

An application of deontic logic to information system constraints

Jose Carmo

Department of Mathematics

University of Madeira

Funchal, Portugal

jcc@math.uma.pt

Robert Demolombe

ONERA

Department of Information Processing and Modeling

Toulouse, France

Robert.Demolombe@cert.fr

Andrew J.I. Jones

Department of Philosophy and

Norwegian Research Center for Computers and Law

Oslo, Norway

a.j.i.jones@filosofi.uio.no

Abstract. In the field of information systems the term “constraint” is applied to statements of various kinds. Here we start from the analysis of a simple example to characterise the different kinds of constraints. It is shown that constraints may be necessary truths or deontic constraints. Moreover, deontic constraints are classified into three different types: deontic constraints about the world, deontic constraints about the representation of the world (self-completeness), and deontic constraints about the links between the world and its representation (validity and completeness).

We describe a modal logical framework to define the different types of constraints, to characterise their violations, and to show how to repair their violations. Two different general

forms of deontic constraints are considered, namely $O(\phi \rightarrow \psi)$ and $\phi \rightarrow O\psi$, and it is shown that, except for deontic constraints about the world, the latter is more appropriate. Special issues related to the definition of quantifiers in the context of modal operators are also considered.

Keywords: Integrity, Information Systems, Modal Logic.

1. Introduction

There are many papers in the literature about integrity constraints in the context of databases or information systems (IS) [15, 16, 18, 19, 21], but there is no consensus about the meaning of the terms “integrity” and “constraints”, and their definitions are not completely clear.

Some authors claim that integrity constraints contribute to the definition of the semantics of data. For instance, Kwast in [12] says that “integrity constraints are formal tools to give meaning to data in the database”, Meyer et al. in [13] say that some of the integrity constraints are “analytical statements, which describe logical consequences that follow from the words used in the statements”. However, Gertz in [9] claims that “from a practical point of view, the traditional notion of data integrity, i.e. the semantic correctness of stored data, plays only a minor role with regard to the reliability of [...] data”, where he informally defines “reliability” in terms of “accuracy and completeness of data”.

In his very popular text book [3] Date defines integrity only in terms of “correctness”. He considers that in a database there are relations, called “base relations”, that play a specific role, and he says: “...it is usual to think of integrity rules as applying to base relations specifically. This is because it is the base relations that are supposed to “reflect reality”, and hence it is the base relations that must be constrained to contain correct, or at least plausible, values....”. He also says that “the term “integrity” refers to accuracy or correctness of the data in the database. To say that the database is in a state of integrity means, therefore, that the database is correct - meaning, precisely, that it does not violate any known integrity rule”. Even if Gertz complains that “there neither exists a formalism nor a technique” to guarantee data reliability, an important pioneer work in this direction is that of Motro in [14]. His paper title: “Integrity = Validity + Completeness”, clearly shows his personal view of what integrity is. More explicitly Motro says “answers [to queries] have integrity if they contain the whole truth (completeness) and nothing but the truth (validity)”, and he gives a formal definition of valid answers and of complete answers to queries expressed in classical first order logic for a restricted class of queries called “conjunctive queries”.

There is also a lack of consensus for the definition of the notion of constraint satisfaction. In [7] Gardarin and Valduriez say that “a database system must guarantee consistency, a condition that exists if the database satisfies a set of rules called semantic integrity constraints or simply integrity constraints”. As Minker et al. say in [10] “there is a broad body of work on logic and relational databases, and a general consensus of what databases (facts and rules) and queries mean. However, there is less work on the meaning of integrity constraints and certainly no

consensus [...] . For instance, one may define that ICs must be consistent with the database, or define they must be provable statements, deducible from the database”. In [20], van der Meyden refers to “unacceptable” database states, and gives the following example: “for a given application not all relations for a schema are sensible. For example, each social security number should identify a unique person, so a relation R with attributes Person and SSN containing tuples (Jones,111-222-333) and (Smith,111-222-333) is unacceptable”. Later on he says that “satisfaction of the dependencies ¹ provides a way of checking that the data are not incoherent”. He mentions that the two definitions of constraint satisfaction mentioned by Minker et al. have been proposed by Kowalski [11] (definition based on consistency) and by Reiter [17] (definition based on entailment), and we fully agree with van der Meyden when he adds that “the distinction between the two definitions blurs somewhat once one considers query processing in the presence of integrity constraints. In relational databases, integrity constraints satisfied by a database may be ignored during query processing [...]. However, ignoring integrity constraints that are satisfied by an indefinite database under the consistency definition may lead to undesirable results”.

This brief overview shows that integrity constraints may have several different implicit connotations. The objective of this paper is to make explicit the notions involved in integrity constraints and to give formal logical definitions for each of them. We shall see that in many cases constraints are obligations, and that is why deontic logic plays a specific role in the formal definitions.

In the next section a simple example is used to analyse informally the most usual definitions of integrity constraints found in the literature. This analysis shows that some constraints are about links between the database and the real world. These links (called “validity” and “completeness”) are formally defined in section 3, and we describe some key properties for reasoning about these notions. In section 4, a classification is presented of the different types of integrity constraints, and for each one we give its formal definition and the definition of violation, and we show how violations could be repaired. In this section, the types of derivations we would like to be able to draw in an appropriate logic are also made explicit. The next section, section 5, presents a proposal for such a logic, and an analysis of two possible definitions of deontic constraints. In the conclusion we raise the possibility of applying this framework to the more general context of multi-agent systems.

2. Analysis of the standard approach

Although, as we have seen in the introduction, there is no consensus about the notion of integrity constraints, and about constraint satisfaction, we shall nevertheless in this section refer to a “standard approach” to integrity constraints. Our objective is to analyse, with the help of a typical example, the most significant features of integrity constraints mentioned by the database community, and then to make explicit the fundamental concepts that are involved in their definitions.

¹Dependencies are a special kind of constraints that have been extensively investigated in the database field.

It is assumed, in this section, that the database content is represented by a set of closed formulas DB of some first order predicate calculus language. In the special case of relational databases DB is restricted to a set of ground atoms. Integrity constraints are represented by a set IC of closed formulas of the same language.

Now let's consider the following example where we have the three integrity constraints:

IC1': every car uses diesel oil or petrol.

IC2': there is no car that uses both diesel oil and petrol.

IC3': every ordered car is in the stock.

To represent these constraints by formulas we consider a language that contains the predicate symbols: $c(x)$, $d(x)$, $p(x)$, $o(x)$ and $s(x)$, whose intuitive meanings respectively are: x is a car, x uses diesel oil, x uses petrol, x has been ordered, and x is in stock. Then the integrity constraints are represented by the formulas:

IC1: $\forall x(c(x) \rightarrow d(x) \vee p(x))$

IC2: $\neg\exists x(d(x) \wedge p(x))$

IC3: $\forall x(o(x) \rightarrow s(x))$

and we have $IC = \{IC1, IC2, IC3\}$. It is also assumed that the database content is represented by the set of formulas $DB = \{c(a), d(b), p(b), o(c)\}$.

Most members of the database community believe that the database represents a unique state of the world. The formal consequence of this assumption is that DB must be a complete theory. Indeed, each formula p or its negation $\neg p$ must be derivable from DB in the same way as each formula p is true or false of the unique state of the world represented by the database. In [17] Reiter has defined a set of axioms, called CWA , whose addition to DB leads to a complete theory, i.e. the theory $DB \cup CWA$ is complete. In our example CWA consists of the set of Unique Name Axioms: $\neg(a = b)$, $\neg(a = c)$ and $\neg(b = c)$, of the Domain Closure Axiom: $\forall x(x = a \vee x = b \vee x = c)$, and of a set of Completion Axioms, one for each predicate symbol. The Completion Axioms state that no other objects satisfy a given predicate than the objects that explicitly satisfy this predicate in DB . For example, for the predicate symbols $o(x)$ and $s(x)$ the Completion Axioms respectively are: $\forall x(o(x) \rightarrow x = c)$ and $\forall x(s(x) \rightarrow false)$. Notice that $DB \not\models p$ implies $DB \cup CWA \vdash \neg p$.

In the standard approach the constraints are satisfied if the database is consistent with the constraints, i.e. $\{IC \cup DB\}$ is consistent, or if the database entails the constraints, i.e. $DB \vdash IC$. The second definition of satisfaction is difficult to justify in the case where DB is an incomplete theory. For instance, in the particular case where the database is empty, i.e. $DB = \emptyset$, it is clear that we do not have $DB \vdash IC$, while there is no intuitive reason to say that constraints are not satisfied. However, if the Closed World Assumption is accepted, the two definitions coincide, since $\{IC \cup DB \cup CWA\}$ is consistent iff $DB \cup CWA \vdash IC$ holds.

To explain which undesirable situations are implicitly characterised by the notion of inconsistency, we have to distinguish what is represented in the database and what is true of the world. For this purpose, a doxastic modality B_{db} is introduced here, whose formal definition will be given in section 3. Formulas of the form $B_{db}p$ and p , where p belongs to a classical first order

language, are respectively read: the database believes p , and p is true of the world. Then, using this modality, the database content is represented by $DB' = \{B_{db}c(a), B_{db}d(b), B_{db}p(b), B_{db}o(c)\}$ instead of DB .

Now we can use this example to show how constraint violations are characterised in the standard approach, and to make explicit the reasons for not accepting the corresponding situations.

Let's take first the constraint $IC2$. The set $\{IC2 \cup DB\}$ is clearly inconsistent, due to the information about the car b . Then this situation is unacceptable. In the standard approach it is said that it is unacceptable because there is an inconsistency, and obviously an inconsistent set of formulas cannot be a representation of the world. Furthermore, it is said that to come back to an acceptable situation either $d(b)$ or $p(b)$ has to be removed from the database. Our explanation of this attitude is that $IC2$ is assumed to be true of the world, and that is why the possibility of removing $IC2$ to restore consistency is discarded. Since the set of formulas $\{IC2, d(b), p(b)\}$ is inconsistent, at least one of these three formulas is false of the world, and since $IC2$ is assumed to be true, either $d(b)$ or $p(b)$ is false of the world. So our explanation is that the situation is unacceptable because the **database contains false data**. If we use the modal language, the interpretation of the unacceptability of the situation is that we have either $B_{db}d(b) \wedge \neg d(b)$ or $B_{db}p(b) \wedge \neg p(b)$.

If we consider the constraint $IC1$, $\{IC1 \cup DB\}$ is consistent, but $\{IC1 \cup DB \cup CWA\}$ is inconsistent. Indeed, we have $DB \vdash c(a)$, $DB \not\vdash d(a)$ and $DB \not\vdash p(a)$, and therefore we have: $DB \cup CWA \vdash c(a) \wedge \neg d(a) \wedge \neg p(a)$. Then, this database state is unacceptable. Since, in the standard approach, it is assumed that $IC1$ is true of the world, to come back to a situation where $\{IC1 \cup DB \cup CWA\}$ is consistent there are three possible cases to consider. If $c(a)$ is true of the world, since $IC1$ is also true we can infer that $d(a) \vee p(a)$ is true of the world, and the reason why the situation is unacceptable is that **some true fact**, either $d(a)$ or $p(a)$ **is missing in the database**. Using the modal language we have either $d(a) \wedge \neg B_{db}d(a)$ or $p(a) \wedge \neg B_{db}p(a)$. If $c(a)$ is false of the world, the situation is unacceptable because the database contains false data, i.e. we have $B_{db}c(a) \wedge \neg c(a)$.

Finally, if we consider the constraint $IC3$, the set $\{IC3 \cup DB \cup CWA\}$ is inconsistent, and the situation is unacceptable. Here, it is not so obvious that $IC3$ is assumed to be true of the world. This assumption might be accepted if, for instance, the company that sells the car is organised in such a way that a car cannot be ordered if it is not already in stock. In this case $IC3$ can be assumed to be true of the world, and the unacceptability of the situation is similar to that which arose for constraints $IC1$ and $IC2$. That is, either $o(c)$ is true of the world and therefore $s(c)$ is also true of the world, and the true fact $s(c)$ **is missing in the database**, or $o(c)$ is false of the world and the **database contains false data**.

However, though this possibility is usually not considered in the standard approach, it may be that the truth of $IC3$ is not guaranteed by the organisation of the company, and that $IC3$ is a norm that should hold in the world, but does not necessarily hold. Carmo and Jones in [1] have called these constraints “deontic constraints”, or “soft constraints” (Meyer et al. in [13] also use the term “soft constraint”). In that case the database is used to detect situations where

$IC3$ is false of the world, that is, situations where the obligation about the world is violated. Here, there is no need to change the database content, nor to remove the constraint $IC3$. In this context $IC3$ would be wrongly represented by a formula in classical logic that is inconsistent with DB . On the contrary, instead of an inconsistency, we should be able to represent an obligation violation using a deontic logic, and, in case of violation there is no need to change the database content. The system should inform users that a violation has occurred, and they should perform appropriate actions in the world in order to try to repair the violation.

There is another kind of constraint that is not considered in the standard approach, and which has been introduced by Reiter in [18]. These constraints are about a kind of completeness that is not defined by reference to the world, but by reference to the database content itself. An example of a constraint of this kind would be that, if the database believes that some x is a car, then it should believe that x uses diesel oil or it should believe that x uses petrol.

The conclusion we draw from this analysis is that in the standard approach consistency checking is used as a formal technique to detect violations of implicit obligations. These implicit obligations either express that if some facts are represented in the database they should be true of the world, or that if some facts are true of the world they should be represented in the database. In the rest of this paper, these two kinds of obligations are respectively called obligations about ‘validity’ and obligations about ‘completeness’.

Another conclusion is that there are constraints that are known to be true of the world and that are not falsifiable, and there are constraints that are obligations about the world and can be violated. These violations can be detected by an information system provided the relevant facts for these detection are either valid or complete or both.

In the following we adopt the view, as does Motro, that integrity is defined in terms of validity and completeness, and we present logical definitions of these notions and of their properties.

3. Information validity and information completeness

First we must draw attention to a terminological problem. We use the terms ‘validity’ and ‘completeness’, as in Motro [14], with a meaning different from that assigned in formal logic, where ‘validity’ may be used to refer to the property of soundness of a given axiomatisation (if an axiomatisation is sound, all theorems are valid formulas), and the ‘completeness’ of an axiomatisation means that it generates all valid formulas as theorems.

We use the term ‘database’ to denote the information about the world managed by the information system. This information can be formally represented by a finite set of sentences (i.e. formulas without free variables) DB of a given first order predicate calculus language L_1 . This language is extended with a doxastic modality B_{db} . If p is in L_1 , $B_{db}p$ can be read ‘the database believes p ’. Let us call this extended language L . L is formally defined as follows:

- if p belongs to L_1 , then p and $B_{db}p$ belong to L ,
- if p and q belong to L , then $\neg p$, $p \vee q$, and $\forall x p$ belong to L ,
- there is no other formula in L .

The other standard logical connectives and quantifiers are defined in the usual way. It is worth noting that there are no nested modalities in L . The explanation is that, although there are information systems, such as dialogue management systems, that contain in their “database” a representation of users’ beliefs, we are here thinking of databases in the standard way, as representing part of the “objective” world, i.e., not including agents’ beliefs. The set DBB of database beliefs is defined from DB as the set of sentences:

$$DBB = \{B_{db}p : \vdash DB \rightarrow p\} \cup \{\neg B_{db}p : \not\vdash DB \rightarrow p\}$$

It is assumed that the modal operator B_{db} is a normal operator, according to the Chellas classification [2]. The (D) schema is not imposed because it may happen, even if it is an undesirable situation, that the database contains inconsistent facts. Theorems of this logic are denoted by \vdash_B .

Let us call W a formal representation of the world in the language L_1 , i.e. W is a set of sentences of L_1 that represents the facts and properties of the actual world that can be represented in L_1 . From an intuitive point of view W is intended to represent the actual world, but that does not mean that the information system knows W . Since the actual world is unique, it is assumed that, for every sentence p in L_1 , we have either $W \vdash p$ or $W \vdash \neg p$. In general, only parts of the information represented in the database coincide with what is represented by W . The theory W is introduced in the logical framework to facilitate reasoning about the links between DB and the world.

Definition 3.1. Information validity. For a given state of the world W and of the database DB , we say that the database is valid for p iff we have ²:

$$W, DBB \vdash_B val(p) \tag{1}$$

where $val(p) \stackrel{\text{def}}{=} B_{db}p \rightarrow p$.

This definition can be extended to a set of facts of the form $p(x)$, where x may be a tuple of variables, by replacing the definition of $val(p)$ by: $val(p(x)) \stackrel{\text{def}}{=} \forall x(B_{db}p(x) \rightarrow p(x))$.

Definition 3.2. Information completeness. For a given state of the world W and of the database DB , we say that the database is complete for p iff we have:

$$W, DBB \vdash_B comp(p) \tag{2}$$

where $comp(p) \stackrel{\text{def}}{=} p \rightarrow B_{db}p$.

This definition can be extended to a set of facts of the form $p(x)$, where x may be a tuple of variables, by replacing the definition of $comp(p)$ by: $comp(p(x)) \stackrel{\text{def}}{=} \forall x(p(x) \rightarrow B_{db}p(x))$.

²According to Chellas (Definition 2.14, page 47 in [2]), we use the notation $T \vdash_B f$ to denote the fact that there exists in T a finite number of formulas A_1, \dots, A_n such that $\vdash_B A_1 \wedge \dots \wedge A_n \rightarrow f$.

The conditional form of $val(p)$ and $comp(p)$ leads to apparently surprising consequences. For instance, for the database DB we considered in section 2, DB is valid for $d(a)$ because we have $DBB \vdash_B \neg B_{db}d(a)$ and $W, DBB \vdash_B B_{db}d(a) \rightarrow d(a)$. For a similar reason, if we have $W \vdash \neg s(c)$, DB is complete for $s(c)$, because we have $W, DBB \vdash_B s(c) \rightarrow B_{db}s(c)$. But we can easily be convinced that these definitions are the right ones if we consider a database state which is not valid (respectively not complete) for p . Indeed, $\neg val(p)$ (respectively $\neg comp(p)$) is equivalent to $B_{db}p \wedge \neg p$ (respectively $p \wedge \neg B_{db}p$), which fits our intuition.

The following theorems express interesting properties that allow the derivation of the validity or completeness of some formulas from a set of assumptions about the validity or completeness of other formulas.

Theorem 3.1. *For all formulas p and q in L_1 we have:*

$$\vdash_B val(p) \wedge val(q) \rightarrow val(p \wedge q) \quad (3)$$

$$\vdash_B comp(p) \wedge comp(q) \rightarrow comp(p \wedge q) \quad (4)$$

$$\vdash_B comp(p) \wedge comp(q) \rightarrow comp(p \vee q) \quad (5)$$

We also have properties that show how the validity and the completeness of formulas are related.

Theorem 3.2. *For all formulas p and q in L_1 we have:*

$$\vdash_B (B_{db}p \vee B_{db}\neg p) \rightarrow (val(p) \rightarrow comp(\neg p)) \quad (6)$$

$$\vdash_B \neg(B_{db}p \wedge B_{db}\neg p) \rightarrow (comp(p) \rightarrow val(\neg p)) \quad (7)$$

Theorem 3.3 expresses some notable negative properties.

Theorem 3.3. *There are formulas p and q in L_1 such that:*

$$\not\vdash_B val(p) \wedge val(q) \rightarrow val(p \vee q) \quad (8)$$

$$\not\vdash_B val(p) \rightarrow val(\neg p) \quad (9)$$

$$\not\vdash_B val(p \vee q) \rightarrow val(p) \quad (10)$$

$$\not\vdash_B val(p \wedge q) \rightarrow val(p) \quad (11)$$

$$\not\vdash_B comp(p) \rightarrow comp(p \vee q) \quad (12)$$

$$\not\vdash_B comp(p) \rightarrow comp(\neg p) \quad (13)$$

$$\not\vdash_B comp(p \vee q) \rightarrow comp(p) \quad (14)$$

$$\not\vdash_B comp(p \wedge q) \rightarrow comp(p) \quad (15)$$

To prove Theorem 3.3 we just need to exhibit counter examples. For instance, for (8), we can check that if $DB = \{p \vee q\}$ and $\{\neg p, \neg q\} \subseteq W$, we have $val(p)$ and $val(q)$, but we do not have $val(p \vee q)$. For (9), a counter example is given by $DB = \{\neg p\}$ and $\{p\} \subseteq W$. More surprising is the property (11). A counter example is given by $DB = \{p\}$ and $\{\neg p\} \subseteq W$. To understand the reason why we have (11) we have to come back to the definitions of $val(p \wedge q)$ and of $val(p)$, which are $B_{db}(p \wedge q) \rightarrow p \wedge q$ and $B_{db}p \rightarrow p$, respectively. We then see that from the assumption $val(p \wedge q)$ we can infer $p \wedge q$, and therefore p , only if we are in a situation where the database believes $p \wedge q$. If the database believes p and does not believe q we cannot infer $p \wedge q$, and p may be false.

The notions of validity and completeness we have defined are about the links between the representation of the world and the representation of the database content. It is interesting to compare these notions with the notions of consistency and “self-completeness” that are defined as follows.

A database is consistent iff for every sentence p in L_1 we have either $DB \not\vdash p$ or $DB \not\vdash \neg p$, and a database is self-complete iff for every sentence p in L_1 we have either $DB \vdash p$ or $DB \vdash \neg p$.

In [4] it has been shown that the database is consistent iff for every sentence p in L_1 we have $DBB \vdash_B \neg(B_{db}p \wedge B_{db}\neg p)$, and it is self-complete iff for every sentence in L_1 we have $DBB \vdash_B B_{db}p \vee B_{db}\neg p$. Even though these two latter notions are independent of the state of the world, they are related to the notions of validity and completeness by the following properties:

$$\vdash_B val(p) \wedge val(\neg p) \rightarrow \neg(B_{db}p \wedge B_{db}\neg p) \quad (16)$$

and

$$\vdash_B comp(p) \wedge comp(\neg p) \rightarrow B_{db}p \vee B_{db}\neg p \quad (17)$$

Therefore, if the database is valid for every sentence in L_1 , it is consistent, and if it is complete for every sentence in L_1 , it is self-complete.

4. Constraints of different types

The intuitive meaning of the term “constraint” suggests the identification of a sub-set of a set of possibilities. Using the terminology of the usual semantics of modal logic, the set of all possibilities is the set of possible worlds, real or imaginary. The sub-sets picked out by constraints, we suggest, may be of two kinds.

First, there are sub-sets which conform to all “hard constraints” - constraints which express analytic truths (“All bachelors are unmarried men”), and constraints which express, or are consequences of, laws of nature. We shall here refer to hard constraints as “necessary truths”, for the reason that any possible world which could possibly be a *real* world must satisfy them: they are unfalsifiable in possible real worlds.

Secondly, there are the sub-sets which conform to “soft constraints” - constraints which express norms that indicate how things are in ideal worlds, how things ought to be. We shall

here refer to soft constraints as “deontic constraints”, and we note that such constraints are violable: there may be real worlds in which deontic constraints are not satisfied. The “unacceptable situations” mentioned in the example in section 2 correspond to violations of deontic constraints. For present purposes we are particularly concerned with deontic constraints which require database completeness or validity.

In what follows, we shall see how violations of deontic constraints may be characterised, how an information system can detect that a violation has occurred, and how violations may be repaired. The logical framework needed for these purposes will be presented in section 5.

We first have to make more precise the kinds of agents involved in information management. Data stored in the database comes from standard users of the information system who send to it commands for data insertion or data deletion. According to these commands the information system inserts or deletes data in the database, or eventually, rejects the command. So we can look upon the database as an agent whose set of beliefs is the set of data represented in the database, where these data in fact represent what standard users believe. In most information systems, when standard users send queries to the information system the queries are transmitted to the database agent, who computes answers according to his set of beliefs.

The information system can also be viewed as an agent who knows exactly what the content of the database is. The information system can also receive commands from a special kind of users called “database administrators”. With these commands the administrators can inform the information system of the deontic constraints for which it should detect violations, and also supply additional information that the information system can use in detecting violations. This additional information may contain necessary truths, or information about data validity or data completeness, or empirical information that can be used by the information system in the same way as necessary truths.

We introduce here several modalities whose formal definitions will be analysed later on. We use Op to represent the fact it is obligatory that p . This modality is used to represent deontic constraints. Necessary truths are represented by the modality N , and Np can be read: p is true in every possible real world. The information known by the information system is represented by the modality K_s , and $K_s p$ can be read: the information system knows that p .

4.1. Deontic constraints about information validity and information completeness

Let us now reconsider the situation which is unacceptable due to inconsistency between $DB \cup CWA$ and $IC1$. In our approach the situation can be represented as follows. The formula $IC1 : \forall x(c(x) \rightarrow d(x) \vee p(x))$ is classified as a necessary truth, and we have $N(\forall x(c(x) \rightarrow d(x) \vee p(x)))$. Data that are involved in the inconsistency are $c(a)$, $d(a)$ and $p(a)$, and, according to the database content, we have: $B_{db}c(a)$, $\neg B_{db}d(a)$, and $\neg B_{db}p(a)$.

We have seen that the inconsistency of $IC1 \cup DB \cup CWA$ is indirectly used to characterise the fact that $d(a)$ or $p(a)$ is missing in the database, if $c(a)$ is true of the world, or to characterise the fact that the database contains false data, if $c(a)$ is false of the world. It might be assumed, for

example, that the standard users who insert data of the form $c(x)$ are known by the information system to be reliable. Then, from $B_{db}c(a)$ we can infer that $c(a)$ is true of the world. Since we have $val(c(x))$ the implicit deontic constraints are about the completeness of $d(a)$ and $p(a)$.

The formal representations of these constraints which most immediately suggest themselves are $O(d(a) \rightarrow B_{db}d(a))$ and $O(p(a) \rightarrow B_{db}p(a))$. A point to be discussed in more detail in section 5, however, is that another possible formal representation of them is $d(a) \rightarrow OB_{db}d(a)$ and $p(a) \rightarrow OB_{db}p(a)$. It is worth noting that the situations where the constraints are violated are the same, whichever of these two formal representations is chosen. Indeed, in both cases a violation is characterised by $d(a) \wedge \neg B_{db}d(a)$ and $p(a) \wedge \neg B_{db}p(a)$.

In general deontic constraints about completeness of facts of the form $\phi(x)$ can be formally represented by: $O(\forall x(\phi(x) \rightarrow B_{db}\phi(x)))$ or by $\forall x(\phi(x) \rightarrow OB_{db}\phi(x))$. The former will be denoted by $O^1/comp(\phi(x))$ and the latter by $O^2/comp(\phi(x))$. Whenever it is not relevant to make explicit our choice, we shall use the notation $O/comp(\phi(x))$. Notice that for both representations a constraint violation is characterised by $\exists x(\phi(x) \wedge \neg B_{db}\phi(x))$, which is logically equivalent to $\neg comp(\phi(x))$.

Likewise, deontic constraints about validity can be formally represented either by $O(\forall x(B_{db}\phi(x) \rightarrow \phi(x)))$ or by $\forall x(\neg\phi(x) \rightarrow O\neg B_{db}\phi(x))$. These formal representations will respectively be denoted by $O^1/val(\phi(x))$ and by $O^2/val(\phi(x))$, and we shall use the notation $O/val(\phi(x))$ when the distinction is irrelevant.

Since the information system knows the database content and the constraints, we finally have:

$$\begin{aligned} &K_s B_{db}c(a), K_s \neg B_{db}d(a), K_s \neg B_{db}p(a), \\ &K_s(N(\forall x(c(x) \rightarrow d(x) \vee p(x))))), \\ &K_s val(c(x)), \\ &K_s O/comp(d(x)), K_s O/comp(p(x)). \end{aligned}$$

As has been mentioned above, violations of the deontic constraints for the car a are characterised by: $d(a) \wedge \neg B_{db}d(a)$ and $p(a) \wedge \neg B_{db}p(a)$.

The reasoning of the information system to detect deontic constraint violations should be the following. From the definition of val , and from $K_s val(c(x))$, we have $K_s(\forall x(B_{db}c(x) \rightarrow c(x)))$, and, if we accept the schema $K_s \forall x \phi(x) \rightarrow K_s \phi(a)$, we have $K_s(B_{db}c(a) \rightarrow c(a))$. Then, from $K_s B_{db}c(a)$, assuming that K_s is a normal modality, we have $K_s c(a)$. From $K_s(N(\forall x(c(x) \rightarrow d(x) \vee p(x))))$, assuming the logical validity of the schema $N\phi \rightarrow \phi$, we have $K_s(\forall x(c(x) \rightarrow d(x) \vee p(x)))$, and $K_s(c(a) \rightarrow d(a) \vee p(a))$. Then, with $K_s c(a)$, we have $K_s(d(a) \vee p(a))$, and from $K_s \neg B_{db}d(a)$ and $K_s \neg B_{db}p(a)$ we have $K_s((d(a) \vee p(a)) \wedge \neg B_{db}d(a) \wedge \neg B_{db}p(a))$, which entails $K_s((d(a) \wedge \neg B_{db}d(a)) \vee (p(a) \wedge \neg B_{db}p(a)))$. In classical logic $\phi(a) \rightarrow \exists x \phi(x)$ is a valid schema, so by the normality of K_s , we have $K_s(\exists x(d(x) \wedge \neg B_{db}d(x)) \vee \exists x(p(x) \wedge \neg B_{db}p(x)))$, that is, by definition of $comp(d(x))$ and of $comp(p(x))$, $K_s(\neg comp(d(x)) \vee \neg comp(p(x)))$.

That means that the information system knows that one of the two constraints $O/comp(d(x))$ or $O/comp(p(x))$ is violated. To know which one is violated it has to know, in addition, whether facts of the form $d(x)$ or of the form $p(x)$ can be assumed to be complete. For example, if it

knows that the database is complete for facts of type $p(x)$, i.e if we have $K_s(comp(p(x)))$, from $K_s \neg B_{db}p(a)$ it can be inferred that $K_s \neg p(a)$, and the only way to repair the violation is to insert $d(a)$. Similarly, if we have $K_s(comp(d(x)))$, to repair the violation $p(a)$ has to be inserted.

4.2. Deontic constraints about the world

Let us now reconsider now the constraint $IC3$, and suppose that it represents a deontic constraint about the world. We have $O(\forall x(o(x) \rightarrow s(x)))$, and the database beliefs are such that we have: $B_{db}o(c)$ and $\neg B_{db}s(c)$. A violation of this constraint for the car c is represented by $o(c) \wedge \neg s(c)$, because $o(c) \wedge \neg s(c)$ implies $\neg \forall x(o(x) \rightarrow s(x))$.

To be able to detect this violation the information system must be able to infer $o(c) \wedge \neg s(c)$ from $B_{db}o(c)$ and $\neg B_{db}s(c)$. That is possible, for example, if it knows that the database is valid for facts of the form $o(x)$ and complete for the facts of the form $s(x)$. In that case the information system knowledge is represented by:

$$\begin{aligned} &K_s B_{db}o(c), K_s \neg B_{db}s(c), \\ &K_s(O(\forall x(o(x) \rightarrow s(x))))), \\ &K_s val(o(x)), K_s comp(s(x)). \end{aligned}$$

Here from $K_s val(o(x))$ we have $K_s(B_{db}o(c) \rightarrow o(c))$, and with $K_s B_{db}o(c)$ we have $K_s o(c)$. In a similar way, from $K_s comp(s(x))$ we have $K_s(s(c) \rightarrow B_{db}s(c))$ and $K_s(\neg B_{db}s(c) \rightarrow \neg s(c))$. Then, with $K_s \neg B_{db}s(c)$, we have $K_s \neg s(c)$. Therefore, we have $K_s(o(c) \wedge \neg s(c))$, and $K_s(\neg(\forall x(o(x) \rightarrow s(x))))$

As we have mentioned before, to repair this violation it is the world which has to be changed, not the database. The only thing that the information system can do in this situation is to inform the users that a violation of this deontic constraint has occurred for the car c .

4.3. Deontic constraints about information self-completeness

The last example serves to illustrate deontic constraints about self-completeness, such as Reiter has proposed in [18]. For instance, the unacceptable situation due to the inconsistency of $IC1 \cup DB \cup CWA$, may be interpreted as a violation of the deontic constraint that informally says that for every x which is believed by the database to be a car, the database should be aware of whether it uses diesel oil or petrol. This constraint can be expressed by $O(\forall x(B_{db}c(x) \rightarrow B_{db}d(x) \vee B_{db}p(x)))$. Then, the information system knowledge is represented by:

$$\begin{aligned} &K_s B_{db}c(a), K_s \neg B_{db}d(a), K_s \neg B_{db}p(a), \\ &K_s(O(\forall x(B_{db}c(x) \rightarrow B_{db}d(x) \vee B_{db}p(x))))). \end{aligned}$$

In that case, since we have $K_s(B_{db}c(a) \wedge \neg B_{db}d(a) \wedge \neg B_{db}p(a))$, we also have $K_s(\exists x(B_{db}c(x) \wedge \neg B_{db}d(x) \wedge \neg B_{db}p(x)))$ and $K_s(\neg \forall x(B_{db}c(x) \rightarrow B_{db}d(x) \vee B_{db}p(x)))$. So the information system can detect the violation of this obligation without information about the world, or about data validity or completeness. To repair the violation there are three possibilities: to delete $c(a)$ from the database, to insert $d(a)$ or to insert $p(a)$. It is reasonable not to choose arbitrarily among these three possibilities, and a sensible justification would be to make the choice in the light of

the facts that are true or false of the world. For instance, if the information system knows that the database is valid for the facts of the form $c(x)$, i.e. $K_s val(c(x))$, it would be odd to repair the violation by deleting $c(a)$, though that would be acceptable from a formal logical point of view. Rational support for the choice is provided if two of the following properties hold: $K_s val(c(x))$, $K_s comp(d(x))$ or $K_s comp(p(x))$.

For instance, if we have $K_s val(c(x))$ and $K_s comp(d(x))$, from $K_s B_{db}c(a)$ and $K_s \neg B_{db}d(a)$ we can infer $K_s c(a)$ and $K_s \neg d(a)$. Then, to repair the violation $p(a)$ has to be inserted. If we have $K_s comp(d(x))$ and $K_s comp(p(x))$, we can infer $K_s \neg d(a)$ and $K_s \neg p(a)$, and $c(a)$ has to be deleted.

It may be remarked that, as in the first example, the same additional information is required in order to know which deontic obligation is violated and how to repair it. Is this an indication that the constraint $O(\forall x(B_{db}c(x) \rightarrow B_{db}d(x) \vee B_{db}p(x)))$, on one hand, and the set of constraints $O/val(c(x))$, $O/comp(d(x))$ and $O/comp(p(x))$, on the other hand, intuitively express the same constraint? We can easily see that the answer is “no”. Indeed, if we have, for example, $DB = \{c(a), d(a)\}$, and $p(a)$ is true of the world, since we have $B_{db}c(a) \wedge B_{db}d(a)$, the constraint $O(\forall x(B_{db}c(x) \rightarrow B_{db}d(x) \vee B_{db}p(x)))$ is satisfied, while the constraint $O/comp(p(x))$ is violated, because we have $p(a) \wedge \neg B_{db}p(a)$. Nevertheless, it may be that they are not independent. Indeed, the constraint $O(\forall x(B_{db}c(x) \rightarrow B_{db}d(x) \vee B_{db}p(x)))$ has to be justified by the truth of $\forall x(c(x) \rightarrow d(x) \vee p(x))$. And, if we adopt the second kind of representation of deontic constraints about validity and completeness, from $O^2/val(c(x))$, $O^2/comp(d(x))$ and $O^2/comp(p(x))$, and $\forall x(c(x) \rightarrow d(x) \vee p(x))$, we can infer in classical logic: $\forall x(O\neg B_{db}c(x) \vee OB_{db}d(x) \vee OB_{db}p(x))$, which entails: $\forall xO(B_{db}c(x) \rightarrow B_{db}d(x) \vee B_{db}p(x))$, if O is a normal operator. Then, if we accept the axiom schema $\forall xO\phi(x) \rightarrow O\forall x\phi(x)$, we finally have: $O(\forall x(B_{db}c(x) \rightarrow B_{db}d(x) \vee B_{db}p(x)))$.

4.4. Violation detection in general

We can easily generalise from the previous examples that deontic constraint violations are represented by conjunctions of formulas of the form p , $B_{db}p$ or $\neg B_{db}p$, where p is in L_1 .

Indeed, violations of constraints of the form $O/val(p)$ and $O/comp(p)$ are respectively represented by $B_{db}p \wedge \neg p$ and $p \wedge \neg B_{db}p$. Violations of constraints of the form $O(p \rightarrow q)$ are represented by $p \wedge \neg q$, and violations of constraints of the form $O(B_{db}p \rightarrow B_{db}q)$ are represented by $B_{db}p \wedge \neg B_{db}q$, where p and q are in L_1 .

Then, to detect violations, the information system may have to derive consequences of the form p , $B_{db}p$ or $\neg B_{db}p$, where p is in L_1 . How can the information system derive this information? We can infer $K_s p$ from $K_s Np$, or from $K_s val(p) \wedge K_s B_{db}p$, or from $K_s comp(\neg p) \wedge K_s \neg B_{db} \neg p$. Formulas of the form $K_s B_{db}p$ or $K_s \neg B_{db}p$ should respectively be derivable from $B_{db}p$ or $\neg B_{db}p$.

We have to pay special attention to the formal representation of sentences of the kind “every x should satisfy $\phi(x)$ ”. They can be represented by $O\forall x\phi(x)$. In this case]] there is a violation when $\neg\forall x\phi(x)$ holds, and a violation is detected by the information system when we have $K_s O\forall x\phi(x)$ and $K_s \neg\forall x\phi(x)$.

A sufficient condition for violation detection is that for some individual a the system knows

that we have $\neg\phi(a)$, i.e. $K_s\neg\phi(a)$. Indeed, if K_s is a normal operator, and if $\neg\phi(a) \rightarrow \exists x\neg\phi(x)$ is a valid formula, as in classical logic, from $K_s\neg\phi(a)$ we can infer $K_s\exists x\neg\phi(x)$ and $K_s\neg\forall x\phi(x)$. If the role of the information system is also to repair the violation, it has to know the particular individual a that causes the violation, and, in that case, the sufficient condition is also a necessary condition.

However, the sentence “every x should satisfy $\phi(x)$ ”, can also be formally represented by $\forall xO\phi(x)$. Then, $\forall xO\phi(x)$ entails for each individual a the obligation $O\phi(a)$, and the definition of a violation of $\forall xO\phi(x)$ is that for some individual a we have $\neg\phi(a)$. Then a violation is detected by the information system when we have $K_s\neg\phi(a)$ for some a .

We see that for both representations $K_s\neg\phi(a)$ is a necessary and sufficient condition for detecting and repairing a violation. For that reason we can consider that the distinction between them is not relevant, and that $O\forall x\phi(x)$ and $\forall xO\phi(x)$ can be considered to be logically equivalent.

5. Toward a general logical framework

In this section we describe and discuss a logical framework that would generate the derivations presented informally in the previous section. The formal language L , defined in section 3, is extended to incorporate the modalities K_s , N and O , in the same way as for the modality B_{db} . As was mentioned in section 3, the logic of modality B_{db} is assumed to be that of a normal modal logic. So we have:

$$B_{db}(\phi \rightarrow \psi) \rightarrow (B_{db}\phi \rightarrow B_{db}\psi) \quad (18)$$

The modalities K_s and N are also assumed to be normal. In addition, their logics will contain the axiom schema (T). Then we have:

$$K_s(\phi \rightarrow \psi) \rightarrow (K_s\phi \rightarrow K_s\psi) \quad (19)$$

$$K_s\phi \rightarrow \phi \quad (20)$$

$$N(\phi \rightarrow \psi) \rightarrow (N\phi \rightarrow N\psi) \quad (21)$$

$$N\phi \rightarrow \phi \quad (22)$$

To represent formally the fact that the information system knows the database content, we adopt the schemas:

$$B_{db}\phi \rightarrow K_sB_{db}\phi \quad (23)$$

$$\neg B_{db}\phi \rightarrow K_s\neg B_{db}\phi \quad (24)$$

Notice that from (20), (23) and (24) we have $K_sB_{db}\phi \leftrightarrow B_{db}\phi$ and $K_s\neg B_{db}\phi \leftrightarrow \neg B_{db}\phi$. For the formalisation of the deontic modality O we can consider several possibilities. The simplest is to assume that the logic of O is that of a normal modal logic. Then we have:

$$O(\phi \rightarrow \psi) \rightarrow (O\phi \rightarrow O\psi) \quad (25)$$

The question now is how to represent the different kinds of deontic constraints. We start with the analysis of deontic constraints about data validity and data completeness.

Let's suppose first that these constraints are respectively represented by $O^1/val(p)$ and $O^1/comp(p)$, that is by: ³ $O(B_{db}p \rightarrow p)$ and $O(p \rightarrow B_{db}p)$. In that case we can draw counter intuitive conclusions. Consider, for example, a situation where the database should be complete for some given formula p , and there is a deontic constraint about the world which says that p should hold, and the later constraint is violated, i.e. $\neg p$ holds, and the database does not believe p . This situation is formally represented by:

$$O(p \rightarrow B_{db}p), Op, \neg p, \neg B_{db}p.$$

For instance, the meaning of the formula p could be that John has a driving licence, and the situation is that John has no driving licence, and the database does not believe that he has a driving licence, which satisfies the deontic constraint about database completeness, but violates the deontic constraint about the world. At this level, this formal description of the situation seems to be acceptable.

However, from $O(p \rightarrow B_{db}p)$ and Op , we can infer $OB_{db}p$ because O is normal, and the derived obligation $OB_{db}p$ is violated because we have $\neg B_{db}p$. Should we have to repair this violation, that is to change the database content in such a way that the database believes p ? Certainly not, because p is not true of the world. The only obligation that is violated is Op , but the obligation about the completeness for p is not violated.

A similar problem arises in a situation formally represented by:

$$O(B_{db}p \rightarrow p), O\neg p, p, B_{db}p.$$

Here, from $O(B_{db}p \rightarrow p)$ and $O\neg p$ we can infer the obligation $O\neg B_{db}p$, which is violated, since we have $B_{db}p$. Suppose the meaning of p is that John has a gun at home. The obligation that he should not have a gun at home is violated, but the database content correctly represents the situation, and there is no violation of the obligation about validity for p .

We can conclude that, if O is a normal operator, deontic constraints about validity or completeness must be represented in a different way.

The second possibility is to adopt the second formal representation $O^2/val(p)$ and $O^2/comp(p)$, that is: $\neg p \rightarrow O\neg B_{db}p$, which can be read: if p is false of the world the database should not believe p , and $p \rightarrow OB_{db}p$, which can be read: if p is true of the world the database should believe p . With these forms for the deontic constraints, the first situation is represented by:

$$p \rightarrow OB_{db}p, Op, \neg p, \neg B_{db}p.$$

and we cannot infer the obligation $OB_{db}p$ about the database content. The second situation is represented by:

$$\neg p \rightarrow O\neg B_{db}p, O\neg p, p, B_{db}p.$$

and we cannot infer the obligation $O\neg B_{db}p$ about the database content. Then, the problems just described disappear.

³Notice that here p denotes a given fixed formula.

It may be noted that if we have deontic constraints about the validity and completeness of some given formula p , we have: $p \rightarrow OB_{db}p$ and $\neg p \rightarrow O\neg B_{db}p$, and then we have $\neg O\neg B_{db}p \rightarrow p$. If we accept the axiom schema $OB_{db}\phi \rightarrow \neg O\neg B_{db}\phi$, which seems reasonable, with $\neg O\neg B_{db}p \rightarrow p$ we have $OB_{db}p \rightarrow p$, and from $p \rightarrow OB_{db}p$ we have $p \leftrightarrow OB_{db}p$. From $\neg O\neg B_{db}p \rightarrow p$ and $p \leftrightarrow OB_{db}p$ we also have $\neg O\neg B_{db}p \rightarrow OB_{db}p$, and from the schema $OB_{db}\phi \rightarrow \neg O\neg B_{db}\phi$ we have $\neg O\neg B_{db}p \leftrightarrow OB_{db}p$. If we use P to denote permission - where this operator is defined, as is usual, as the dual of the obligation operator - we finally have: $p \leftrightarrow OB_{db}p$ and $OB_{db}p \leftrightarrow PB_{db}p$. Were that to be an axiom schema, it would be unacceptable; but these properties hold only for a given sentence for which there are deontic constraints regarding both its validity and its completeness.

Let us see now how to represent deontic constraints about self-completeness. The first possibility is to represent them by formulas of the form $O(B_{db}p \rightarrow B_{db}q)$, or $O(B_{db}p \rightarrow B_{db}q \vee B_{db}r)$, or $O(B_{db}p \rightarrow \exists x B_{db}q(x))$. However, if we consider a situation formally represented by:

$$p \rightarrow OB_{db}p, O(B_{db}p \rightarrow B_{db}q), p, \neg B_{db}p, \neg B_{db}q,$$

from $p \rightarrow OB_{db}p$ and p we can infer $OB_{db}p$, which is violated, because we have $OB_{db}p$. Moreover, since O is a normal operator, from $OB_{db}p$ and $O(B_{db}p \rightarrow B_{db}q)$ we have $OB_{db}q$, and the obligation $OB_{db}q$ is also violated because we have $\neg B_{db}q$. To repair this second violation the database should be changed in order to have $B_{db}q$. But it is counter intuitive to impose this database change as long as the database does not believe p . That is, the obligation $OB_{db}q$ should be conditioned by the fact that $B_{db}p$ holds. Therefore, we think that a better formalisation of deontic constraints about self-completeness should have the form $B_{db}p \rightarrow OB_{db}q$, or $B_{db}p \rightarrow O(B_{db}q \vee B_{db}r)$, or $B_{db}p \rightarrow O(\exists x B_{db}q(x))$. With this formal representation the previous situation is represented by:

$$p \rightarrow OB_{db}p, B_{db}p \rightarrow OB_{db}q, p, \neg B_{db}p, \neg B_{db}q.$$

Here, since the database does not believe p we cannot infer that it should believe q , and, as expected, there is no violation of the constraint about self-completeness.

Finally, we may also note that - using these formal representations of deontic constraints about validity, completeness and self-completeness - we cannot derive deontic constraints about the world. If the latter are represented by formulas of type $O(p \rightarrow q)$, then - since we cannot infer Op from the other types of constraints - there is no risk of interference in the form of the derivation of intuitively unacceptable obligation sentences.

In section 4.4 specific issues related to quantifiers and modal operators have been mentioned. We think that for our purpose we can accept the simplest interpretation of quantifiers, and assume that in possible worlds structures the same domain is assigned to all the worlds of a given structure (see [8, 6]). This position is exactly formalised in the axiomatics by Barcan formulas and converse Barcan formulas. That is, for each normal modal operator M : K_s , B_{db} and O , we accept the axiom schema:

$$\forall x M\phi(x) \leftrightarrow M\forall x\phi(x) \tag{26}$$

It is also assumed that constant symbols have rigid interpretations, that is they are inter-

preted by the same domain element in every world of a given structure. With this additional assumption the following axiom schemas are validated for formulas ϕ where the modal operators are normal operators:

For every a :

$$\forall x \phi(x) \rightarrow \phi(a) \quad (27)$$

$$\phi(a) \rightarrow \exists x \phi(x) \quad (28)$$

In this logic the different types of deontic constraints and their violations are formally represented as follows:

1. deontic constraints about validity: $\forall x(\neg p(x) \rightarrow O\neg B_{db}p(x))$, violation: $\neg p(a) \wedge B_{db}p(a)$ for some a ,
2. deontic constraints about completeness: $\forall x(p(x) \rightarrow OB_{db}p(x))$, violation: $p(a) \wedge \neg B_{db}p(a)$ for some a ,
3. deontic constraint about self-completeness: $\forall x(B_{db}p(x) \rightarrow OB_{db}q(x))$, violation: $B_{db}p(a) \wedge \neg B_{db}q(a)$ for some a ,
4. deontic constraints about the world: $O\forall x(p(x) \rightarrow q(x))$, violation: $p(a) \wedge \neg q(a)$ for some a .

The violation of deontic constraints about validity is detected by the information system if we have $K_s(\forall x(\neg p(x) \rightarrow O\neg B_{db}p(x)))$ and $K_s(\neg p(a) \wedge B_{db}p(a))$. Indeed, from the schema (27), $\forall x(\neg p(x) \rightarrow O\neg B_{db}p(x)) \rightarrow (\neg p(a) \rightarrow O\neg B_{db}p(a))$ is a valid formula, and since K_s is normal we have $K_s(\forall x(\neg p(x) \rightarrow O\neg B_{db}p(x))) \rightarrow (\neg p(a) \rightarrow O\neg B_{db}p(a))$. Then, from $K_s(\forall x(\neg p(x) \rightarrow O\neg B_{db}p(x)))$ we can infer $K_s(\neg p(a) \rightarrow O\neg B_{db}p(a))$. From $K_s(\neg p(a) \wedge B_{db}p(a))$ we have $K_s(\neg p(a))$, and with $K_s(\neg p(a) \rightarrow O\neg B_{db}p(a))$ we have $K_s(O\neg B_{db}p(a))$. From $K_s(\neg p(a) \wedge B_{db}p(a))$ we also have $K_s(B_{db}p(a))$. Then the information system knows that there is a violation of $O\neg B_{db}p(a)$.

The violation detection can be made explicit in a different way. From the axiom schema (28), $\neg p(a) \wedge B_{db}p(a) \rightarrow \exists x(\neg p(x) \wedge B_{db}p(x))$ is a valid formula, and since K_s is normal we have $K_s(\neg p(a) \wedge B_{db}p(a) \rightarrow \exists x(\neg p(x) \wedge B_{db}p(x)))$. Then, from $K_s(\neg p(a) \wedge B_{db}p(a))$ we have $K_s(\exists x(\neg p(x) \wedge B_{db}p(x)))$, and $K_s(\neg(\forall x(\neg p(x) \rightarrow O\neg B_{db}p(x))))$.

Violation detection for the other types of deontic constraints can be formalised in a very similar way.

6. Conclusion

We have suggested in this paper that there are two principal categories of constraints: those which express necessary truths, in the sense that they are unfalsifiable in real situations; and those which express norms, the deontic constraints. The latter have been classified into three different types: deontic constraints about the world, deontic constraints about the representation

of the world (self-completeness), and deontic constraints about the links between the world and its representation (validity and completeness).

A modal logical framework has been proposed to define the different types of constraints and their violations. In this framework the modal operators K_s , B_{db} , and O are normal operators. The general form of deontic constraints about validity, completeness and self-completeness is: $\phi \rightarrow O\psi$. We have shown that the form $O(\phi \rightarrow \psi)$ leads to counter-intuitive consequences, except for deontic constraints about the world. For quantifiers we have adopted the simplest solution, which is to assume that for every normal operator M , $\forall x M\phi(x)$ and $M\forall x\phi(x)$ are logically equivalent. This assumption may deserve closer inspection, in particular for the deontic modal operator.

The problems raised by “integrity constraints” in the context of information systems could be generalised to the problems of the links between the world and its representation for any kind of agent. The concepts we have introduced and their formal definitions could be used, for example, for the analysis of the following scenario.

Consider two agents: a car driver and a policeman. We suppose that the car speed is 65km/h, and that there is a sign that indicates that car speed should be less than 60km/h. The “driver’s database” may contain information about the car speed that is acquired by the driver when he watches the speedometer. The driver knows that the speed limitation is 60km/h, and that the maximum speed of his car is 180km/h. In his “database” he has the information that the car speed is 59km/h. However, it is not the case that the driver knows that his database is valid for the car speed, because the speedometer is not perfectly reliable, and it is not the case that he knows that his database is complete for the car speed, because the driver is not always watching the speedometer. The “policeman’s database” contains the fact that the car speed is 65km/h. The car-speed information in the policeman’s database comes from a radar system that measures car speeds, and the policeman is always watching the radar. For these reasons, the policeman knows that his database is valid and complete for the car speed. Of course, the policeman also knows that the speed limitation is 60km/h.

The scenario might be significantly complicated if we want to describe a dialogue between the car driver and the policeman about the car speed, in particular in the case where the agents do not necessarily trust each other agents about their sincerity and credibility (see [5] for a formal analysis of these notions).

In future work we may extend our framework, in order to be able to formalise this kind of scenario, with potential applications to multi-agent systems.

References

- [1] J. Carmo and A.J.I. Jones. Deontic Database Constraints and the Characterisation of Recovery. In A.J.I.Jones and M.Sergot, editors, *2nd Int. Workshop on Deontic Logic in Computer Science*, pages 56–85. Tano A.S., 1994.
- [2] B. F. Chellas. *Modal Logic: An introduction*. Cambridge University Press, 1988.

- [3] C. Date. *Introduction to Database Systems*. Addison-Wesley, 1995.
- [4] R. Demolombe. Database validity and completeness: another approach and its formalisation in modal logic. In E. Franconi and M. Kifer, editor, *Proc. of the 6th International Workshop on Knowledge representation meets Data Bases*, 1999.
- [5] R. Demolombe. To trust information sources: a proposal for a modal logical framework. In C. Castelfranchi and Y-H. Tan, editor, *Trust and Deception in Virtual Societies*. Kluwer, 1999.
- [6] M. Fitting and R. L. Mendelsohn. *First-Order Modal Logic*. Kluwer, 1998.
- [7] G. Gardarin and P. Valduriez. *Relational Databases and Knowledge Bases*. Addison-Wesley, 1989.
- [8] J. W. Garson. Quantification in Modal Logic. In D. Gabbay and F. Guenther, editor, *Handbook of Philosophical Logic, Extensions of Classical Logic, Vol II*. D. Reidel Publishing Company, 1984.
- [9] M. Gertz. Managing data quality and integrity in Federated Databases. In S. Jajodia and W. List and G. W. McGregor and L. Strous, editor, *IFIP TC11 Conference on: Integrity and Internal control in Information Systems*. D. Reidel Publishing Company, 1998.
- [10] P. Godfrey, J. Grant, J. Gryz, and J. Minker. Integrity constraints: semantics and applications. In J. Chomicki and G. Saake, editors, *Logic for Databases and Information Systems*. Kluwer, 1998.
- [11] R. Kowalski. Logic for data description. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*. Plenum Press, 1978.
- [12] K. Kwast. A deontic operator for database integrity. In J-J. Ch. Meyer and R.J. Wieringa, editors, *Deontic Logic in Computer Science*. University of Amsterdam, 1991.
- [13] J-J. Ch. Meyer, R.J. Wieringa, and F. Dignum. The role of Deontic Logic in the specification of Information Systems. In J. Chomicki and G. Saake, editors, *Logic for Databases and Information Systems*. Kluwer, 1998.
- [14] A. Motro. Integrity = Validity + Completeness. *ACM TODS*, 14(4), 1989.
- [15] J-M. Nicolas and K. Yazdanian. Integrity checking in Deductive Databases. In H. Gallaire and J. Minker, editors, *Logic and Databases*. Plenum, 1982.
- [16] A. Olivé. Integrity checking in Deductive Databases. In *17th Int. Conf. on Very Large Data Bases*, 1991.
- [17] R. Reiter. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages*. Springer Verlag, 1983.
- [18] R. Reiter. What Should a Database Know? *Journal of Logic Programming*, 14(2,3), 1992.
- [19] E. Teniente and A. Olivé. The Events Method for View Updating in Deductive Databases. In *Proceedings of the International Conference on Extending Data Base Technology*, 1992.
- [20] R. van der Meyden. Logical approaches to incomplete information: a survey. In J. Chomicki and G. Saake, editors, *Logic for Databases and Information Systems*. Kluwer, 1998.
- [21] R.J. Wieringa and J-J. Meyer. Applications of Deontic Logic in Computer Science: a concise overview. In J-J. Meyer and R.J. Wieringa, editors, *Proc. First Int. Workshop on Deontic Logic in Computer Science*, 1991.