Tsoukiàs A. A first-order, four valued, weakly paraconsistent logic and its relation to rough sets semantics. *Foundations of Computing and Decision Sciences*, 2002. **12**:85–108.

10. Reiter R. On Closed World Data Bases. In: Gallaire H, Minker J (eds.), Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, France, 1977, Advances in Data Base Theory. Plemum Press, New York, 1977 pp. 55–76. doi:10.1007/978-1-4684-3384-5_3. URL https://doi.org/10.1007/978-1-4684-3384-5_3.
11. Bidoit N. Negation in Rule-Based Database Languages: A Survey. *Theor. Comput. Sci.*, 1991. **78**(1):3–83. doi:10.1016/0304-3975(51)90003-5. URL https://doi.org/10.1016/0304-3975(51)90003-5.
12. Laurent D, Luong VP, Spyratos N. The Use of Deleted Tuples in Database, Querying and Updating. *Acta Inf.*, 1997. **34**(12):905–925. doi:10.1007/s002360050111. URL https://doi.org/10.1007/s002360050111.
13. Alves MHF, Laurent D, Spyratos N. Update Rules in Datalog Programs. *J. Log. Comput.*, 1998. **8**(6):745–775. doi:10.1093/logcom/8.6.745. URL https://doi.org/10.1093/logcom/8.6.745.
14. Minker J, Seipel D, Zaniolo C. Logic and Databases: A History of Deductive Databases. In: Siekmann JH (ed.), Computational Logic, volume 9 of *Handbook of the History of Logic*, pp. 571–627. Elsevier, 2014. doi:10.1016/B978-0-444-51624-4.50013-7. URL https://doi.org/10.1016/B978-0-444-51624-4.50013-7.
15. Bergman M. The Open World Assumption: Elephant in the room. In AI3:::Adaptative Information. www.mkbergman.com/852/the-open-world-assumption-elephant-in-the-room, 2009. Online; accessed 22 April 2020.
16. de Amo S, Pais MS. A paraconsistent logic programming approach for querying inconsistent databases. *Int. J. Approx. Reason.*, 2007. **46**(2):366–386. doi:10.1016/j.ijar.2006.09.009. URL https://doi.org/10.1016/j.ijar.2006.09.009.
17. Gelder AV, Ross KA, Schlipf JS. The Well-Founded Semantics for General Logic Programs. *J. ACM*, 1991. **38**(3):620–650. doi:10.1145/116825.116838. URL https://doi.org/10.1145/116825.116838.
18. Grahne G, Moallemi A. A useful four-valued database logic. In: Desai BC, Flesca S, Zumpano E, Masciari E, Caroprese L (eds.), Proceedings of the 22nd International Database Engineering & Applications Symposium, IDEAS 2018, Villa San Giovanni, Italy, June 18-20, 2018. ACM, 2018 pp. 22–30. doi:10.1145/3216122.3216157. URL https://doi.org/10.1145/3216122.3216157.
19. Grahne G, Moallemi A. Universal (and Existential) Nulls. *Fundam. Inform.*, 2019. **167**(4):287–321. doi:10.3233/FI-2019-1819. URL https://doi.org/10.3233/FI-2019-1819.
20. Reiter R. On Formalizing Database Updates: Preliminary Report. In: Pirotte A, Delobel C, Gottlob G (eds.), Advances in Database Technology - EDBT'92, 3rd International Conference on Extending Database Technology, Vienna, Austria, March 23-27, 1992, Proceedings, volume 580 of *Lecture Notes in Computer Science*. Springer, 1992 pp. 10–20. doi:10.1007/BFb0032420. URL https://doi.org/10.1007/BFb0032420.
21. Afrati FN, Kolaitis PG. Repair checking in inconsistent databases: algorithms and complexity. In: Database Theory - ICDT, 12th International Conference, Russia, March 23-25, 2009, Proceedings. 2009 pp. 31–41.
22. Greco G, Greco S, Zumpano E. A Logical Framework for Querying and Repairing Inconsistent Databases. *IEEE Trans. Knowl. Data Eng.*, 2003. **15**(6):1389–1408. doi:10.1109/TKDE.2003.1245280. URL https://doi.org/10.1109/TKDE.2003.1245280.
23. Greco S, Molinaro C, Trubitsyna I. Computing Approximate Query Answers over Inconsistent Knowledge Bases. In: Lang J (ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. ijcai.org, 2018 pp. 1838–1846. doi:10.24963/ijcai.2018/254. URL https://doi.org/10.24963/ijcai.2018/254.
24. Gabbay D, Hunter A. Making inconsistency respectable: A logical framework for inconsistency in reasoning, part I — A position paper. In: Jorrand P, Kelemen J (eds.), Fundamentals of Artificial Intelligence Research. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-38420-5, 1991 pp. 19–32.
25. Loyer Y, Spyratos N, Stamate D. Hypothesis-based semantics of logic programs in multivalued logics. *ACM Trans. Comput. Log.*, 2004. **5**(3):508–527. doi:10.1145/1013560.1013565. URL https://doi.org/10.1145/1013560.1013565.
26. Fitting M. A Kripke-Kleene Semantics for Logic Programs. *J. Log. Program.*, 1985. **2**(4):295–312. doi:10.1016/S0743-1066(85)80005-4. URL https://doi.org/10.1016/S0743-1066(85)80005-4.

27. Omori H, Wansing H. 40 years of FDE: An Introductory Overview. *Studia Logica*, 2017. **105**(6):1021–1049. doi:10.1007/s11225-017-9748-6. URL `https://doi.org/10.1007/s11225-017-9748-6`.