Deductive databases in four-valued logic: rule semantics and models

Dominique Laurent, Nicolas Spyratos

Manucript submitted to Journal of Logic and Computation

Deductive Databases in Four-Valued Logic: Rule Semantics and Models

Dominique Laurent · Nicolas Spyratos

Abstract In this paper, we investigate rule semantics for deductive databases in the context of Four-valued logic. In our approach a database is a pair $\Delta = (E, R)$, where E is a set of pairs, each pair associating a ground fact with a truth value (thus allowing to store true, false or inconsistent facts) and R is a set of rules generalizing standard Datalog rules in the following sense: (i) the head of a rule can be a positive or a negative atom, and (ii) the body can involve any among the connectors of Four-valued logic. We define the database semantics as the least fixed point of a monotonic operator and we compare this semantics with that of k-existential programs defined by Fitting and paraconsistent extended logic programs defined by Arieli.

Our main contribution is to show that, if we consider rules as implications (that is, if we view the database as a set of formulas) then the semantics of the database is the unique minimal model of the set of database formulas. Here, minimality is understood with respect to the knowledge ordering of Four-valued logic satisfying a monotonicity property whereby the truth value of the head of an instantiated rule is greater than that of the body. Moreover, we address the issue of updating databases with finite semantics in the context of Four-valued logic. We argue that our approach allows for a new kind of updates, in which the update result depends not only on the fact involved in the update, but also on its current truth value in the database.

Keywords Four-valued logic . Deductive database . Least fix point . Rule semantics . Database model . Database update

Dominique Laurent
ETIS Laboratory - ENSEA, CY Cergy Paris University, CNRS
F-95000 Cergy-Pontoise, France
dominique.laurent@cyu.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 extend the notion of deductive databases [10,17] to the context of Four-valued logic [7], a formalism known to be suitable for data integration, as it allows to deal with *unknown*, *inconsistent*, *true* and *false* information.

In our approach, a database is a pair $\Delta = (E, R)$ where E is the database extension and R is a set of rules. More precisely:

- the database extension E is a set of pairs of the form $\langle \varphi, \mathtt{v} \rangle$, where φ is a ground fact and $\mathtt{v}$ its truth value, thus allowing to store true, false or inconsistent facts
- the rules generalize standard Datalog rules in the sense that: (i) the head of a rule can be a positive or negative fact, and (ii) the body can involve any among the connectors of Four-valued logic, namely $\neg$, $\vee$, $\wedge$, $\oplus$ and $\otimes$.

Database semantics is defined as the least fixed point of a monotonic operator, as done for paraconsistent extended logic programs defined in [3] and for k-existential programs defined by Fitting [14]. A comparison between our semantics and the semantics in [3] and [14] is discussed in detail later in the paper. Before going into technical details, let us motivate our approach through an example that we shall use as our running example throughout the paper.

Running Example. Consider bags of rice grains that are transported from rice farms to warehouses where they are stored. Two important factors that (among others) influence the storage of rice, are color and humidity of the rice grains [6]. Each bag is tested for color and humidity in two different sites, first just before leaving the rice farm and then just before entering the warehouse.

Assuming that each bag has a unique identifier, the bag with identifier ID is associated with two atomic formulas, namely $W\_Fit(ID)$ and $H\_Fit(ID)$ whose truth values respectively account for its fitness with respect to 'whiteness' and for its fitness with respect to 'humidity'. To pass the 'whiteness' criterion, *i.e.,* to ensure that $W\_Fit(ID)$ is *true*, the tests should say that the grains in the bag are indeed white, while to pass the 'humidity' criterion, *i.e.,* to ensure that $H\_Fit(ID)$ is *true*, the tests should say that the grains in the bag are *not* humid.

Then, the overall fitness for a bag with identifier ID, denoted by $Fit(ID)$, is assessed through the conjunction $W\_Fit(ID) \wedge H\_Fit(ID)$, which is generalized as the rule $Fit(x) \leftarrow W\_Fit(x) \wedge H\_Fit(x)$, where x is a variable.

Assume now that the tests are conducted by sensors: two sensors at the rice farm, one for color, denoted W_1, and one for humidity denoted H_1; and two sensors at the warehouse denoted W_2 and H_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 color (and similarly for the sensors testing humidity):

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.

Then the question is: what value should we assign to $W\_Fit(ID)$ in each of the four cases above? In our approach, we answer this question by 'integrating' the outputs of W_1 and W_2 as follows (and similarly for the outputs of H_1 and H_2):

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

It should be clear that we need more than the standard truth values, `True` and `False`, to express cases 2 and 4 above. It turns out that the Four-valued logic introduced in [7] is actually the right formalism as it provides the additional truth values needed and also the appropriate connectors to work with these additional truth values. For instance, using a connector denoted by $\oplus$ we can express all four cases above as a single expression: $W_1(ID) \oplus W_2(ID)$. We can therefore generalize this situation through the rule $W\_Fit(x) \leftarrow W_1(x) \oplus W_2(x)$, where x is a variable. The situation regarding humidity is expressed by means of a similar rule whose head involves negation, namely $\neg H\_Fit(x) \leftarrow H_1(x) \oplus H_2(x)$. □

In the light of this example, our contributions are the following:

1. We define the formalism for expressing rules as discussed above, and then we define database semantics as the least fixed point of a monotonic operator. Moreover, we compare our semantics with that of k-existential programs as defined by Fitting in [14,15], and that of paraconsistent extended logic programs as defined by Arieli in [3].
2. We show that if one considers the rules as implications then database semantics is the *unique minimal model* of the set of implications. Here minimality is understood with respect to the knowledge ordering of the Four-valued logic satisfying a monotonicity property whereby the truth value of the head of an instantiated rule is greater than that of the body. Our notion of minimality is then compared with that in [3].
3. We characterize databases having finite semantics by means of the concept of *safe rule*. This is an important issue as it guarantees that all answers to queries are finite (except if we query for unknown facts). This is an issue *not* addressed in [14,15] or [3].
4. We investigate the issue of database updating in the context of Four-valued logic. This issue is not addressed in [14,15] or in [3], although knowledge revision is considered in [3]. We argue that the context of Four-valued logic allows for defining a new kind of updates by which the result of an update depends not only on the fact involved in the update, but also on its current truth value in the database content.

The paper is organized as follows: In Section 2 we review the formalism related to Four-valued logic. Section 3 provides the definitions of syntax and semantics of databases in the context of Four-valued logic; in this section, we also investigate properties of our semantics, in particular, in relation with [14,15] and [3]. In Section 4, we address the issue of safe rules, to guarantee finite database semantics. Section 5 is devoted to database updates in the context of our approach. In Section 6, we review some of the numerous approaches related to our work. Finally, in Section 7, we provide an overview of our results and outline directions for further research.

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

2 Background: Four-Valued Logic

2.1 Basics of Four-Valued Logic

Four-valued logic was introduced by Belnap in [7], who argued that this formalism could be of interest when integrating data from various data sources. To this end, denoting by t, b, n and f the four truth values, the usual connectives $\neg$, $\vee$ and $\wedge$ have been defined as shown in Figure 1. A prominent feature of this Four-valued logic is that truth values can be compared according to two partial orderings, known as *truth ordering* and *knowledge ordering*, denoted by $\preceq_t$ and $\preceq_k$, respectively, and defined as follows:

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

We recall from [7] that the notation b and n for *inconsistent* and *unknown*, can be read respectively as *both* and *none*. This is because considering the set $\{\mathtt{t}, \mathtt{b}, \mathtt{n}, \mathtt{f}\}$ as the power set of $\{\mathtt{True}, \mathtt{False}\}$, the truth values n, f, t, b are respectively associated with $\emptyset$, $\{\mathtt{False}\}$, $\{\mathtt{True}\}$, $\{\mathtt{True}, \mathtt{False}\}$. We note that, under this association, the ordering $\preceq_k$ and the connectors $\oplus$ and $\otimes$ are respectively set theoretic inclusion, union and intersection applied to the power set of $\{\mathtt{True}, \mathtt{False}\}$.

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, 7, 14, 15] 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.

It should be emphasized that, not surprisingly, some basic properties holding in standard logic do not hold in this Four-valued logic. 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 [4, 23, 32] that defining the implication $\Phi_1 \Rightarrow \Phi_2$ by $\neg\Phi_1 \vee \Phi_2$, is problematic.

To see this, as done in [4, 7, 23, 32], the *distinguished truth values* in our approach, are t and b, meaning that distinguished truth values are those that 'contain some truth'. Then, a given propositional formula Φ is said to be *valid* for a given truth value assignment of its propositional variables if the corresponding truth value of Φ is distinguished.

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

$\rightarrow_1$	t	b	n	f
t	t	b	n	f
b	t	b	n	f
n	t	t	t	t
f	t	t	t	t

$\rightarrow_2$	t	b	n	f
t	t	b	n	f
b	t	t	n	n
n	t	b	t	b
f	t	t	t	t

$\rightarrow_3$	t	b	n	f
t	t	b	n	f
b	t	b	n	f
n	b	b	b	b
f	b	b	b	b

$\overset{*}{\rightarrow}_1$	t	b	n	f
t	t	f	n	f
b	t	b	n	f
n	t	n	t	n
f	t	t	t	t

$\overset{*}{\rightarrow}_2$	t	b	n	f
t	t	f	f	f
b	t	t	f	f
n	t	f	t	f
f	t	t	t	t

$\overset{*}{\rightarrow}_3$	t	b	n	f
t	b	f	f	f
b	b	b	f	f
n	b	f	b	f
f	b	b	b	b

Fig. 2 Truth tables of $\Rightarrow$ and of implications in Four-Valued Logic

Considering now the formula Φ defined by $(\Phi_1 \wedge (\Phi_1 \Rightarrow \Phi_2)) \Rightarrow \Phi_2$, Figure 2 shows that if v is a truth value assignment such that $v(\Phi_1) = \mathtt{n}$ and $v(\Phi_2) = \mathtt{f}$, then $v(\Phi_1 \Rightarrow \Phi_2) = \mathtt{n}$ and thus, $v(\Phi) = \mathtt{n}$. Hence in this case, Φ is *not* valid, which is counter-intuitive, because it is expected that Φ should be valid in any case. This remark leads to discard $\Rightarrow$ as an implication.

2.2 Implication in Four-Valued Logic

Although rule based languages and implication are two distinct concepts, we recall that, in standard logic programming, there is a strong link between the two. Indeed, it is well known that the semantics of a positive logic program (*i.e.,* a set of rules with no negation) is the least model of the set of first order formulas obtained by considering the rules as implications. In order to set a similar result in our framework, we need to define an implication and, as our earlier discussion shows, this implication cannot be $\Rightarrow$.

Among the various proposals introduced in the literature (see [4,23,29,32]), we mention six implications whose truth tables are shown in Figure 2. Considering $\rightsquigarrow$ as a generic symbol standing for any of the symbols $\rightarrow_i$ or $\overset{*}{\rightarrow}_i$ $(i = 1, 2, 3)$, let $\overline{\Phi}$ be the formula $(\Phi_1 \rightsquigarrow (\Phi_1 \rightsquigarrow \Phi_2)) \rightsquigarrow \Phi_2$. Then, contrary to Φ, the formula $\overline{\Phi}$ has truth value $\mathtt{t}$ of $\mathtt{b}$ for all truth value assignments of Φ_1 and Φ_2. In other words, $\overline{\Phi}$ is *always valid*, whatever the truth values assigned to Φ_1 and Φ_2. Thus, all six implications $\rightarrow_i$ and $\overset{*}{\rightarrow}_i$ for $i = 1, 2, 3$, are acceptable regarding the validity of $\overline{\Phi}$.

Although our purpose in this paper is not to assess which implication could be 'better' than another, it is nevertheless relevant to recall some of their basic properties. In doing so, we shall argue that, in our context, for every $i = 1, 2, 3$, the implication $\rightarrow_i$ has suitable properties, whereas such is *not* the case for $\overset{*}{\rightarrow}_i$.

First, it is well known that none of these implications can be expressed using the standard connectors $\neg$, $\vee$ and $\wedge$ (see Corollary 9 in [4] regarding $\rightarrow_1$), and that this becomes possible when considering the additional connectors available in the Four-valued logic. To see this, considering as in [32] the basic connectors $\neg$, $\vee$, $\wedge$, $\not\rightarrow$, $\oplus$ and $\otimes$, the connector $\sim$ is defined for every formula ϕ by:

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

Moreover, the additional connectors **T**, **B**, **N** and **F** are introduced in [32], with truth tables as shown in Figure 3. These connectors allow to 'characterize' each

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

ϕ	$\diamond\phi$
t	n
b	n
n	b
f	b

Fig. 3 Truth tables of unary connectors from [32] and of connectors $\circ$ and $\diamond$

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 $\mathbf{V}$, 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.

Then, considering that two propositional formulas ϕ_1 and ϕ_2 are *equivalent*, denoted by $\phi_1 \equiv \phi_2$, if they have the same truth tables, the following equivalences are shown in [32]:

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

Now, let the additional connectors $\circ$ and $\diamond$ be defined by:

$$\circ\phi = \neg(\mathbf{T}(\phi) \vee \mathbf{B}(\phi)) \qquad \text{and} \qquad \diamond\phi = \not\sim \neg(\mathbf{T}(\phi) \vee \mathbf{B}(\phi)).$$

It turns out that $\circ$ 'characterizes' the non validity of a formula ϕ in terms of the truth values t and f. In other words, as shown by Figure 3, $\circ\phi$ is true if ϕ is not valid and false otherwise. The connector $\diamond$ expresses a similar formula, using truth values b and n instead of t and f.

These connectors provide the following simple formulas expressing the implications whose truth tables are shown in Figure 2.

1. $\phi_1 \rightarrow_1 \phi_2 \equiv \circ\phi_1 \vee \phi_2$ (easy to prove based on the truth tables)
2. $\phi_1 \rightarrow_2 \phi_2 \equiv \sim \phi_1 \vee \phi_2$ (see [32] for the proof of this result)
3. $\phi_1 \rightarrow_3 \phi_2 \equiv \diamond\phi_1 \oplus \phi_2$ (easy to prove based on the truth tables)
4. $\phi_1 \stackrel{*}{\rightarrow}_1 \phi_2 \equiv (\phi_1 \rightarrow_1 \phi_2) \wedge (\neg\phi_2 \rightarrow_1 \neg\phi_1)$
5. $\phi_1 \stackrel{*}{\rightarrow}_2 \phi_2 \equiv (\phi_1 \rightarrow_2 \phi_2) \wedge (\neg\phi_2 \rightarrow_2 \neg\phi_1)$
6. $\phi_1 \stackrel{*}{\rightarrow}_3 \phi_2 \equiv (\phi_1 \rightarrow_3 \phi_2) \wedge (\neg\phi_2 \rightarrow_3 \neg\phi_1)$

The last three items above show that, for $i = 1, 2, 3$, $\stackrel{*}{\rightarrow}_i$ has been introduced due to the fact that $\rightarrow_i$ does not satisfy the appealing property of contraposition of implication, namely $\phi_1 \rightarrow_i \phi_2$ and $\neg\phi_2 \rightarrow_i \neg\phi_1$ are *not* equivalent. It is of course easy to see that for $i = 1, 2, 3$, $\stackrel{*}{\rightarrow}_i$ satisfies the property of contraposition of implication, *i.e.,* the equivalence $\phi_1 \stackrel{*}{\rightarrow}_i \phi_2 \equiv \neg\phi_2 \stackrel{*}{\rightarrow}_i \neg\phi_1$ holds.

On the other hand, since $\circ\phi$ can be read as ϕ *is not valid*, the first item above shows that $\phi_1 \rightarrow_1 \phi_2$ can be read as *either* ϕ_1 *is not valid or* ϕ_2, as mentioned in [3]. A similar remark holds regarding $\rightarrow_3$, using $\not\sim \neg$ instead of $\neg$ as negation, and $\oplus$ instead of $\vee$ as disjunction. As for $\rightarrow_2$, it is argued in [32] that $\sim \phi$ can be read as the 'complement' of ϕ, and so, $\phi_1 \rightarrow_2 \phi_2$ can be read as *either the complement of* ϕ_1 *or* ϕ_2.

For the purpose of our work, we emphasize that, as Figure 2 shows, the implications $\phi_1 \rightarrow_i \phi_2$, for $i = 1, 2, 3$, are valid in the *same* cases, namely when ϕ_1 and ϕ_2 are valid or when ϕ_1 is not valid. This remark shows that, regarding validity, the implications $\rightarrow_i$ behave in a similar way as $\Rightarrow$ does in the context of two-valued logic (that is, $\phi \Rightarrow \phi_2$ is true if and only if ϕ_1 and ϕ_2 are true or ϕ_1 is false). However, this remark does *not* hold for the implications $\stackrel{*}{\rightarrow}_i$, $i = 1, 2, 3$.

As we shall see later (Definition 5), the notion of database model relies only on the *validity* of implications.It therefore follows from the above discussion that choosing one implication among $\rightarrow_1$, $\rightarrow_2$ or $\rightarrow_3$ makes no difference in the results obtained in our approach (see Section 4): these results hold no matter which of these implications is chosen. However, the above discussion also shows that the implications $\stackrel{*}{\rightarrow}_i$, $i = 1, 2, 3$, have to be discarded in the context of our approach.

Therefore, in this paper we choose $\rightarrow$ as a generic symbol for implication. One can replace it by any of the symbols $\rightarrow_1$, $\rightarrow_2$ or $\rightarrow_3$, with no effect on the results.

3 Four-Valued Logic and Databases

3.1 Database Syntax

As usual, when dealing with deductive databases, the alphabet is made up of constants, variables and predicate symbols with a fixed arity. We thus assume a fixed set of constants, called the *Herbrand Universe* and denoted by $\mathcal{H}$. Note that $\mathcal{H}$ may be infinite.

As in traditional approaches, a *term* is either a constant from $\mathcal{H}$ or a variable, and an *atomic formula* (also called *atom*) is a formula of the form $P(t_1, t_2, \ldots, t_k)$ where P is a k-ary predicate and t_i is a term, for every $i = 1, 2, \ldots, k$. A formula is said to be a *ground formula* 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 $\mathcal{H}$, denoted by $\mathcal{HB}$, is the set of all facts that can be built up using the constants in $\mathcal{H}$ and the predicates. Clearly, if $\mathcal{H}$ is infinite, then so is $\mathcal{HB}$.

In this setting, a *valuation* (or interpretation) of $\mathcal{HB}$ is defined to be a truth value assignment associating every fact in $\mathcal{HB}$ with a truth value in $\{\mathtt{t}, \mathtt{b}, \mathtt{n}, \mathtt{f}\}$. More formally, a valuation is a total function $v : \mathcal{HB} \rightarrow \{\mathtt{t}, \mathtt{b}, \mathtt{n}, \mathtt{f}\}$, and for every fact φ in $\mathcal{HB}$, $v(\varphi)$ is referred to as the *truth value of φ under v*. The set $S_v = \{\langle \varphi, \mathtt{v} \rangle \mid v(\varphi) \neq \mathtt{n}\}$ is called the *known part of v*.

Conversely, let S be a set of pairs of the form $\langle \varphi, \mathtt{v} \rangle$ where φ is in $\mathcal{HB}$ and $\mathtt{v}$ is a truth value. Then, based on S, we can define a valuation v_S if the following assumptions hold:
(1) $\mathtt{v}$ is different than $\mathtt{n}$, *i.e.,* $\mathtt{v}$ is in $\{\mathtt{t}, \mathtt{b}, \mathtt{f}\}$,
(2) for all $\langle \varphi_1, \mathtt{v}_1 \rangle$ and $\langle \varphi_2, \mathtt{v}_2 \rangle$ in S, if $\varphi_1 = \varphi_2$ then $\mathtt{v}_1 = \mathtt{v}_2$,
(3) for every φ not occurring in S, $v_S(\varphi) = \mathtt{n}$.

Such set S is called a *valuated-set*, or *v-set* for short. As for the above assumptions, we emphasize the following:

1. Assumption (1) means that if a fact φ appears in S then it is in the known part of v_S, *i.e.,* $v_S(\varphi) \neq \mathtt{n}$.
2. Assumption (2) means that every fact φ occurring in S is associated with *one and only one* truth-value (which according to (1), belongs to $\{\mathtt{t}, \mathtt{b}, \mathtt{f}\}$). We shall refer to this assumption as the 'functionality' of S.
3. Assumption (3) means that every fact φ not occurring in S is 'unknown', *i.e.,* $v_S(\varphi) = \mathtt{n}$. We shall refer to this assumption as the 'Open World Assumption' (or OWA for short).

We are now ready to define the concept of database in Four-valued logic.

Definition 1 A database over $\mathcal{HB}$ is a pair $\Delta = (E, R)$ such that:

- E is a finite v-set over $\mathcal{HB}$ called the *extension* of Δ.
- R is a set of rules called the *rule set* of Δ. A rule in R is an expression of the form $\rho : h(X) \leftarrow B(X, Y)$ where X and Y are vectors of variables such that all variables in X occur in $B(X, Y)$. Moreover:
  - $h(X)$ is a positive or negative literal
  - $B(X, Y)$ is a quantifier free, well formed formula built up using the connectors $\neg$, $\vee$, $\wedge$, $\oplus$ and $\otimes$.

Given a rule $\rho : h(X) \leftarrow B(X, Y)$, $h(X)$ is called the *head* of ρ, denoted by $head(\rho)$, and $B(X, Y)$ is called the *body* of ρ, denoted by $body(\rho)$. □

It should be clear that our definition of database generalizes that of standard Datalog^{neg} databases ([9,10]) in significant ways. Indeed:

- Contrary to Datalog databases, the database extension E is not a set of ground facts, but a *v-set*. Therefore E stores not only true facts, but also false facts or inconsistent facts. Moreover, contrary to Datalog databases, we allow the predicate symbols occurring in E to also occur in the rule heads. In other words we do not assume that the set of predicate symbols is partitioned into a set of *extensional* and a set of *intentional* predicate symbols.
- As in Datalog, all variables occurring in the head of a rule also occur in the body. However, in our approach, the rule body may involve any among the connectors $\neg$, $\vee$, $\wedge$, $\oplus$ and $\otimes$, and the head may be a positive or a *negative* literal.
- Our definition generalizes rules as defined in [24] where the bodies of the rules are restricted to be conjunctions only.

Before investigating the close relationship between our approach and the works in [14,15] and [3], we illustrate Definition 1 in the context of our Running Example.

Example 1 In our Running Example, let us consider the set of predicate symbols $Pred = \{H_1, H_2, W_1, W_2, H\_Fit, W\_Fit, Fit, Species, Alert\}$, the set of constants $\mathcal{H} = \{101, 202, 303, s_1, s_2, low, high\}$ and the corresponding Herbrand base $\mathcal{HB}$. Moreover, let us define the database $\Delta = (E, R)$ as follows:

$$
\begin{aligned}
E = \{\ & \langle W_1(101), \mathtt{t}\rangle, \langle W_1(202), \mathtt{f}\rangle, \langle W_2(202), \mathtt{t}\rangle, \langle W_1(303), \mathtt{f}\rangle, \\
& \langle H_1(101), \mathtt{f}\rangle, \langle H_2(101), \mathtt{f}\rangle, \langle H_2(202), \mathtt{t}\rangle, \\
& \langle Species(101, s_1), \mathtt{t}\rangle, \langle Species(202, s_1), \mathtt{t}\rangle, \langle Species(303, s_2), \mathtt{t}\rangle, \\
& \langle Fragile(s_1, low), \mathtt{t}\rangle, \langle Fragile(s_2, high), \mathtt{t}\rangle \qquad\qquad \}
\end{aligned}
$$

$R = \{\rho_1, \rho_1', \rho_2, \rho_2', \rho_3, \rho_3', \rho_4\}$ where:

$$
\begin{aligned}
&\rho_1 :\ W\_Fit(x) \leftarrow W_1(x) \oplus W_2(x) \qquad \rho_1' : \neg W\_Fit(x) \leftarrow \neg(W_1(x) \oplus W_2(x)) \\
&\rho_2 : \neg H\_Fit(x) \leftarrow H_1(x) \oplus H_2(x) \qquad \rho_2' :\ H\_Fit(x) \leftarrow \neg(H_1(x) \oplus H_2(x)) \\
&\rho_3 :\ Fit(x) \leftarrow W\_Fit(x) \wedge H\_Fit(x) \\
&\rho_3' :\ \neg Fit(x) \leftarrow \neg(W\_Fit(x) \wedge H\_Fit(x)) \\
&\rho_4 : Alert(x, z) \leftarrow \neg Fit(x) \otimes (Species(x, y) \wedge Fragile(y, z))
\end{aligned}
$$

We refer to the introductory section regarding the intuition for the rules ρ_1, ρ_2 and ρ_3. Regarding their 'dual' versions ρ_1', ρ_2' and ρ_3', these rules are introduced because negation $\neg$ in Four-valued logic is explicit, in the sense that $\neg\varphi$ is inferred only if

either this piece of information is given in the database extension, or if it is *explicitly* obtained through the rules. Thus, if no rule head involves, say $\neg W\_Fit(x)$, for some instance ID of x then it is not possible to state that $\neg W\_Fit(ID)$ is true (*i.e.,* that $W\_Fit(ID)$ is false). In fact, considering ρ_1 and ρ'_1 amounts to considering these rules as an *equivalence* that is $W\_Fit(ID)$ holds if and only if so does $W_1 \oplus W_2$. We shall come back to this important point later in the paper, when comparing our approach with those in [14, 15] and [3].

Regarding ρ_4, the intuition is that an alert should be raised in case of an unFit bag whose content is fragile, and moreover the level of the alert is set to be that of species fragility (*high* or *low* in our example).

More precisely, based on the truth tables shown in Figure 1, it turns out that the formula $\neg Fit(ID) \otimes (Species(ID, s) \wedge Fragile(s, l))$ has truth value $\mathtt{t}$ when the truth value of $\neg Fit(ID)$ is $\mathtt{t}$ or $\mathtt{b}$ and that of $Species(ID, s) \wedge Fragile(s, l)$ is $\mathtt{t}$. Moreover, if facts as $Species(ID, s)$ or $Fragile(s, l)$ are:
− either true (*i.e.,* the pairs $\langle Species(ID, s), \mathtt{t}\rangle$ or $\langle Fragile(s, l), \mathtt{t}\rangle$ occur in E)
− or unknown (*i.e.,* nothing has been stored regarding these facts),
then the formula $Species(ID, s) \wedge Fragile(s, l)$ has truth value $\mathtt{t}$ if and only if $Species(ID, s)$ and $Fragile(s, l)$ both have truth value $\mathtt{t}$.

In other words, the intuition behind rule ρ_4 is that an alert of level l is raised for the bag ID and species s if the following holds: (i) the grains are unFit or an inconsistent measure was made, and (ii) the grains are of species s whose fragility level is l. Notice in this respect that raising such an alert needs to know that $Fit(ID)$ is false, which is achieved through the rule ρ'_3, which in turn needs the rules ρ'_1 and ρ'_2 to be properly triggered. □

We end this section by observing that, from a technical point of view, the database extension E is actually a means for the finite representation of a potentially infinite function over $\mathcal{HB}$ under the assumptions seen earlier in the definition of v_E (namely the functionality of E and OWA).

3.2 Database Semantics

In our approach, as for Datalog, database semantics is defined as the least fixed point of a monotonic operator. The definition of this operator relies on the following extensions of the orderings $\preceq_k$ and $\preceq_t$ to the family of all v-sets over $\mathcal{HB}$.

Definition 2 For all v-sets S_1 and S_2 over $\mathcal{HB}$, $S_1 \preceq_k S_2$, respectively $S_1 \preceq_t S_2$, holds if for every φ in $\mathcal{HB}$, $v_{S_1}(\varphi) \preceq_k v_{S_2}(\varphi)$, 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.

We now state the definition of the operator allowing for the computation of database semantics. In this definition as well as in the remainder of the paper, $\bigoplus$ denotes the least upper bound with respect to $\preceq_k$.

Definition 3 Given a set of rules R, let $inst(R)$ be the set of all instantiations of the rules in R based on the atoms in $\mathcal{HB}$. For every v-set V and every φ in $\mathcal{HB}$, let $B^+(\varphi)$ and $B^-(\varphi)$, be respectively defined by:

- $B^+(V,\varphi) = \{\rho \in inst(R) \mid head(\rho) = \varphi \wedge v_V(body(\rho)) \in \{\mathtt{t},\mathtt{b}\}\}$
- $B^-(V,\varphi) = \{\rho \in inst(R) \mid head(\rho) = \neg\varphi \wedge v_V(body(\rho)) \in \{\mathtt{t},\mathtt{b}\}\}$.

The *semantic immediate consequence operator* associated with a database $\Delta = (E,R)$, denoted by Ψ_Δ, associates a v-set V with the set $\Psi_\Delta(V)$ containing all pairs $\langle\varphi,\mathtt{v}\rangle$ where φ is in $\mathcal{HB}$ and $\mathtt{v}$ is a truth value different than $\mathtt{n}$ defined by:

$$\begin{aligned}\mathtt{v} = v_E(\varphi) \oplus & \bigoplus\{v_V(body(\rho)) \mid \rho \in B^+(V,\varphi)\} \\ \oplus & \bigoplus\{\neg v_V(body(\rho)) \mid \rho \in B^-(V,\varphi)\}.\end{aligned}$$

□

Notice that if $B^+(V,\varphi) \cup B^-(V,\varphi) = \emptyset$ (*i.e.,* if there is no instantiated rule whose body is valid in V and whose head involves φ) and if φ does not occur in E then the content of Δ does not provide any relevant piece of information regarding φ. In this case, it is intuitively justified to state that the truth value of φ in $\Psi_\Delta(V)$ is $\mathtt{n}$. This is precisely what is done according to Definition 3 because, in this case, no pair involving φ is inserted in $\Psi_\Delta(V)$.

It should also be emphasized that, according to Definition 3, for every v-set V, $\Psi_\Delta(V)$ is a v-set in which, for every $\langle\varphi,\mathtt{v}\rangle$ in E, $\Psi_\Delta(V)$ contains $\langle\varphi,\mathtt{v}'\rangle$ such that $\mathtt{v} \preceq_k \mathtt{v}'$. The following lemma shows a basic property of the operator Ψ_Δ.

Lemma 1 *For every $\Delta = (E,R)$, let $\left(\Psi^i\right)_{i\geq 0}$ be the sequence defined by*

$\Psi^0 = \emptyset$, $\Psi^n = \Psi_\Delta(\Psi^{n-1})$ *for $n \geq 1$, and $\Psi^\lambda = \bigoplus_{\alpha<\lambda}\Psi^\alpha$ if λ is a limit ordinal.*

The sequence $\left(\Psi^i\right)_{i\geq 0}$ has a limit which is the unique least fixed point of Ψ_Δ with respect to $\preceq_k$. Moreover, the least fixed point is reached for an ordinal at most ω.

Proof First, we notice that the connectors involved in standard rules 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). Therefore Ψ_Δ is monotonic with respect to $\preceq_k$, which by the Knaster-Tarski theorem implies that the sequence $\left(\Psi^i\right)_{i\geq 0}$ has a unique least fixed point. Moreover, since the only infinitary operator in our computation is $\bigoplus$, Ψ_Δ can be shown to be continuous as in [14]. Therefore, the ordinal for which the least fixed point is reached is at most ω. □

We are now ready to define what we call *database semantics* in our approach.

Definition 4 The *semantics* of a given database $\Delta = (E,R)$, denoted by $sem(\Delta)$, is defined as the least fixed point of the sequence $\left(\Psi^i\right)_{i\geq 0}$.

The valuation defined by the v-set $sem(\Delta)$ is denoted by v_Δ and for every φ in $\mathcal{HB}$, $v_\Delta(\varphi)$ is referred to as the *truth value* of φ in Δ. □

We note that according to the above definition, the truth value of the head is changed only if that of the body is designated (either $\mathtt{t}$ or $\mathtt{b}$), as Arieli does in [3], and contrary to Fitting in [14,15], who does so also when the body is false. However, in our approach, if the body is inconsistent, the head is inconsistent as well, whereas in [3] it is set to *true* in this case. A more detailed comparison is provided in the next sub-section.

The following example illustrates Definitions 3 and 4 and Lemma 2 in the context of our running example.

Example 2 Continuing with our running example, let $\Delta = (E, R)$ be the database as defined in Example 1. Then the v-set $sem(\Delta)$ is computed as follows, starting from $\Psi^0 = \emptyset$ and denoting by $v_i(\varphi)$ the truth value of φ in Ψ^i for every $i \geq 0$:

(1) For every φ in $\mathcal{HB}$, we have $v_0(\varphi) = \mathtt{n}$, and thus for every ρ in $inst(R)$, $v_0(body(\rho)) = \mathtt{n}$. Hence, $B^+(\Psi_0, \varphi) = B^-(\Psi_0, \varphi) = \emptyset$, and $\Psi^1 = \Psi_\Delta(\Psi^0) = E$.

(2) Computing $\Psi^2 = \Psi_\Delta(\Psi^1)$, we first consider ρ_1 and ρ_1', which are the only rules whose head involves W_Fit.

• For $\varphi = W\_Fit(101)$, $v_1(W_1(101) \oplus W_2(101)) = \mathtt{t} \oplus \mathtt{n} = \mathtt{t}$, thus $B^+(\Psi^1, \varphi) = \{W\_Fit(101) \leftarrow W_1(101) \oplus W_2(101)\}$, and $B^-(\Psi^1, \varphi) = \emptyset$, as $v_1(\neg(W_1(101) \oplus W_2(101))) = \mathtt{f}$. Hence $\langle W\_Fit(101), \mathtt{t}\rangle$ is inserted in Ψ^2.

• For $\varphi = W\_Fit(202)$, $v_1(W_1(202) \oplus W_2(202)) = \mathtt{f} \oplus \mathtt{t} = \mathtt{b}$, implying that $B^+(\Psi^1, \varphi) = \{W\_Fit(202) \leftarrow W_1(202) \oplus W_2(202)\}$, and that $B^-(\Psi^1, \varphi) = \{\neg W\_Fit(202) \leftarrow \neg(W_1(202) \oplus W_2(202))\}$. So $\langle W\_Fit(202), \mathtt{b}\rangle$ is inserted in Ψ^2.

• For $\varphi = W\_Fit(303)$, $v_1(W_1(303) \oplus W_2(303)) = \mathtt{f} \oplus \mathtt{n} = \mathtt{f}$, thus $B^+(\Psi^1, \varphi) = \emptyset$, and $B^-(\Psi^1, \varphi) = \{\neg W\_Fit(303) \leftarrow \neg(W_1(303) \oplus W_2(303))\}$. So, $\langle W\_Fit(303), \mathtt{f}\rangle$ is inserted in Ψ^2.

Considering now ρ_2 and ρ_2', which are the only rules whose head involves H_Fit, we have the following:

• For $\varphi = H\_Fit(101)$, $v_1(H_1(101) \oplus H_2(101)) = \mathtt{f} \oplus \mathtt{f} = \mathtt{f}$, thus $B^+(\Psi^1, \varphi) = \{H\_Fit(101) \leftarrow \neg(H_1(101) \oplus H_2(101))\}$, and $B^-(\Psi^1, \varphi) = \emptyset$. Hence $\langle H\_Fit(101), \mathtt{t}\rangle$ is inserted in Ψ^2.

• For $\varphi = H\_Fit(202)$, $v_1(H_1(202) \oplus H_2(202)) = \mathtt{n} \oplus \mathtt{t} = \mathtt{t}$, thus we obtain that $B^+(\Psi^1, \varphi) = \emptyset$, and that $B^-(\Psi^1, \varphi) = \{\neg H\_Fit(202) \leftarrow H_1(202) \oplus H_2(202)\}$. Hence $\langle H\_Fit(202), \mathtt{f}\rangle$ is inserted in Ψ^2.

• For $\varphi = H\_Fit(303)$, $v_1(H_1(303) \oplus H_2(303)) = \mathtt{n} \oplus \mathtt{n} = \mathtt{n}$, thus $B^+(\Psi^1, \varphi) = B^-(\Psi^1, \varphi) = \emptyset$, which implies no modification of Ψ^2.

Since no other rule can be applied at this stage, we have:

$$\Psi^2 = E \cup \{\langle W\_Fit(101), \mathtt{t}\rangle, \langle W\_Fit(202), \mathtt{b}\rangle, \langle W\_Fit(303), \mathtt{f}\rangle, \langle H\_Fit(101), \mathtt{t}\rangle, \langle H\_Fit(202), \mathtt{f}\rangle\}.$$

(3) When computing $\Psi^3 = \Psi_\Delta(\Psi^2)$, ρ_3 and ρ_3' are the only rules involving atoms occurring in Ψ^2 and not in Ψ^1. The computations are as follows:

• For $\varphi = Fit(101)$, $v_2(W\_Fit(101) \wedge H\_Fit(101)) = \mathtt{t} \wedge \mathtt{t} = \mathtt{t}$, thus $B^+(\Psi^2, \varphi) = \{Fit(101) \leftarrow W\_Fit(101) \wedge H\_Fit(101)\}$, and $B^-(\Psi^2, \varphi) = \emptyset$. Hence $\langle Fit(101), \mathtt{t}\rangle$ is inserted in Ψ^3.

• For $\varphi = Fit(202)$, $v_2(W\_Fit(202) \wedge H\_Fit(202)) = \mathtt{b} \wedge \mathtt{f} = \mathtt{f}$, thus we obtain that $B^+(\Psi^1, \varphi) = \emptyset$, and that $B^-(\Psi^1, \varphi) = \{\neg Fit(202) \leftarrow \neg(W\_Fit(202) \wedge H\_Fit(202))\}$. Hence $\langle Fit(202), \mathtt{f}\rangle$ is inserted in Ψ^3.

• For $\varphi = Fit(303)$, $v_2(W\_Fit(303) \wedge H\_Fit(303)) = \mathtt{f} \wedge \mathtt{n} = \mathtt{f}$, thus $B^+(\Psi^1, \varphi) = \emptyset$ and $B^-(\Psi^1, \varphi) = \{\neg Fit(303) \leftarrow \neg(W\_Fit(303) \wedge H\_Fit(303))\}$. Hence $\langle Fit(303), \mathtt{f}\rangle$ is inserted in Ψ^3. We therefore obtain:

$\Psi^3 = \Psi^2 \cup \{\langle Fit(101), \mathtt{t}\rangle, \langle Fit(202), \mathtt{f}\rangle, \langle Fit(303), \mathtt{f}\rangle\}$.

(4) When computing $\Psi^4 = \Psi_\Delta(\Psi^3)$, ρ_4 is the only rule applying to the atoms occurring in Ψ^3 and not in Ψ^2, and this rule generates ground atoms of the form $Alert(ID, l)$. In this case, for every such atom, $B^-(Alert(ID, l), \Psi_3)$ is always empty. On the other hand, since for all ID and l, $\neg Fit(ID) \otimes (Species(ID, s) \wedge Fragile(s, l))$ is valid if and only if all involved literals are valid, the only cases for which $B^+(Alert(ID, l), \Psi_3)$ is not empty are when the triple (ID, s, l) is either $(202, s_1, low)$ or $(303, s_2, high)$. The corresponding computations are as follows:
• For $(202, s_1, low)$ thus for $\varphi = Alert(202, low)$, $v_3(\neg Fit(101) \otimes (Species(202, s_1) \wedge Fragile(s_2, low))) = \mathtt{t} \otimes (\mathtt{t} \wedge \mathtt{t}) = \mathtt{t}$. Thus, $\langle Alert(202, low), \mathtt{t}\rangle$ is inserted in Ψ^4.
• Similar computations for $\varphi = Alert(303, high)$ lead to insert $\langle Alert(303, high), \mathtt{t}\rangle$ in Ψ^4. We therefore obtain:

$\Psi^4 = \Psi^3 \cup \{\langle Alert(202, low), \mathtt{t}\rangle, \langle Alert(303, high), \mathtt{t}\rangle\}$.

(5) As no further computations can generate new elements based on Ψ^4 above, we obtain that $sem(\Delta) = \Psi^4$. □

The following proposition states basic properties of our database semantics.

Proposition 1 *Given a database $\Delta = (E, R)$:*

1. $sem(\Delta)$ is a v-set such that $E \preceq_k sem(\Delta)$.
2. For every rule ρ in $inst(R)$ such that $v_\Delta(body(\rho))$ is in $\{\mathtt{t}, \mathtt{b}\}$, it holds that $v_\Delta(body(\rho)) \preceq_k v_\Delta(head(\rho))$.

Proof 1. $sem(\Delta)$ is a v-set because the definition of Ψ_Δ implies that it is not possible for a ground atom φ to occur in two distinct pairs in $sem(\Delta)$. Moreover, $v_E(\varphi) \preceq_k v_\Delta(\varphi)$ also holds for every φ as a consequence of the definition of Ψ_Δ.
2. Let ρ be in $inst(R)$ such that $v_\Delta(body(\rho))$ is in $\{\mathtt{t}, \mathtt{b}\}$. Let i_0 be the least index such that the truth value $\mathtt{v}_b$ of $body(\rho)$ in Ψ^{i_0} is $\mathtt{t}$ or $\mathtt{b}$ when computing $sem(\Delta)$. By Definition 3, if $\mathtt{v}_h$ denotes the truth value of $head(\rho)$ in Ψ^{i_0}, we have $\mathtt{v}_b \preceq_k \mathtt{v}_h$, and by monotonicity of Ψ_Δ, this holds in $sem(\Delta)$, which completes the proof. □

We emphasize that, by Proposition 1(1), we have $E \preceq_k sem(\Delta)$, which intuitively means that the semantics *extends* the knowledge provided by the database extension E. This also means that our semantics gives priority to the rules over the extension. It should be noticed that such is not the case in most approaches to deductive databases. Indeed, in standard deductive approaches [10], as well as in [14], the atoms in E are restricted to involve *extensional* predicate symbols, that is predicate symbols *not* allowed to occur in the heads of the rules. On the other hand, although extensional predicate symbols are not considered in [24], priority is nevertheless given to the database extension, because instantiated rules whose head involves an atom occurring in E are not 'triggered'. Notice however that considering such restrictions in the context of the present work is possible and raises no difficulty.

3.3 Comparison with [14,15] and with [3]

Considering first [14,15], our approach is closely related to what is called *k-existential programs*, where the rules are expressions of the form $h(X) \leftarrow body(X, Y)$ such that:

– The head $h(X)$ is a *positive* literal whose free variables in X all occur in $body(X, Y)$. It is also assumed that the variables in X are the only free variables in $body(X, Y)$.
– The right hand-side $body(X, Y)$ is a well-formed formula involving $\neg$, $\wedge$, $\vee$, $\otimes$, $\oplus$ and the quantifier $\sum$ (where the truth value of $(\sum Y)(\Phi(Y))$ is defined as the least upper-bound with respect to $\preceq_k$ of the truth values of all instances of $\Phi(Y)$).
– It is not allowed that the heads of two distinct rules in a given program involve the same predicate symbol.

It turns out that, with respect to rules as defined in Definition 1, the main differences are as follows:

1. We allow positive *and* negative literals in the heads of the rules, whereas negative literals are not allowed in the head of the rules in [14].
2. We do not restrict different rules to involve different predicate symbols in their heads, as done in [14].
3. However, the quantifier $\sum$ is allowed in the bodies of the rules in [14], which is not the case in our approach. Notice in this respect that our approach implicitly assumes that in a rule $h(X) \leftarrow body(X, Y)$, the variables in Y are bounded to a quantifier $\sum$. That is, according to [14], the rule would be written as $h(X) \leftarrow (\sum Y)\, body(X, Y)$.

Additionally to these syntactical differences, rules are not processed in [14,15] as we do in our approach. Indeed, although database semantics is defined as the least fixed point of a monotonic operator with respect to $\preceq_k$, the two operators behave in different ways, given an instantiated rule $\varphi \leftarrow \Phi$ and a v-set V:

– According to [14,15], the operator assigns φ the truth value in V of Φ.
– In our approach, assuming that there is no other rule whose head is φ, φ is assigned the truth value in V of Φ *only if this truth value is designated* (*i.e.,* `t` or `b`). Otherwise the truth value of ϕ is set to `n`.

This difference in computing database semantics is very important because it implies that the intuition behind the rules is different. Indeed, roughly speaking, according to [14,15], rules are seen as *equivalences* since the head and the body are assigned the same truth value (assuming no interaction between the rules), whereas in our approach, rules are considered as *implications*, as will be discussed in the next section.

To illustrate the difference, let $Student(Bob) \leftarrow PhD\_Student(Bob)$ be an instantiated rule whose intuition is obvious. If it is known in the database that $PhD\_Student(Bob)$ has truth value `f` then, according to [14,15], $Student(Bob)$ has also truth value `f`, which is counter intuitive. In our approach, the truth value of $Student(Bob)$ is `n`, which better fits the intuition.

Comparing now our approach with that in [3], we notice the following syntactical similarities and differences:

– According to [3], rule heads may involve positive or negative literals, as we do in our approach.
– However, in [3], rule bodies are restricted to only involve the connectors $\neg$ and $\wedge$ as for Datalog^{neg} rules [10], whereas in our approach more connectors are allowed.

– In [3], as in our approach, but contrary to [14,15], several rules involving the same predicate are allowed.

Regarding semantics, rules are triggered in [3] according to the same policy as in our approach, *but* in the following different way: given an instantiated rule $\varphi \leftarrow \Phi$ and a v-set V, if the truth value in V of Φ is designated (*i.e.,* $\mathtt{t}$ or $\mathtt{b}$), then the truth value of φ is set to $\mathtt{t}$. Otherwise the truth value of φ is set to $\mathtt{n}$.

To see the impact of the above way of handling rules, consider again the instantiated rule $Student(Bob) \leftarrow PhD\_Student(Bob)$. As above, if it is known in the database that $PhD\_Student(Bob)$ has truth value $\mathtt{f}$ then, according to [3], the truth value of $Student(Bob)$ is $\mathtt{n}$, as done in our approach.

On the other hand, assume that a problem regarding Bob's status has lead to assign $PhD\_Student(Bob)$ the truth value $\mathtt{b}$. Then, according to [3], the truth value of $Student(Bob)$ is set to $\mathtt{t}$, whereas, according to our approach, $Student(Bob)$ is assigned the truth value $\mathtt{b}$, which we think, better fits the intuition.

To end the comparison with [3], we note that [3] addresses two important issues not addressed in the present paper, namely (i) the semantics of negation as failure in the context of Four-valued logic, and (ii) prioritized logic programs. We do not discuss these issues any further as they lie outside the scope of this paper.

We end the section by providing an example to illustrate the three approaches more thoroughly than done above.

Example 3 We illustrate below the main differences between the three approaches using very simple examples. In doing so we denote by $sem_F(\Delta)$ and $sem_A(\Delta)$ the semantics of a database Δ according to Fitting's and Arieli's semantics, respectively. The explanations on how the semantics is obtained are omitted, since they are very easy.

Case 1. Let $\Delta_1 = (\{\langle q, \mathtt{f}\rangle\}, \{p \leftarrow q\})$. In this case, $sem_F(\Delta_1) = \{\langle q, \mathtt{f}\rangle, \langle p, \mathtt{f}\rangle\}$, $sem_A(\Delta_1) = \{\langle q, \mathtt{f}\rangle\}$, and $sem(\Delta_1) = \{\langle q, \mathtt{f}\rangle\}$.
Case 2. Let $\Delta_2 = (\{\langle q, \mathtt{b}\rangle\}, \{p \leftarrow q\})$. In this case, $sem_F(\Delta_2) = \{\langle q, \mathtt{b}\rangle, \langle p, \mathtt{b}\rangle\}$, $sem_A(\Delta_2) = \{\langle q, \mathtt{b}\rangle, \langle p, \mathtt{t}\rangle\}$, and $sem(\Delta_2) = \{\langle q, \mathtt{b}\rangle, \langle p, \mathtt{b}\rangle\}$.
Case 3. Let $\Delta_3 = (\{\langle q, \mathtt{t}\rangle\}, \{\neg p \leftarrow q\})$. In this case, $sem_F(\Delta_3)$ is undefined, $sem_A(\Delta_3) = \{\langle q, \mathtt{t}\rangle, \langle p, \mathtt{f}\rangle\}$, and $sem(\Delta_3) = \{\langle q, \mathtt{t}\rangle, \langle p, \mathtt{f}\rangle\}$.
Case 4. Let $\Delta_4 = (\{\langle q, \mathtt{t}\rangle\}, \{p \leftarrow \neg q\})$. In this case, $sem_F(\Delta_4) = \{\langle q, \mathtt{t}\rangle, \langle p, \mathtt{f}\rangle\}$, $sem_A(\Delta_4) = \{\langle q, \mathtt{t}\rangle\}$, and $sem(\Delta_4) = \{\langle q, \mathtt{t}\rangle\}$.
Case 5. Let $\Delta_5 = (\{\langle q, \mathtt{f}\rangle\}, \{p \leftarrow q \oplus r\})$. In this case, $sem_F(\Delta_5) = \{\langle q, \mathtt{f}\rangle, \langle p, \mathtt{f}\rangle\}$, $sem_A(\Delta_5)$ is undefined, and $sem(\Delta_5) = \{\langle q, \mathtt{f}\rangle\}$.
Case 6. Let $\Delta_6 = (\{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle\}, \{p \leftarrow q \oplus r\})$. In this case, $sem_F(\Delta_6) = \{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle, \langle p, \mathtt{b}\rangle\}$, $sem_A(\Delta_6)$ is undefined, and $sem(\Delta_6) = \{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle, \langle p, \mathtt{b}\rangle\}$.
Case 7. Let $\Delta_7 = (\{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle\}, \{p \leftarrow q, p \leftarrow r\})$. In this case, $sem_F(\Delta_7)$ is undefined, $sem_A(\Delta_7) = \{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle, \langle p, \mathtt{t}\rangle\}$, and $sem(\Delta_7) = \{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle, \langle p, \mathtt{t}\rangle\}$.
Case 8. Let $\Delta_8 = (\{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle\}, \{p \leftarrow q \vee r\})$. In this case, $sem_F(\Delta_8) = \{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle, \langle p, \mathtt{t}\rangle\}$, $sem_A(\Delta_8)$ is undefined, and $sem(\Delta_8) = \{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle, \langle p, \mathtt{t}\rangle\}$.
Case 9. Let $\Delta_9 = (\{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle\}, \{p \leftarrow q \vee r, p \leftarrow q\})$. In this case, $sem_F(\Delta_9)$ is undefined, $sem_A(\Delta_9)$ is undefined, and $sem(\Delta_9) = \{\langle q, \mathtt{t}\rangle, \langle r, \mathtt{f}\rangle, \langle p, \mathtt{t}\rangle\}$.

To illustrate a case with a recursive set of rules with unstratified negation, let $R = \{p \leftarrow \neg p\}$. First, if $\Delta = (\emptyset, R)$, it is easy to see that $sem_F(\Delta) = sem_A(\Delta) = sem(\Delta) = \emptyset$, because the rule is not triggered.

If $\Delta = (\{\langle p, \mathtt{t}\rangle\}, R)$ then $sem_F(\Delta)$ is not defined because according to [14, 15], Δ contains two rules with head equal to p. On the other hand, we have $sem_A(\Delta) = sem(\Delta) = \{\langle p, \mathtt{t}\rangle\}$. Simlarly, it is easy to see that if $\Delta = (\{\langle p, \mathtt{f}\rangle\}, R)$ then $sem_F(\Delta)$ is not defined, whereas $sem_A(\Delta) = sem(\Delta) = \{\langle p, \mathtt{b}\rangle\}$. □

4 Database Semantics vs. Database Models

4.1 Database Models

Based on the above discussion about k-existential clauses as defined in [14], every rule $\rho : h(X) \leftarrow B(X, Y)$ in our approach is associated with the following formula in the Four-valued logic, denoted by $\rho^{\rightarrow}$:

$\rho^{\rightarrow} : (\forall X)\,((\sum Y)\,(B(X, Y) \rightarrow h(X)))$.

We recall that the implication $\rightarrow$ in $\rho^{\rightarrow}$ is generic, that is it can be replaced by any of the three implications $\rightarrow_1$, $\rightarrow_2$ or $\rightarrow_3$ (see our discussion at the end of Section 2). The notion of database *model* is now defined as follows.

Definition 5 Given $\Delta = (E, R)$, a v-set M is a *model of* Δ if:
− $E \preceq_k M$, and
− for every ρ in R, $\rho^{\rightarrow}$ is valid in M (*i.e.,* $v_M(\rho^{\rightarrow})$ is $\mathtt{t}$ or $\mathtt{b}$).

A model M is said to be *k-minimal* if there does not exist a model M' such that $M' \neq M$ and $M' \preceq_k M$.

A model M is said to be *k-increasing* if for every ρ in $inst(R)$ such that $v_M(body(\rho))$ is in $\{\mathtt{t}, \mathtt{b}\}$, $v_M(body(\rho)) \preceq_k v_M(head(\rho))$. □

To state that $sem(\Delta)$ is the *unique k-minimal and k-increasing model* of Δ, we show the following lemma.

Lemma 2 *If M_1 and M_2 are two v-sets, let us denote by $M_1 \otimes M_2$ the v-set M such that for every φ in $\mathcal{HB}$, $v_M(\varphi) = v_{M_1}(\varphi) \otimes v_{M_2}(\varphi)$.*

Given $\Delta = (E, R)$, if M_1 and M_2 are k-increasing models of Δ then $M = M_1 \otimes M_2$ is also a k-increasing model of Δ.

Proof For every $\langle \varphi, \mathtt{v}\rangle$ in E, we have $v_E(\varphi) \preceq_k v_{M_i}(\varphi)$ for $i = 1, 2$. Thus, $v_E(\varphi) \preceq_k v_{M_1}(\varphi) \otimes v_{M_2}(\varphi)$, that is $v_E(\varphi) \preceq_k v_M(\varphi)$ holds. Therefore, $E \preceq_k M$ holds.

Let $\rho : h(X) \leftarrow B(X, Y)$ be in R and $h(\alpha) \leftarrow B(\alpha, \beta)$ an instance of ρ such that $v_M(body(\rho))$ is in $\{\mathtt{t}, \mathtt{b}\}$. Since for $i = 1, 2$, $M \preceq_k M_i$, $v_{M_i}(body(\rho))$ is in $\{\mathtt{t}, \mathtt{b}\}$. Hence, for $i = 1, 2$, $v_{M_i}(B(\alpha, \beta)) \preceq_k v_{M_i}(h(\alpha))$ holds because M_i is k-increasing. On the other hand, since the connectors involved in $B(\alpha, \beta)$ are monotonic with respect to $\preceq_k$, $v_M(B(\alpha, \beta)) \preceq_k v_{M_i}(B(\alpha, \beta))$ for $i = 1, 2$. We thus obtain for $i = 1, 2$, $v_M(B(\alpha, \beta)) \preceq_k v_{M_i}(B(\alpha, \beta)) \preceq_k v_{M_i}(h(\alpha))$, hence that $v_M(B(\alpha, \beta)) \preceq_k v_{M_1}(h(\alpha)) \otimes v_{M_2}(h(\alpha))$. That is $v_M(B(\alpha, \beta)) \preceq_k v_M(h(\alpha))$.

To show that $\rho^{\rightarrow}$ is valid in M, we notice that $\phi_1 \rightarrow \phi_2$ is valid in M if either ϕ_1 is not valid in M, or otherwise, if $v_M(\phi_1) \preceq_k v_M(\phi_2)$ (this can be seen from the truth tables of $\rightarrow_i$ for $i = 1, 2, 3$, as shown in Figure 2). Hence, for every ρ in R, all instances of $\rho^{\rightarrow}$ are valid in M, that is for all instances α and β of X and Y respectively, $B(\alpha, \beta) \rightarrow h(\alpha)$ is valid in M. This implies that $(\sum Y)(B(\alpha, Y)) \rightarrow h(\alpha)$ is valid in M (because by the truth table of $\oplus$ in Figure 1, the 'sum' of valid formulas is a valid formula). Since the truth table of $\wedge$ in Figure 1 shows that

the conjunction of valid formulas is valid, $(\forall X)((\sum Y)(B(X,Y) \rightarrow h(X)))$, that is $\rho^{\rightarrow}$, is valid in M. Thus M is a k-increasing model of Δ, and the proof is complete. □

Proposition 2 *Given a database $\Delta = (E, R)$, $sem(\Delta)$ is the unique k-minimal and k-increasing model of Δ.*

Proof We first show that $sem(\Delta)$ is a k-increasing model of Δ. First, for every φ occurring in E, $v_E(\varphi) \preceq_k v_\Delta(\varphi)$ holds because of Proposition 1(1). For every $\rho : h(X) \leftarrow B(X,Y)$ in R, $\rho^{\rightarrow}$ is shown to be valid in $sem(\Delta)$ in the same way as, in the previous proof, $\rho^{\rightarrow}$ has been shown to be valid in M. Since Proposition 1(2) shows that $sem(\Delta)$ is k-increasing, $sem(\Delta)$ is a k-increasing model of Δ.

Regarding k-minimality, let M be a k-increasing model of Δ such that $M \prec_k sem(\Delta)$. Then, for every ground atom φ, $v_M(\varphi) \preceq_k v_\Delta(\varphi)$ and there exists at least one ground atom φ_0 such that $v_M(\varphi_0) \neq v_\Delta(\varphi_0)$. This implies that every such atom φ_0 is such that $v_\Delta(\varphi_0) \neq \mathtt{n}$, and so, that φ_0 occurs in $sem(\Delta)$.

We now argue that every such φ_0 occurs in the head of at least one rule in $inst(R)$ such that $v_\Delta(body(\rho))$ is in $\{\mathtt{t}, \mathtt{b}\}$. Indeed, otherwise, φ_0 would occur in E and we would have $v_\Delta(\varphi_0) = v_E(\varphi_0)$. As M is a model of Δ, we have $v_E(\varphi_0) \preceq_k v_M(\varphi_0)$, and so $v_\Delta(\varphi_0) \preceq_k v_M(\varphi_0)$, which is a contradiction.

Hence, given such an atom φ_0, let i_0 be the least index such that $v_M(\varphi_0) \prec_k v_{i_0}(\varphi_0)$, where v_{i_0} denotes the interpretation induced by Ψ^{i_0}. Since $\Psi^0 \preceq_k M$ (because $\Psi^0 = \emptyset$), it must be that $i_0 > 0$, meaning that $\Psi^{i_0} = \Psi_\Delta(\Psi^{i_0-1})$ and $\Psi^{i_0-1} \preceq_k M$. Moreover, by definition of Ψ_Δ, $v_{i_0}(\varphi_0)$ is written as:

$$v_{i_0}(\varphi_0) = v_E(\varphi_0) \oplus \bigoplus\nolimits_{\rho \in B^+(\Psi^{i_0-1}, \varphi_0)} (v_{i_0-1}(body(\rho))) \oplus \bigoplus\nolimits_{\rho \in B^-(\Psi^{i_0-1}, \varphi_0)} (\neg v_{i_0-1}(body(\rho)))$$

where v_{i_0-1} is the interpretation induced by Ψ^{i_0-1}. Since for every $body(\rho)$ involved in the above expression we have $v_{i_0-1}(body(\rho)) \preceq_k v_\Delta(body(\rho)) \preceq_k v_M(body(\rho))$ and since we assume that M satisfies $v_M(body(\rho)) \preceq_k v_M(head(\rho))$, we obtain $v_{i_0-1}(\varphi_0) \preceq_k v_M(head(\rho))$. Since $head(\rho)$ is either φ_0 (if $\rho \in B^+(\varphi_0)$) or $\neg\varphi_0$ (if $\rho \in B^-(\varphi_0)$), we have $v_{i_0-1}(\varphi_0) \preceq_k v_M(\varphi_0)$ if $\rho \in B^+(\varphi_0)$ and, if $\rho \in B^-(\varphi_0)$, $v_{i_0-1}(\varphi_0) \preceq_k v_M(\neg\varphi_0)$, which is equivalent to $\neg v_{i_0-1}(\varphi_0) \preceq_k v_M(\varphi_0)$ (since for all truth values $\mathtt{v}_1$ and $\mathtt{v}_2$, $\mathtt{v}_1 \preceq_k \mathtt{v}_2$ holds if and only if $\neg\mathtt{v}_1 \preceq_k \neg\mathtt{v}_2$ holds).

Therefore, $v_{i_0}(\varphi_0) \preceq_k v_E(\varphi_0) \oplus v_M(\varphi_0)$, and as $v_E(\varphi_0) \preceq_k v_M(\varphi_0)$ (because M is a model of Δ), it follows that $v_{i_0}(\varphi_0) \preceq_k v_M(\varphi_0)$. This is a contradiction with our assumption that $v_M(\varphi_0) \prec_k v_{i_0}(\varphi_0)$, and so, $sem(\Delta)$ is k-minimal.

To show the unicity of $sem(\Delta)$, let M be another k-minimal and k-increasing model of Δ. By Lemma 2, $M' = sem(\Delta) \otimes M$ is also a k-increasing model of Δ, and moreover, we have $M' \preceq_k M$ and $M' \preceq_k sem(\Delta)$. Since M and $sem(\Delta)$ are assumed to be k-minimal, we also have $M \preceq_k M'$ and $sem(\Delta) \preceq_k M'$. Therefore, $M = M' = sem(\Delta)$, and the proof is complete. □

It turns out that Proposition 2 expresses properties of database semantics similar to well-known properties of Datalog databases [10] whereby the semantics of a Datalog database is its unique minimal (with respect to set-inclusion) model, when rules $h(X) \leftarrow B(X,Y)$ are seen as formulas $(\forall X)((\exists Y)(B(X,Y) \Rightarrow h(X)))$.

Regarding the comparison of the truth values of the body and the head of the rules in the case of Datalog databases, when considering the ordering $\preceq_t$, it is easy to see that for all instances α and β of X and Y, respectively, $v_I(B(\alpha,\beta)) \preceq_t v_I(h(\alpha))$ holds whenever $B(\alpha,\beta) \Rightarrow h(\alpha)$ is valid in a two-valued interpretation I.

The result in Proposition 2 shows that, as for k-existential program semantics [14] and paraconsistent logic programs semantics in [3], database semantics in our approach is minimal with respect to the knowledge ordering. However, it should be noticed that:

- In [14] k-minimality is expressed with respect to the set of fix-points of the associated consequence operator, with no consideration of the most generic notion of database model.
- In [3], the results are shown only for the implication denoted here by $\rightarrow_1$, whereas our results hold for $\rightarrow_1$ as well as for two other implications, namely $\rightarrow_2$ and $\rightarrow_3$.
- In [3], the notion of model is similar to ours, but k-minimality is shown in a context more restrictive than our context.
- In [3], the uniqueness of the k-minimal model is shown by minimizing inconsistency, whereas in our approach, the uniqueness of the k-minimal model is shown using a monotonic criterion involving the head and the body of instantiated rules.

4.2 Finite Database Semantics

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 a database $\Delta = (E, R)$ 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 $\mathcal{H}$, $Q(\alpha, \beta) \vee S(\alpha)$ is true if so is $S(\alpha)$. Hence, $sem(\Delta) = \{\langle S(a), \mathtt{t}\rangle\} \cup \{\langle P(a, \beta), \mathtt{t}\rangle \mid \beta \in \mathcal{H}\}$, which is infinite when $\mathcal{H}$ is infinite.

To cope with this difficulty, we define the notion of *safe* rules, inspired by the case of Datalog^{neg} databases [10]. To see how the two 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} database, whose semantics is $\{S(a)\} \cup \{P(a, \beta) \mid \beta \in \mathcal{H}\}$. 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 define safe rules in our context, we need some preliminary definitions and notation as detailed next. First, we adapt the notion of *active domain* in relational databases [17] to our approach as follows. Given a universe $\mathcal{H}$, its associated Herbrand base $\mathcal{HB}$ and a database $\Delta = (E, R)$ over $\mathcal{HB}$, we call *active domain of* Δ, denoted by $\mathcal{A}(\Delta)$, the subset of $\mathcal{H}$ 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 $\mathcal{H}$ is infinite, because E and R are assumed to be finite. We first define the notion of *active domain preserving rule* as follows.

Definition 6 Given a Herbrand base $\mathcal{HB}$, a rule ρ is said to be *active domain preserving* if for every database $\Delta = (E, \{\rho\})$ where E is an arbitrary finite v-set involving facts in $\mathcal{HB}$, $sem(\Delta)$ involves only facts in $\mathcal{AB}(\Delta)$. □

We also adapt the usual notion of *disjunctive normal form* of a formula to the context of Four-valued logic. To this end, we recall from [15,32] 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 an equivalent formula 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 $\vee$. Every Φ_i is called a $\vee$-component of Φ.
2. $\oplus$-*transformation:* 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 $\oplus$. Every Φ_i^j is called a $\oplus$-component of Φ_i.
3. $\wedge$-*transformation:* For every i in $\{1, 2, \ldots, n\}$ and every j in $\{1, 2, \ldots, p_i\}$, Φ_i^j is transformed into its $\wedge$-form $\Phi_i^j \equiv (\phi_1^{ij} \wedge \phi_2^{ij} \wedge \ldots \wedge \phi_{q_{ij}}^{ij})$ where, for k in $\{1, 2, \ldots, q_{ij}\}$, ϕ_k^{ij} does not involve the connector $\wedge$. Every ϕ_k^{ij} is called a $\wedge$-component of Φ_i^j.
4. $\otimes$-*transformation:* For every i in $\{1, 2, \ldots, n\}$, every j in $\{1, 2, \ldots, p_i\}$, and every k in $\{1, 2, \ldots, q_{ij}\}$, ϕ_k^{ij} is transformed into its $\otimes$-form $\phi_k^{ij} \equiv (\lambda_k^1 \otimes \lambda_k^2 \otimes \ldots \otimes \lambda_k^{r_k})$ where, for every l in $\{1, 2, \ldots, r_k\}$, every λ_k^l is a literal, that is of the form φ or $\neg\varphi$ where φ is an atom.

Combining these transformations yields a formula equivalent to Φ, called the *Four-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, that is if and only if at least one $\oplus$-component of Φ is valid in S.

Furthermore, every $\oplus$-component $\Phi_{i_0}^{j_0}$ is valid in S if and only if each of its $\wedge$-component $\phi_k^{i_0 j_0}$ is valid in S, which holds if and only if every literal λ occurring in one of its $\otimes$-components is valid in S. It follows that, for every λ in the $\otimes$-components of $\phi_k^{i_0 j_0}$, $v_S(\lambda)$ is `t` or `b` if $\lambda = \varphi$, and $v_S(\lambda)$ is `f` or `b` if $\lambda = \neg\varphi$.

The syntactic notion of *safe rule* is defined based on Four-normal form as follows.

Definition 7 Given a Herbrand base $\mathcal{HB}$, let $\rho : h(X) \leftarrow B(X, Y)$ be a rule such that $B(X, Y)$ is written in its Four-normal form. Then ρ is said to be *safe* if every $\oplus$-component in $B(X, Y)$ involves at least all variables in X. □

We notice that Φ is valid in S if and only if there exists an $\oplus$-component $\Phi_{i_0}^{j_0}$ in its Four-normal form, 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 provides a syntactic characterization of active domain preserving rules as defined in Definition 6.

Proposition 3 *Let $\rho : h(X) \leftarrow B(X, Y)$ be a rule such that $B(X, Y)$ is written in its Four-normal form. Then ρ is safe if and only if every ρ is active domain preserving.*

Proof Assume first that ρ is not safe. By Definition 7, 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 $X_1 X_2$ 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 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 but might involve facts not in $\mathcal{AB}(\Delta)$. Hence, $inst^*(h(X_1X_2))$ belongs to the semantics of Δ, meaning that, according to Definition 6, ρ is not active domain preserving.

Conversely, assume that ρ is safe. By Definition 7, for every i in $\{1, 2, \ldots, n\}$ and every j in $\{1, 2, \ldots, p_i\}$, Φ_i^j involves at least all variables in X. Then, whatever the valid $\oplus$-component $\phi_{i_0}^{j_0}$ to make $B(X, Y)$ valid in S, 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, according to Definition 6, ρ is active domain preserving, and the proof is complete. □

The following corollary holds as an immediate consequence of Proposition 3 and of the fact that the cardinality of $\mathcal{AB}(\Delta)$ is finite.

Corollary 1 *Given $\Delta = (E, R)$, if every rule in R is safe, then $sem(\Delta)$ is finite.*

Corollary 1 shows that, if rules are safe then the v-set $sem(\Delta)$ provides a finite representation of the valuation v_Δ, in much the same way as the database extension E provides a finite representation of the valuation v_E.

Example 4 To illustrate Proposition 3 and Corollary 1 in the context of our running example, consider the rule $\rho_4 : Alert(x, z) \leftarrow \neg Fit(x) \otimes (Species(x, y) \wedge Fragile(y, z))$ in Example 1. It is easy to see that the Four-normal form of $body(\rho_4)$, hereafter denoted by Φ, is $(\neg Fit(x) \otimes Species(x, y)) \wedge (\neg Fit(x) \otimes Fragile(y, z))$.

It is clear that Φ has one $\vee$-component and one $\oplus$-component, namely itself. Thus, ρ_4 is safe, and by Proposition 3, ρ_4 is active domain preserving. As it is easy to see that all other rules in R are safe, Corollary 1 ensures that, whatever the database extension E, $sem(\Delta)$ is finite. □

5 Database Updating

In our approach to updating we follow standard approaches [2, 25] in which updating concerns only the database extension and does not modify the rule set. We note that, to the best of our knowledge, our work is the first to address the issue of database updating in the context of Four-value logic. Previous work has addressed a different but related topic, namely knowledge revision [3].

In the next section, we introduce the notion of *basic update*, that is how to change the database extension so as to assign a desired truth value to a given fact of $\mathcal{HB}$. Then in the following section, we introduce the concept of *update policy* as a means to specify and enforce update properties.

5.1 Basic Updates

Let us first recall that the database extension is a finite v-set (see Definition 1) and that a v-set is a means to store a valuation of $\mathcal{HB}$. We also recall that storing a valuation of $\mathcal{HB}$ in a database extension E is done under two assumptions,

Algorithm 1 Basic update

Input: $\Delta = (E, R)$, a fact φ and a truth value $\mathtt{v}$.
Output: The updated database extension E' such that $v_{E'}(\varphi) = \mathtt{v}$.
1: $current\_v := v_E(\varphi)$
2: **if** $\mathtt{v} = current\_v$ **then**
3: $\quad E' := E$
4: **else if** $current\_v = \mathtt{n}$ **then**
5: $\quad E' := E \cup \{\langle \varphi, \mathtt{v} \rangle\}$
6: **else if** $\mathtt{v} = \mathtt{n}$ **then**
7: $\quad E' := E \setminus \{\langle \varphi, current\_v \rangle\}$
8: **else**
9: $\quad E' := (E \setminus \{\langle \varphi, current\_v \rangle\}) \cup \{\langle \varphi, \mathtt{v} \rangle\}$
10: **return** E'

namely (1) E is 'functional', that is for all $\langle \varphi_1, \mathtt{v}_1 \rangle$ and $\langle \varphi_2, \mathtt{v}_2 \rangle$ in E, if $\varphi_1 = \varphi_2$ then $\mathtt{v}_1 = \mathtt{v}_2$, and (2) the Open World Assumption, meaning that a fact φ in $\mathcal{HB}$ is known (*i.e.,* $v_E(\varphi)$ is different than $\mathtt{n}$) if and only if $\langle \varphi, v_E(\varphi) \rangle$ is E. Both these assumptions must also hold in the updated extension and therefore, they work as constraints that must be satisfied by the updated database.

Definition 8 Let $\Delta = (E, R)$ be a database, φ a fact in $\mathcal{HB}$ and $\mathtt{v}$ a truth value in $\{\mathtt{t}, \mathtt{b}, \mathtt{n}, \mathtt{f}\}$. The *basic update of* Δ *by* φ *and* $\mathtt{v}$ is defined to be the database $\Delta' = (E', R)$ such that $v_{E'}(\varphi) = \mathtt{v}$, and $v_{E'}(\varphi') = v_E(\varphi')$, for every $\varphi' \neq \varphi$. □

It is easily seen that for all $\Delta = (E, R)$, φ and $\mathtt{v}$, the database Δ' always exists and it is unique. Algorithm 1 provides an effective way of computing E'. Indeed:

- If the test on line 2 succeeds then the update requires no change, which is achieved by letting E' to be E.
- Otherwise, we have $\mathtt{v} \neq current\_v$, and if the test on line 4 succeeds, E contains no pair involving φ. As in this case we have $\mathtt{v} \neq \mathtt{n}$, $\langle \varphi, \mathtt{v} \rangle$ is inserted in E so as to take the update into account while generating a proper v-set E'.
- If the test on line 4 fails but the test on line 6 succeeds, then we have $\mathtt{v} \neq current\_v$, $current\_v \neq \mathtt{n}$ and $\mathtt{v} = \mathtt{n}$. In this case, $\langle \varphi, current\_v \rangle$ is deleted from E so as to take the update into account.
- The last case is that on line 8, where $\mathtt{v} \neq current\_v$, $current\_v \neq \mathtt{n}$ and $\mathtt{v} \neq \mathtt{n}$. In this case, the pair $\langle \varphi, current\_v \rangle$ in E is changed to $\langle \varphi, \mathtt{v} \rangle$ so as to take the update into account while generating a proper v-set E'.

Example 5 In the context of our running example, due to the disagreement of the sensors measuring the humidity of bag 202, assume that new measurements have been conducted for this bag and that the sensors now output the following: $\langle H_1(202), \mathtt{t} \rangle$, $\langle H_2(202), \mathtt{t} \rangle$, $\langle W_1(202), \mathtt{t} \rangle$ and $\langle W_2(202), \mathtt{n} \rangle$.

Applying Algorithm 1 successively to each of these pairs, the database Δ of Example 1 is modified as follows:
- The pair $\langle H_1(202), \mathtt{t} \rangle$ is inserted.
- The pair $\langle H_2(202), \mathtt{t} \rangle$ is left unchanged.
- The pair $\langle W_1(202), \mathtt{f} \rangle$ is changed to $\langle W_1(202), \mathtt{t} \rangle$.
- The pair $\langle W_2(202), \mathtt{t} \rangle$ is deleted.

The resulting database extension E' is thus the following v-set:

$$E' = \{\ \langle W_1(101), \mathtt{t}\rangle, \langle W_1(202), \mathtt{t}\rangle, \langle W_1(303), \mathtt{f}\rangle,$$
$$\langle H_1(101), \mathtt{f}\rangle, \langle H_2(101), \mathtt{f}\rangle, \langle H_1(202), \mathtt{t}\rangle, \langle H_2(202), \mathtt{t}\rangle,$$
$$\langle Species(101, s_1), \mathtt{t}\rangle, \langle Species(202, s_1), \mathtt{t}\rangle, \langle Species(303, s_2), \mathtt{t}\rangle,$$
$$\langle Fragile(s_1, low), \mathtt{t}\rangle, \langle Fragile(s_2, high), \mathtt{t}\rangle \qquad \}$$ □

5.2 Update Policies

Although basic updates as defined earlier allow to perform any modification of the database extension, it is common practice to specify an *update policy* in order to enforce given properties on the update processing or on the resulting database.

The most typical situation is when constraints must be satisfied by the database, in which case it is required that updates preserve constraint satisfaction. In other words, assuming that the database satisfies the constraints before the update, it is required that the updated database also satisfies the constraints. This is for example very common when dealing with relational databases whose relations have to satisfy functional dependencies such as key or foreign key constraints [17]. In this case, one of two policies is adopted regarding acceptance of the update:

1. *Policy 1*: If the updated database does not satisfy the constraints then reject the update. Roughly speaking, this policy gives priority to 'old' knowledge over 'new' knowledge (*i.e.*, it privileges the knowledge already stored in the database).
2. *Policy 2*: Always accept the update and restore constraint satisfaction accordingly by performing auxiliary updates. Roughly speaking, this policy gives priority to 'new' knowledge (as expressed by the update) over 'old' knowledge.

In our case, as mentioned earlier, the database must satisfy two constraints, namely the functionality of the database extension and the Open World Assumption. Both these constraints must hold in the updated database and their satisfaction can be enforced by applying one of the two policies above. Below, we describe the update algorithm for each policy, recalling that each algorithm takes as input a database $\Delta = (E, R)$, a fact φ and a truth value $\mathtt{v}$, and aims at returning a database $\Delta = (E', R)$ such that $v_{E'}(\varphi) = \mathtt{v}$.

1. *Algorithm for Policy 1* (give priority to old knowledge over new knowledge):
   if $v_E(\varphi) = \mathtt{n}$ and $\mathtt{v} \neq \mathtt{n}$ **then** $E' := E \cup \{\langle \varphi, \mathtt{v}\rangle\}$
   else if $v_E(\varphi) \neq \mathtt{n}$ and $\mathtt{v} = \mathtt{n}$ **then** $E' := E \setminus \{\langle \varphi, v_E(\varphi)\rangle\}$
   else $E' := E$
   return E'
2. *Algorithm for Policy 2* (give priority to new knowledge over old knowledge):
   if $v_E(\varphi) = \mathtt{v}$ **then** $E' := E$
   else if $v_E(\varphi) = \mathtt{n}$ and $\mathtt{v} \neq \mathtt{n}$ **then** $E' := E \cup \{\langle \varphi, \mathtt{v}\rangle\}$
   else if $v_E(\varphi) \neq \mathtt{n}$ and $\mathtt{v} = \mathtt{n}$ **then** $E' := E \setminus \{\langle \varphi, v_E(\varphi)\rangle\}$
   else $E' := (E \setminus \{\langle \varphi, v_E(\varphi)\rangle\}) \cup \{\langle \varphi, \mathtt{v}\rangle\}$
   return E'

It should be noticed that each of the above two algorithms (Algorithm for Policy 1 and Algorithm for Policy 2) processes two truth values, namely $\mathtt{v}$ and $v_E(\varphi)$, and computes a third truth value, namely $v_{E'}(\varphi)$ based on which it computes E'.

Therefore each of these algorithms computes the values of a binary operator over $\{\mathtt{t}, \mathtt{b}, \mathtt{n}, \mathtt{f}\}$. We can view this operator as 'defining' the corresponding policy and the truth table of the operator as 'implementing' the policy.

For example, let π_{old} be the operator defining Policy 1 and let its truth table be as shown in Figure 4. Then for a pair $(\mathtt{v}, v_E(\varphi))$ such that $\mathtt{v} = \mathtt{t}$ and $v_E(\varphi) = \mathtt{f}$, the value of $\pi_{old}(\mathtt{t}, \mathtt{f})$ (*i.e.,* the value found at row $\mathtt{t}$ and column $\mathtt{f}$ of the truth table of π_{old}) is $\mathtt{f}$. As a consequence, according to Policy 1, if the update aims at assigning truth value $\mathtt{t}$ to φ and if the current truth value of φ is $\mathtt{f}$ (*i.e.,* $v_E(\varphi) = \mathtt{f}$), then in the updated database, the truth value of φ is not changed (as stated by the last 'else' statement of Algorithm for Policy 1 above).

π_{old}	t	b	n	f
t	t	b	t	f
b	t	b	b	f
n	n	n	n	n
f	t	b	f	f

π_{new}	t	b	n	f
t	t	t	t	t
b	b	b	b	b
n	n	n	n	n
f	f	f	f	f

Fig. 4 The truth tables of the operators π_{old} and π_{new}

Similarly, let π_{new} be the operator defining Policy 2 and let its truth table be as shown in Figure 4. Then for the same pair as above, that is the pair $(\mathtt{v}, v_E(\varphi))$ such that $\mathtt{v} = \mathtt{t}$ and $v_E(\varphi) = \mathtt{f}$, the value of $\pi_{new}(\mathtt{t}, \mathtt{f})$ is $\mathtt{t}$. Hence, according to Policy 2, if the update aims at assigning the truth value $\mathtt{t}$ to φ and if the current truth value of φ is $\mathtt{f}$ (*i.e.,* $v_E(\varphi) = \mathtt{f}$), then in the updated database, the truth value of φ is changed from $\mathtt{f}$ to $\mathtt{t}$ (as stated by the last 'else' statement of Algorithm for Policy 2 above).

Now, an important remark is in order here regarding binary connectors implementing update policies. Indeed, since the Four-valued logic has been shown to be functionally complete [4], it turns out that any binary operator over $\{\mathtt{t}, \mathtt{b}, \mathtt{n}, \mathtt{f}\}$ can be defined by an expression whose operations are the basic connectors of Four-valued logic. Thus, any update policy defined through a binary connector can also be defined using a well-formed formula of the Four-valued logic.

For example, the expression $\Phi_{new}(\phi_1, \phi_2)$ defining the operator π_{new} is simply $\Phi_{new}(\phi_1, \phi_2) = \phi_1$. This expression shows that, according to π_{new}, all updates are processed as specified by the user. As another example, the expression $\Phi_{reject}(\phi_1, \phi_2) = \phi_2$ defines the (useless) update policy whereby all updates are rejected. On the other hand, the expression $\Phi_{old}(\phi_1, \phi_2)$ defining π_{old} is more involved, since it follows from [26] that

$$\Phi_{old}(\phi_1, \phi_2) = ((\Phi_{\mathtt{t}} \vee \neg\Phi_{\mathtt{f}}) \otimes \sim\not\sim \Phi_{\mathtt{n}}) \oplus \not\sim \Phi_{\mathtt{b}}$$

where[1]

$\Phi_{\mathtt{t}} = (\mathbf{T}\phi_1 \wedge \mathbf{T}\phi_2) \vee (\mathbf{T}\phi_1 \wedge \mathbf{N}\phi_2) \vee (\mathbf{B}\phi_1 \wedge \mathbf{T}\phi_2)$
$\Phi_{\mathtt{f}} = (\mathbf{T}\phi_1 \wedge \mathbf{F}\phi_2) \vee (\mathbf{B}\phi_1 \wedge \mathbf{F}\phi_2) \vee (\mathbf{F}\phi_1 \wedge \mathbf{N}\phi_2) \vee (\mathbf{F}\phi_1 \wedge \mathbf{F}\phi_2)$
$\Phi_{\mathtt{n}} = (\mathbf{N}\phi_1 \wedge \mathbf{T}\phi_2) \vee (\mathbf{N}\phi_1 \wedge \mathbf{B}\phi_2) \vee (\mathbf{N}\phi_1 \wedge \mathbf{N}\phi_2) \vee (\mathbf{N}\phi_1 \wedge \mathbf{F}\phi_2)$
$\Phi_{\mathtt{b}} = (\mathbf{T}\phi_1 \wedge \mathbf{B}\phi_2) \vee (\mathbf{B}\phi_1 \wedge \mathbf{B}\phi_2) \vee (\mathbf{B}\phi_1 \wedge \mathbf{N}\phi_2) \vee (\mathbf{F}\phi_1 \wedge \mathbf{B}\phi_2)$.

[1] The expressions involving the basic connectors for the unary connectors **T**, **B**, **N** and **F** used here, can be found in Section 2

The technical details on how to obtain the above formula are omitted since they lie outside the scope of the present paper. Whatever the way the update policy is defined, whether through a binary operator or through an expression, the notion of *update policy* is formally defined as follows.

Definition 9 Let $\Delta = (E, R)$ be a database and let π be a binary operator over $\{\mathtt{t}, \mathtt{b}, \mathtt{n}, \mathtt{f}\}$ called an *update policy*. Then given a fact φ of $\mathcal{HB}$ and truth value $\mathtt{v}$, the *update of* Δ *by* φ *and* $\mathtt{v}$*, using policy* π, is defined to be the database $\Delta' = (E', R)$ such that $v_{E'}(\varphi) = \pi(\mathtt{v}, v_E(\varphi))$ and $v_{E'}(\varphi') = v_E(\varphi')$, for every $\varphi' \neq \varphi$. □

Based on Definition 8 and Definition 9, it is easy to see that, given Δ, φ and $\mathtt{v}$, the basic update of Δ by φ and $\mathtt{v}$ yields the same result as the update of Δ by φ and $\mathtt{v}$, *using policy* π_{new}. As a consequence, basic updates can be seen as particular updates using a policy, namely those updates using policy π_{new}.

We end this section by giving further examples of update policies that might be relevant in practice.

π_1	t	b	n	f
t	t	b	t	f
b	t	b	n	f
n	t	b	n	f
f	t	b	f	f

π_2	t	b	n	f
t	t	t	t	t
b	t	b	n	b
n	t	n	n	n
f	t	b	n	f

Fig. 5 The truth tables of the connectors defining update policies π_1 and π_2

The first update policy we would like to introduce is that of *data integration*. According to this policy, updates are performed so as the truth value $\mathtt{v}$ specified in the update be combined with the existing one, $v_E(\varphi)$, through the connector $\oplus$, whose truth table is shown in Figure 1. This update policy is used when integrating data from various sources.

The data integration policy can be 'smoothed' by preventing new inconsistencies from being introduced in a source from data coming from other sources (through updates). More precisely, given φ and $\mathtt{v}$, the update involving φ and $\mathtt{v}$ is performed as follows:
– If $v_E(\varphi) \oplus \mathtt{v} \neq \mathtt{b}$ then set the truth value of φ to $v_E(\varphi) \oplus \mathtt{v}$;
– Otherwise reject the update.
The truth table of the associated connector π_1 is shown in Figure 5. It should be noticed from this truth table that, for instance, when $\mathtt{v} = \mathtt{t}$ and $v_E(\varphi) = \mathtt{f}$, $v_E(\varphi) \oplus \mathtt{v} = \mathtt{b}$, thus implying that $\pi_1(\mathtt{t}, \mathtt{f}) = \mathtt{f}$. On the other hand, for $\mathtt{v} = \mathtt{f}$ and $v_E(\varphi) = \mathtt{n}$, $v_E(\varphi) \oplus \mathtt{v} = \mathtt{f}$, and $\pi_1(\mathtt{f}, \mathtt{n}) = \mathtt{f}$.

As a last example of update policy consider what we could call *truth increasing policy*. This policy states that an update involving φ and $\mathtt{v}$ is processed only if it results in an increase with respect to the truth ordering of the truth value of φ. In other words:
– If $v_E(\varphi) \preceq_t \mathtt{v}$ then set the truth value of φ to $\mathtt{v}$;
– Otherwise reject the update.
The truth table of the associated operator π_2 is shown in Figure 5. We notice from this truth table that, for instance, $\pi_2(\mathtt{t}, \mathtt{b}) = \mathtt{t}$ because $\mathtt{b} \preceq_t \mathtt{t}$ holds, and that $\pi_2(\mathtt{f}, \mathtt{n}) = \mathtt{n}$ because $\mathtt{n} \preceq_t \mathtt{f}$ does not hold.

It should be clear by now that many other update policies could be defined. However, considering every possible binary truth table as an update policy is unrealistic. In any case, the examples presented in this section show that our approach allows to specify various update policies in a way which is simple, easy to implement and powerful as it allows to specify complex update policies.

We recall in this respect that the notion of update policy was first introduced in [5] in the context of updating views in relational databases. This work was thus the earliest one that provided a theoretical insight in the difficult problem of specifying and implementing sophisticated database updates.

6 Related Work

Comparing our approach with all related work in the literature is simply impossible due to the huge number of papers published during the past four or five decades on the topics addressed here. In what follows we focus mainly on related work in three areas: logic and databases, inconsistent databases, and multi-valued logic.

6.1 Logic and Databases

We refer to [10,17,28] for surveys of standard approaches to Datalog databases, and to [9] for a more detailed survey of the problem of negation. We note that all these works use the Closed World Assumption (CWA [30]), which leads to difficulties when dealing with false facts - a problem that does not arise in our framework, as we use the Open World Assumption (OWA).

Working under OWA is not new [8] and the need to do so emerged in problems of data integration on the web. Indeed, when a piece of information is not retrieved in the answer to a query, one cannot assume that this piece of information is *false*, but rather that it has not been searched properly. Therefore it is more appropriate to consider that this piece of information is *unknown*.

On the other hand, as the examples in this paper suggest, when integrating information from several sources, contradictions may occur, and this motivates the introduction of *inconsistent* as a truth value. This point of view has also been considered in [11] but in a logical framework that differs from ours. Indeed, in [11], the underlying four valued logic is not the one in [7], although the considered implication looks similar to implication denoted here by $\rightarrow_1$. Moreover, in [11], the authors consider two negations in the context of CWA and propose an alternating strategy for computing the database semantics, inspired from well-founded semantics [18].

The approach in [3] also considers two negations as done in [11]. As already discussed in much detail, this approach is closely related to ours when not using negation as failure - although the semantics is slightly different. However, our approach is more expressive than [3] and [11], since we allow rules not allowed in [3] and [11]. Investigating how to express negation as failure in our approach is a topic of future work.

As seen earlier, the work in [14,15] is closely related to our approach because the underlying logic in [14,15] is that of [7], and also because the syntax of the rules in [14] is similar to ours - although not strictly comparable to ours. We

recall again that the semantics of k-existential programs defined in [14] differs from ours, because the rules in [14] are seen as equivalences, which is not the case in our approach.

In [19, 20], related work following similar semantics can be found in the context of relational databases. In that work, reasoning with four truth values is modeled in two distinct ways: one for deducing true information and one for deducing false information, while inconsistency is considered as information obtained by the 'intersection' of these two ways of reasoning.

The problem of updating deductive databases was first addressed in [31]. Our work in [2] was among the first to suggest storing false facts and giving priority to most recent updates. The present work builds upon these basic ideas in a much wider context, while the introduction of update policy stems from the work in [5]. Moreover, and to the best of our knowledge, this work is the first one to address the issue of updates in the framework of Four-valued logic.

6.2 Inconsistent Databases

In this paper, in contrast to previous work on inconsistent databases, we propose a radically different approach. Indeed, the goal of previous work dealing with inconsistencies, is either to define and study 'repairs' so as to make the database consistent ([1, 21]), and/or to identify queries whose answers are independent from any contradiction ([22]). Instead, we propose an approach in which inconsistent information can be stored or deduced through rules, and our goal is not to eliminate or avoid contradictions.

Our semantics allows for handling inconsistent information as such, thus reflecting what happens in real world applications in which true, false, inconsistent and unknown information have to be dealt with. This is particularly true in data integration environments. In our work, we follow the position in [16], 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. We are currently studying this topic.

The approach in [27] addresses the issue of data inconsistency due to data integration according to a specific scenario: a central server collects facts from autonomous sources and then tries to combine them using the syntax and semantics of [14, 15], on the one hand, and a set of *hypotheses* H, representing the server's own estimates, on the other hand. 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 is then shown to provide hypothesis-based semantics for the class of programs defined in [15]; and in the case of Datalog^{neg} programs, these semantics have been shown to extend well-founded semantics of [18] and Kripke Kleen semantics of [12].

6.3 Multi-valued Logic

The Four-valued logic that we consider in this work has been introduced in [7] and has since motivated several works in non standard logic. Again, our aim here is not to review all these works, and we refer to [29] for a survey of this topic. Here, we focus on works most closely related to ours. In [4] the issue of the functional completeness has been addressed among others and their results have inspired our concern on this issue. On the other hand, the bi-lattice structure of this logic has been widely studied in [14,15], where the concept of logic programs in this framework was first introduced. Rule semantics in Four-valued logic has been investigated in [14,15] and in [3]. As seen in sub-Section 3.3, these two approaches are related to ours although not always comparable. We notice here that our approach offers a generic framework for expressing rules in Four-valued context: as shown in Example 3, it happens that rules in our approach might not be expressible in the approach of [14,15] or in the approach of [3]. Moreover the notion of database model, as considered in this work, has been investigated in [3], but not in [14,15]. As a final remark, we note that the work in [32] established a strong relationship between Four-valued logic and rough set theory.

7 Concluding remarks

In this paper we have introduced a novel approach to deductive databases dealing with contradictory information. We stress again that our 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 these issues we proposed a deductive database approach based on the Four-valued logic initially introduced in [7]. We have studied the strong relationship between our approach and those in [14,15] and [3]. One of our main contributions is to have shown that database semantics can be seen as the unique k-minimal and k-increasing database model, and that this holds for three implications proposed so far in the literature. Two other important contributions of our work are (*i*) to characterize safe rules, ensuring that the database semantics is finite, and (*ii*) to propose a new kind of update allowing to 'combine' the new truth value of a fact with its current truth value in the database. As shown in Section 5, this new way of updating is of particular interest when it comes to enforcing updates satisfying properties of various kinds.

Based on these results, we are currently investigating the following issues. First, as rules can contradict each other (something that happens frequently in real life), it is important to characterize the exact situations when these contradictions occur, so as to take appropriate actions, as suggested in [16]. We approach this important issue by extending the form of the rules to allow in their body additional connectors (such as connector **B** introduced in [32] and recalled in Section 2). Another extension of our work is to investigate negation as failure in our context, inspired by the work in [3,11]. Last but not least, we conjecture that the Four-valued framework provides the right context for defining new measures related to *data quality*, a research topic that we also plan to investigate in the future.

Declarations

Author contributions: The two authors contributed to the study, conception and design. Both read and approved the submitted manuscript.

Funding: No funds, grants, or other support was received for conducting this study.

Financial interests: The authors declare they have no financial interests.

Non-financial interests: The authors declare they have no non-financial interests.

Data availability: Data sharing is not applicable to this article as no data sets were generated or analyzed during the current study.

References

1. Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *Database Theory - ICDT, 12th International Conference, Russia, March 23-25, 2009, Proceedings*, pages 31–41, 2009.
2. Mirian Halfeld Ferrari Alves, Dominique Laurent, and Nicolas Spyratos. Update rules in datalog programs. *J. Log. Comput.*, 8(6):745–775, 1998.
3. Ofer Arieli. Paraconsistent declarative semantics for extended logic programs. *Ann. Math. Artif. Intell.*, 36(4):381–417, 2002.
4. Ofer Arieli and Arnon Avron. The value of the four values. *Artif. Intell.*, 102(1):97–141, 1998.
5. François Bancilhon and Nicolas Spyratos. Update semantics of relational views. *ACM Trans. Database Syst.*, 6(4):557–575, 1981.
6. Yan L. Batay. Maintaining grain quality during storage and transport. In *Cereal Grains, Assessing and Managing Quality, Second Edition*, pages 571–590. Woodhead Publishing Series in Food Science, Technology and Nutrition, 2017.
7. Nuel D. Belnap. A useful four-valued logic. In J. Michael Dunn and George Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 5–37, Dordrecht, 1977. Springer Netherlands.
8. Mike Bergman. 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.
9. Nicole Bidoit. Negation in rule-based database languages: A survey. *Theor. Comput. Sci.*, 78(1):3–83, 1991.
10. Stefano Ceri, Georg Gottlob, and Letizia Tanca. *Logic Programming and Databases*. Surveys in computer science. Springer, 1990.
11. Sandra de Amo and Mônica Sakuray Pais. A paraconsistent logic programming approach for querying inconsistent databases. *Int. J. Approx. Reason.*, 46(2):366–386, 2007.
12. Melvin Fitting. A kripke-kleene semantics for logic programs. *J. Log. Program.*, 2(4):295–312, 1985.
13. Melvin Fitting. Negation as refutation. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, pages 63–70. IEEE Computer Society, 1989.
14. Melvin Fitting. Bilattices in logic prograrnming. In *Proceedings of the 20th International Symposium on Multiple-Valued Logic, ISMVL 1990, Charlotte, NC, USA, May 23-25, 1990*, pages 238–246. IEEE Computer Society, 1990.
15. Melvin Fitting. Bilattices and the semantics of logic programming. *J. Log. Program.*, 11(1&2):91–116, 1991.
16. Dov Gabbay and Anthony Hunter. Making inconsistency respectable: A logical framework for inconsistency in reasoning, part i — a position paper. In Philippe Jorrand and Jozef Kelemen, editors, *Fundamentals of Artificial Intelligence Research*, pages 19–32, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.
17. Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database systems - the complete book (2. ed.)*. Pearson Education, 2009.

18. Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.
19. Gösta Grahne and Ali Moallemi. A useful four-valued database logic. In Bipin C. Desai, Sergio Flesca, Ester Zumpano, Elio Masciari, and Luciano Caroprese, editors, *Proceedings of the 22nd International Database Engineering & Applications Symposium, IDEAS 2018, Villa San Giovanni, Italy, June 18-20, 2018*, pages 22–30. ACM, 2018.
20. Gösta Grahne and Ali Moallemi. Universal (and existential) nulls. *Fundam. Inform.*, 167(4):287–321, 2019.
21. Gianluigi Greco, Sergio Greco, and Ester Zumpano. A logical framework for querying and repairing inconsistent databases. *IEEE Trans. Knowl. Data Eng.*, 15(6):1389–1408, 2003.
22. Sergio Greco, Cristian Molinaro, and Irina Trubitsyna. Computing approximate query answers over inconsistent knowledge bases. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1838–1846. ijcai.org, 2018.
23. Allen P. Hazen and Francis Jeffry Pelletier. K3, l3, lp, rm3, a3, FDE: how to make many-valued logics work for you. *CoRR*, abs/1711.05816, 2017.
24. Dominique Laurent. 4-valued semantics under the OWA: A deductive database approach. In Giorgos Flouris, Dominique Laurent, Dimitris Plexousakis, Nicolas Spyratos, and Yuzuru Tanaka, editors, *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*, pages 101–116. Springer, 2019.
25. Dominique Laurent, Viet Phan Luong, and Nicolas Spyratos. The use of deleted tuples in database, querying and updating. *Acta Inf.*, 34(12):905–925, 1997.
26. Dominique Laurent and Nicolas Spyratos. Four-valued semantics for deductive databases. *CoRR*, abs/2108.02587, 2021.
27. Yann Loyer, Nicolas Spyratos, and Daniel Stamate. Hypothesis-based semantics of logic programs in multivalued logics. *ACM Trans. Comput. Log.*, 5(3):508–527, 2004.
28. Jack Minker, Dietmar Seipel, and Carlo Zaniolo. Logic and databases: A history of deductive databases. In Jörg H. Siekmann, editor, *Computational Logic*, volume 9 of *Handbook of the History of Logic*, pages 571–627. Elsevier, 2014.
29. Hitoshi Omori and Heinrich Wansing. 40 years of FDE: an introductory overview. *Studia Logica*, 105(6):1021–1049, 2017.
30. Raymond Reiter. On closed world data bases. In Hervé Gallaire and Jack Minker, editors, *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, pages 55–76, New York, 1977. Plemum Press.
31. Raymond Reiter. On formalizing database updates: Preliminary report. In Alain Pirotte, Claude Delobel, and Georg Gottlob, editors, *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*, pages 10–20. Springer, 1992.
32. A. Tsoukiàs. A first-order, four valued, weakly paraconsistent logic and its relation to rough sets semantics. *Foundations of Computing and Decision Sciences*, 12:85–108, 2002.