Integrity Constraints Revisited (Preliminary version)

Robert Demolombe
ONERA/CERT
Toulouse

Andrew Jones
Department of Philosophy
University of Oslo

June 25, 1993

1 Introduction

When computerised databases began to be used in business applications, there naturally arose a concern with how the integrity the stored data could be checked. This task was implemented by large sets of programs, depending on each particular application.

Relational DBMS have significantly improved the situation in supporting automated integrity checking, based on some particular kinds of Integrity Constraints (ICs). Despite this significant practical improvement many researchers realized that the role and status of Integrity Constraints should be defined more precisely. This question is presently more important since Deductive DBMS prototypes have been implemented, and the justification of the distinctions between the role played by the rules that are used to derive answers, and the role played by the rules that are used to check updates, also needs to be clarified.

There are many papers in the literature that discuss such issues as: should ICs be characterized in terms of syntactical criteria like: atomic facts vs. general rules, or Horn clauses vs. non Horn clauses?, How are IC violations to be formally characterized? What is the intuitive interpretation of IC violations? What is an ICs epistemic status?.

The objective of this paper is to try to provide answers to such kinds of questions. We have tried to understand what the status and role of Integrity

Constraints are, and we propose formal definitions and properties to represent our interpretation. However, algorithms for integrity checking, and any considerations of performance, are outside the scope of this paper.

We start in Section 2 with an analysis, based on a simple example, of the standard treatment of ICs. Then we recall in Section 3 how Reiter has provided a clear definition of the epistemic status of ICs with regard to the Database content. We also show that in his approach the ICs are implictly supported by statements that are guaranteed to be true. This property is explicit in the standard view of ICs, and in section 4 we show how the standard view of ICs can be refined in the light of Reiter's work. In section 4 are also given formal definitions of the properties that have to be enforced using ICs, namely: Validity and Completeness. General formal results are presented that can be used to support techniques to check these properties. Finally in section 5 we propose to split the overall information involved in Integrity Checking into three components called: DB, SAF and IC. DB represents a description of the world that is not necessarily guaranteed to be correct, SAF represents the information about the world that is guaranteed to be true, and which is used to check violations of Validity or Completeness, and IC defines the parts of DB for which the property of Validity or Completeness has to be enforced.

2 Standard view of Integrity Constraints

The standard view of Integrity Constraints in the field of Relational Data Bases is that ICs represent true statements about the world, and that they are used to check if there are some false statements in the description of the world stored in the Data Base DB [?, ?, ?, ?, ?].

To illustrate the role played by ICs, we consider a very simple example, where DB is a First Order Theory, whose language is defined from the predicate expressions: p(x), m(x) and w(x), whose respective meanings are: x is a person, x is a man, and x is a woman.

The set of Integrity Constraints IC for DB is the set of sentences:

IC1: $\forall x(p(x) \rightarrow m(x) \vee w(x))$

IC2: $\forall x(m(x) \wedge w(x) \rightarrow)$

IC3: $\forall x(m(x) \rightarrow p(x))$

and the Database content is:

$$DB = \{ p(a), m(b), w(b), m(c) \}$$

In this DB state the constraint IC2 is said to be violated because we have:

$$DB \vdash m(b) \wedge w(b) \quad \text{and} \quad IC2 \vdash \neg(m(b) \wedge w(b))$$

Indeed, the inconsistency between IC2 and DB shows that there is something wrong.

Since it is implicitly assumed that Integrity Constraints are true in the world, the standard interpretation of the inconsistency is that either m(b) or w(b) is false in the world, and this observation suggests that one of them should be removed from DB.

The DB content is also considered as unacceptable with regard to the fact m(c) and IC3. Indeed, if the question: "is p(c) true ?" is asked of DB, the answer will be: "we don't know", while from DB and IC3, it is known by the system that the answer is: "yes". In formal terms we have:

$$DB \nvdash p(c) \quad \text{and} \quad DB, IC \vdash p(c)$$

This situation is interpreted, in the standard view, as a lack of information in DB, and this suggests that p(c) should be inserted in DB.

The DB content can be shown to be unacceptable in a different way if CWA [?] is assumed. Under this assumption we have:

$$CWA, DB \vdash m(c) \wedge \neg p(c) \quad \text{and} \quad IC \vdash m(c) \rightarrow p(c)$$

In that case we have an inconsistency, and it is more obvious that the DB content is unacceptable. Nevertheless the reason why it is not acceptable is different. The inconsistency, here, is caused by the CWA which allows the inference of $\neg p(c)$ from DB. In informal terms, the CWA has the same effect as inserting $\neg p(c)$ in DB. But this "insertion" leads to an inconsistency, and this suggests that $\neg p(c)$ should be "removed". However, in the context of CWA, either p(c) or $\neg p(c)$ is derivable from DB, and to "remove" $\neg p(c)$, p(c) has to be inserted.

We see that whether or not CWA is adopted we are lead to the same conclusion, that is: p(c) should be inserted.

A similar situation arises with the fact p(a). Since, we have:

$$DB \not\vdash m(a) \vee w(a) \quad \text{and} \quad DB, IC \vdash m(a) \vee w(a)$$

the sentence $m(a) \vee w(a)$ should be derivable from DB. In the context of a standard Relational Database that suggests that either m(a) or w(a) should be inserted in DB. We get the same conclusion if the CWA is adopted, because the CWA plays the same role as "inserting" $\neg m(a)$ and $\neg w(a)$, and $p(a) \wedge \neg m(a) \wedge \neg w(a)$ is inconsistent with IC1. So, to remove this inconsistency, either m(a) or w(a) should be inserted.

This example shows that a DB state may be unacceptable because it is inconsistent with Integrity Constraints, or because different answers to a query will be given depending on whether or not the ICs are used. In the latter case, if a sentence p can be infered from: DB∪IC, and cannot be infered from DB alone, then the sentence $\neg p$ can be derived from: CWA ∪ DB, and CWA ∪ DB ∪ IC is inconsistent. The explanation of the inconsistency is that the "insertion" of $\neg p$, due to CWA, inserts false information.

We have seen that inconsistencies can be interpreted in different ways. Sometimes they indicate that false facts are in DB which have to be removed; sometimes, they indicate a lack of information in DB, and some facts have to be inserted. In the literature there are several proposed definitions for Integrity Constraint violation. For some authors there is a violation if IC ∪ DB is not consistent, for others there is a violation if IC is not a logical consequence of DB. Notice that the two definitions are equivalent if CWA is assumed.

From this analysis of the standard view of Integrity Constraints we can conclude that two questions need clearer answers: *what is the epistemic status of Integrity Constraints ?* And: *how are Integrity Constraint violations to be interpreted ?.*

3 Reiter's view of Integrity Constraints

In his paper: "What should a Database know?" [?], Reiter proposes a clear answer to the question of the status of Integrity Constraints. In his view the Data

Base DB is a set of statements about the world, that are formally represented by First Order sentences, and the ICs are a set of statements about the DB content that are formally represented by First Order Epistemic sentences, where the epistemic modality K is formalized by the KFOPCE Logic defined by Levesque [?]. Sentences of the form: Kp are intended to mean: the Data Base DB believes that p holds.

The epistemic sentences allow one to impose constraints about what DB believes. For instance the Integrity Constraints IC1, IC2 and IC3 can be reformulated as:

KIC1: $\forall x(Kp(x) \rightarrow Km(x) \vee Kw(x))$

KIC2: $\forall x(Km(x) \wedge Kw(x) \rightarrow)$

KIC3: $\forall x(Km(x) \rightarrow Kp(x))$

The Data Base DB is represented as in the standard approach:

DB = { p(a), m(b), w(b), m(c) }

The intuitive meaning of the constraint KIC1 is that if DB believes that some x is a person, then either it should believe that x is a man, or it should believe that x is a woman. That is, a DB state where there is some person "in" DB such that DB ignores if he/she is a man or a woman is not acceptable. The role of this constraint is to prevent DB ignorance about the sex of people. The constraint KIC3 plays a similar role.

The constraint KIC2 plays a different role. It prevents DB states where DB believes that some person is both a man and a woman. The reason is that we know that in this situation one of these two DB beliefs is not true in the world.

Roughly speaking there are constraints to prevent the **lack of DB beliefs**, and there are others to prevent **false DB beliefs**. From a technical point of view the two cases lead to inconsistencies in the KFOPCE Logic, even if CWA is not assumed. Indeed this Logic has the following property. If p is a First Order sentence, we have:

$$DB|\approx Kp \text{ iff } DB \vdash p$$
$$DB|\approx \neg Kp \text{ iff } DB \nvdash p$$

where: $DB|\approx Kp$ means that Kp logically follows from DB in the KFOPCE

Logic.

In this framework, a DB state is acceptable iff DB$|\approx$ IC holds. In the example we have: DB$|\approx$ Kp(a) $\wedge$ $\neg$Km(a) $\wedge$ $\neg$Kw(a); then IC1 is violated, and since we have: DB$|\approx$ Km(b) $\wedge$ Kw(b), IC2 is also violated. Though the two violations are represented by the same formal property, i.e. an inconsistency, they have two different interpretations. For this reason we think that even if Reiter proposes a clear answer to the question: *what is the status of Integrity Constraints ?*, the answer to the question: *how are Integrity Constraint violations to be interpreted ?* has to be refined.

Another issue that needs clarification in his approach is the justification of integrity constraints. It is clear that integrity constraints are constraints about the DB content, not about the world. Nevertheless they are supported by true beliefs about the world. For example the constraint: KIC1: $\forall$x(Kp(x) $\rightarrow$ Km(x) $\vee$ Kw(x)) is supported by the fact that we know that: $\forall$x(p(x) $\rightarrow$ m(x) $\vee$ w(x)) is true in the world. So the question: *how are Integrity Constraints supported ?* needs clarification.

4 Refinement of the Standard view

We want to exhibit the kind of role Integrity Constraints are intended to play in regard to checking some properties of the DB content. The properties to be checked have been defined, in the context of standard Relational Data Bases, by Motro in his paper: "Integrity = Validity + Completeness". The formal definitions of these properties we present in this paper are more general, in the sense that DB can be any kind of First Order Theory, and that their logical definitions allow one to reason about them. For example, in the formalism presented below it is possible to demonstrate how the Validity of a conjunction is related to the Validity of its conjuncts.

From an intuitive point of view we say that DB is Valid vis-à-vis a sentence p if, whenever DB believes p, then p is true in the world, and we say that DB is Complete vis-à-vis p if, whenever p is true in the world, then p is believed by DB.

For the definition of these properties we define DB as a consistent First Order Theory, and from DB we define DBB, the set of DB beliefs as:

$$DBB = \{Bp \ : \ DB \vdash p\} \cup \{\neg Bp \ : \ DB \nvdash p\}$$

where B is defined by the Logic KD45 [?].

Let W be a First Order Theory that has a unique model (up to an isomorphism). W is intended to represent the world. The concepts of Validity and Completeness are formally defined by the definitions 1 and 2.

Definition 1: Validity of DB vis-à-vis a sentence p.

Let W and DBB be the representations of the world, and of the DB beliefs, respectively; we say that the sentence p is Valid, with respect to W and DB, iff $W, DBB \vdash Bp \rightarrow p$ holds.

In formal terms we have:

Val(W,DB,p) iff $W, DBB \vdash Bp \rightarrow p$.

Notice that $Bp \rightarrow p$ is logically equivalent to $\neg Bp \vee p$. As a consequence, DB is valid vis-à-vis p whenever $\neg Bp$ belongs to DBB. In informal terms, Val(W,DB,p) holds whenever DB has no opinion about p.

Invalidity is defined by:

Inval(W,DB,p) iff $W, DBB \vdash (Bp) \wedge \neg p$

The Property1 shows that Invalidity, as it is defined, corresponds to the opposite property of Validity.

Property 1:

Inval(W,DB,p) holds iff Val(W,DB,p) does not hold [1].

The definition of Validity can be extended to open formulas, of the form p(x), with free variable x. In that case the definition says that all the instances of p(x) are Valid:

Val(W,DB,p(x)) iff $W, DBB \vdash \forall x(Bp(x) \rightarrow p(x))$

The same extension holds for Invalidity:

[1] Proofs of all the presented properties are given in the Appendix

Inval(W,DB,p(x)) iff $W, DB \vdash \exists x(Bp(x) \wedge \neg p(x))$

The definition of Completeness is quite similar to Validity.

Definition 2: Completeness of DB vis-à-vis a sentence p.

Let W and DBB be the representations of the world, and of the DB beliefs, respectively; we say that the sentence p is Complete with respect to W and DB iff
$W, DBB \vdash p \rightarrow Bp$ holds.

In formal terms we have:

Comp(DB,W,p) iff $W, DBB \vdash p \rightarrow Bp$.

Notice that $p \rightarrow Bp$ is logically equivalent to $\neg p \vee Bp$. As a consequence, DB is Complete vis-à-vis p, provided p is not true in the world, even if DB has no opinion about p,

Incompleteness is defined by:

Incomp(DB,W,p) iff $W, DBB \vdash p \wedge \neg Bp$

The Property2 shows that Incompleteness, as it is defined, corresponds to the opposite property of Completeness.

Property 2:

Incomp(W,DB,p) holds iff Comp(W,DB,p) does not hold.

As for Validity, extensions of the definitions to open formulas lead to:

Comp(DB,W,p(x)) iff $W, DBB \vdash \forall x(p(x) \rightarrow Bp(x))$.

Incomp(DB,W,p(x)) iff $W, DBB \vdash \exists x(p(x) \wedge \neg Bp(x))$

If W and DB are clearly defined in a given context, Val(W,DB,p), Inval(W,DB,p), Comp(W,DB,p), and Incomp(W,DB,p), are respectively abbreviated to: Val(p), Inval(p), Comp(p) and Incomp(p).

We have proved that the following properties hold for Validity and Completeness.

Properties of Validity:

The properties listed below show how the concept of Validity behaves with respect to the logical connectives. Some of them, like for example Property V.4, are a bit surprising. In fact they can be understood if we keep in mind the specific logical properties of the belief modality and the fact that Validity is defined in the conditional form: $Bp \rightarrow p$. So, for instance, for (V.4), the fact that $Val(p \vee q)$ is not entailed by the conjunction of the two conditional sentences $Bp \rightarrow p$, $Bq \rightarrow q$, turns essentially on the fact that $B(p \vee q) \rightarrow (p \vee q)$ is not valid.

As a shorthand, in the following, connectives in the meta language are denoted like connectives in the object language. For example "Val(p) and Val(q)" is denoted "$Val(p) \wedge Val(q)$".

(V.1) $Val(p) \wedge Val(q) \Rightarrow Val(p \wedge q)$

(V.2) $Val(p \vee q) \Rightarrow Val(p) \vee Val(q)$

(V.3) $Inval(p \wedge q) \Rightarrow Inval(p) \vee Inval(q)$

(V.4) $Val(p) \wedge Val(q) \not\Rightarrow Val(p \vee q)$

(V.5) $Val(p) \not\Rightarrow Val(\neg p)$

(V.6) $Val(p \vee q) \not\Rightarrow Val(p)$

(V.7) $Val(p \wedge q) \not\Rightarrow Val(p)$

Properties of Completeness:

(C.1) $Comp(p) \wedge Comp(q) \Rightarrow Comp(p \wedge q)$

(C.2) $Comp(p) \wedge Comp(q) \Rightarrow Comp(p \vee q)$

(C.3) $Incomp(p \vee q) \Rightarrow Incomp(p) \vee Incomp(q)$

(C.4) $Incomp(p \wedge q) \Rightarrow Incomp(p) \vee Incomp(q)$

(C.5) $Comp(p \wedge q) \not\Rightarrow Comp(p)$

(C.6) $Comp(p) \not\Rightarrow Comp(\neg p)$

(C.7) $\text{Comp}(p \vee q) \not\Rightarrow \text{Comp}(p)$

(C.8) $\text{Comp}(p \vee q) \not\Rightarrow \text{Comp}(p) \vee \text{Comp}(q)$

Properties of Validity and Completeness:

(VC.1) If DB is consistent: $\text{Comp}(\neg p) \Rightarrow \text{Val}(p)$

(VC.2) If DB is complete: $\text{Val}(p) \Rightarrow \text{Comp}(\neg p)$

5 Refinement of Reiter's view

The definitions of Validity and Completeness we have presented in the previous section refer to W, but one may note that usually a system does not know how things are in the world, unless the system is told that such and such part of DB are true of the world. This is the case for what, in the standard view, are called Integrity Constraints. In fact ICs can be used to check the properties of Validity or Completeness because they are guaranteed to be true in the world. Indeed, in the standard view, it is assumed that we have: $W \vdash IC$.

According to Reiter's view it is not appropriate to identify Integrity Constraints with sentences that have the property of being true description of the world. For this reason, in the following, we call SAF the set of sentences that are known by the system to be true of the world. So, SAF has the formal property:

$$W \vdash SAF$$

In other words, DB represents a set of **beliefs of the system**, and SAF a set of **true beliefs of the system**. From a theoretical point of view, SAF is a subset of DB, because true beliefs are beliefs. However for *technical* reasons, the two sets of beliefs, DB and SAF, may be managed in two different ways. In particular, for query evaluation, a Relational DBMS cannot use rules, and DB is restricted to a set of atomic facts, and for a Deductive Relational DBMS of DATALOG type, DB is restricted to be a set of Horn clauses. In these cases only part of SAF is included in DB for query evaluation, and the overall content of SAF is used for Integrity Constraint checking. In the case where the system has the same deductive capabilities as a general theorem prover, SAF is included in DB.

In this new approach, the same example, in the context of a Relational DBMS, is represented by:

$SAF = \{ \forall x(p(x) \rightarrow m(x) \vee w(x)), \forall x(m(x) \wedge w(x) \rightarrow), \forall x(m(x) \rightarrow p(x)) \}$

$DB = \{ p(a), m(b), w(b), m(c) \}$

In the context of DATALOG it is represented by:

$SAF = \{ \forall x(p(x) \rightarrow m(x) \vee w(x)), \forall x(m(x) \wedge w(x) \rightarrow), \forall x(m(x) \rightarrow p(x)) \}$

$DB = \{ p(a), m(b), w(b), m(c), \forall x(m(x) \rightarrow p(x)) \}$

According to Reiter's view Integrity Constraints are statements about DB, but, to be more specific, we propose to make explicit the properties of DB content we want to enforce. Then IC defines the part of DB content for which properties of Validity or Completeness have to be enforced . For instance, with the same example, we may have:

$IC = \{ Val(p(x)), Val(m(x)), Val(w(x)), Comp(m(x)), Comp(w(x)) \}$

The intuitive meaning of IC is that, in any state of W and DB, all the sentences of the form: p(x), m(x), and w(x), should be Valid, and that all the sentences of the form: m(x), and w(x), should be Complete,

The following properties give a method for checking Validity or Completeness.

Property 3: Validity checking

From $DBB \vdash Bp$ and $SAF \vdash \neg p$, we can infer $\neg Val(p)$.

Property 4: Completeness checking

From $DBB \vdash \neg Bp$ and $SAF \vdash p$, we can infer $\neg Comp(p)$.

The Properties 3 and 4 allow one to check whether or not properties of the form Val(p) or Comp(p) in IC are violated, depending on the content of SAF and DB.

Another formulation of Property 4 might be to have the premiss: $DB \nvdash p$, instead of: $DBB \vdash \neg Bp$. This formulation shows that, in the standard view of

IC, a violation of a constraint about Completeness would not lead to a formal inconsistency, as is the case in the proposed view.

Property 5: Validity and Completeness checking.

From DBB $\vdash Bp \wedge \neg Bq$ and SAF $\vdash p \rightarrow q$ we can infer $\neg Val(p) \vee \neg Comp(q)$.

The Property 5 can be easily extended to the case where p is the conjunction of several sentences, and q is the disjunction of several sentences.

Property 6: General Validity and Completeness checking.

From DBB $\vdash B(p_1 \wedge \ldots \wedge p_m) \wedge \neg B(q_1 \vee \ldots \vee q_n)$, and:
SAF $\vdash p_1 \wedge \ldots \wedge p_m \rightarrow q_1 \vee \ldots \vee q_n$ we can infer:
$\neg Val(p_1) \vee \ldots \vee \neg Val(p_m) \vee \neg Comp(q_1) \vee \ldots \vee \neg Comp(q_n)$.

One may note that Properties 3, 4 and 5 are particular cases of Property 6 where n and m take the value 0 or 1.

In the case where SAF is included in DB, Properties 3 to 6 are of no interest because the premisses of the properties are never satisfied. For example, for Property 3, if SAF is included in DB, from the premiss SAF $\vdash \neg p$ we infer DB $\vdash \neg p$, and from DBB $\vdash Bp$, we infer DB $\vdash p$. That means that DB would be inconsistent. In the case of Property 5, if SAF is included in DB, from the premiss SAF $\vdash p \rightarrow q$ we infer DB $\vdash p \rightarrow q$ and DBB $\vdash B(p \rightarrow q)$. From the first premiss we have DBB $\vdash Bp$, and then we have DBB $\vdash Bq$, which is inconsistent with DBB $\vdash \neg Bq$.

However, even if SAF is included in DB, the following Property 7 can be used to check Validity or Completeness.

Property 7: General Validity and Completeness checking.

From DBB $\vdash B(p_1 \wedge \ldots \wedge p_m) \wedge \neg B(q_1) \wedge \ldots \wedge \neg B(q_n)$, and:
SAF $\vdash p_1 \wedge \ldots \wedge p_m \rightarrow q_1 \vee \ldots \vee q_n$ we can infer:
$\neg Val(p_1) \vee \ldots \vee \neg Val(p_m) \vee \neg Comp(q_1) \vee \ldots \vee \neg Comp(q_n)$.

If, from a given W and DB, we can infer some sentence about Val and Comp which is inconsistent with IC, this DB state is not acceptable, and we know the property that is violated.

For instance, in the same example (above page 11), from the definition of

SAF it follows immediately:

$$SAF \vdash p(a) \rightarrow m(a) \vee w(a)$$

and from DBB we can infer:

$DBB \vdash Bp(a) \wedge \neg B(m(a) \vee w(a))$, because we have: $DB \vdash p(a)$, and:$DB \nvdash m(a) \vee w(a)$.

Then, from Property 5 we have: $Inval(p(a)) \vee Incomp(m(a) \vee w(a))$; using (C.3) we derive: $Inval(p(a)) \vee Incomp(m(a)) \vee Incomp(w(a))$, and this is equivalent to: $\neg(Val(p(a)) \wedge Comp(m(a)) \wedge Comp(w(a)))$, which is inconsistent with IC, where we have: $Val(p(x)) \wedge Comp(m(x)) \wedge Comp(w(x))$. Here, rejection of the DB state can be interpreted as: either p(a) is not true in the world, or one of the two sentences m(a) or w(a) is missing in DB.

It is interesting to note that in Reiter's approach the inconsistency with IC1 does not suggest that the violation may be due to the fact p(a) is not Valid.

6 Violation of Integrity Constraints

We return to the above example in which:

$$SAF = \{ \forall x(p(x) \rightarrow m(x) \vee w(x)), \forall x(m(x) \wedge w(x) \rightarrow), \forall x(m(x) \rightarrow p(x)) \}$$

and

$$IC = \{ Val(p(x)), Val(m(x)), Val(w(x)), Comp(m(x)), Comp(w(x)) \}$$

We have made explicit here the constraints about the Validity of p(x), m(x) and w(x). However, for any application it should be implicit that for any sentence p the constraint Val(p) is in IC. Indeed, we cannot imagine a situation where the system knows that p is false ($SAF \vdash \neg p$), and it also believes p ($DB \vdash p$). Moreover we should have Val(p) in IC for any sentence, not only for atomic facts, because, for example, the constraints Val(m(x)) and Val(w(x)) do not entail $Val(m(x) \vee w(x))$ (see V.4).

Suppose that initially the DB contains no information about the specific individual a (for instance we can consider that DB is empty), and consider a class of situations in which, were the formula p(a) to be inserted into DB, one

or more ICs would be violated. We distinguish four members of this class:

(i) $W \vdash p(a)$ and $p(a) \notin SAF$.
(ii) $W \vdash \neg p(a)$ and $p(a) \notin SAF$.
(iii) $W \vdash p(a)$ and $p(a) \in SAF$.
(iv) $W \vdash \neg p(a)$ and $\neg p(a) \in SAF$.

Consider first (i). From SAF it follows that $W \vdash p(a) \rightarrow m(a) \vee w(a)$, and thus $W \vdash m(a) \vee w(a)$ (since, by (i), $W \vdash p(a)$). Since W is a complete theory it follows that either $W \vdash m(a)$ holds, or $W \vdash w(a)$ holds.

Since $DB \nvdash m(a)$ and $DB \nvdash w(a)$, it follows that $DBB \vdash \neg Bm(a)$ and $DBB \vdash \neg Bw(a)$. Thus $W, DBB \vdash m(a) \wedge \neg Bm(a)$ or $W, DBB \vdash w(a) \wedge \neg Bw(a)$. Hence, either $\neg Comp(m(a))$ or $\neg Comp(w(a))$. So one of the last two members of the set IC must be violated.

Note that it follows directly from SAF and the definition of Comp, that either Comp(m(a)) or Comp(w(a)), simply because SAF requires that $\neg m(a)$ or $\neg w(a)$ is true, since $\forall x(\neg m(x) \vee \neg w(x))$ is in SAF, and what follows from SAF follows from W, and thus from W,DBB. In general from SAF=$\{\neg p\}$ we can infer Comp(W,DB,p) whatever is in DB. There is nothing odd here if we keep in mind that Comp(W,DB,p) requires something about DB **only** in those situations where $W \vdash p$ holds.

Compare this outcome with what happens when the formula m(a), rather than p(a), is inserted into the intially empty DB (and suppose that $W \vdash m(a)$).

Since $SAF \vdash \neg(m(a) \wedge w(a))$, it follows that $W \vdash \neg w(a)$. Hence, since $DBB \vdash Bm(a) \wedge \neg Bw(a)$, we have $W, DBB \vdash Bm(a) \wedge m(a) \wedge \neg Bw(a) \wedge \neg w(a)$. So that, now, neither Comp(m(x)) nor Comp(w(x)) is violated.

If, however, Comp(p(x)) were to be added to IC, we would again have a violation. Since $SAF \vdash m(a) \rightarrow p(a)$ and $W \vdash m(a)$ it follows that $W \vdash p(a)$. But $DBB \vdash \neg Bp(a)$. Hence $W, DBB \vdash p(a) \wedge \neg Bp(a)$, i.e. $\neg Comp(p(a))$.

However, though the constraint Comp(p(x)) is violated, only an agent who knows what W is can know that there is a violation. An agent who only knows SAF and DB (for instance the DB management system) cannot know that $\neg Comp(p(a))$, even if $\neg Comp(p(a))$ holds. That means that to detect all the violations one has to know what W is. This is because the notions of Completeness and Validity refer to the links between W and DB, and usually one has only a partial knowledge of what W is. It is the role of SAF to represent this partial knowledge, and the distinction between W and SAF, in the formal-

ization, allows to distinguish what is true of the world, and what is known by the system to be true of the world.

Return now to our original scenario, where p(a) is to be inserted into an initially empty DB, and consider case (ii), where $W \vdash \neg p(a)$. Since $DBB \vdash Bp(a)$, we have $W, DBB \vdash Bp(a) \wedge \neg p(a)$ and this means $\neg Val(p(a))$, i.e. there is a violation of the first member of IC. Note that like for (i) this violation is not known by the system. None of the other members of IC is violated, however.

We move on next to case (iii), where $p(a) \in SAF$. Since $W \vdash p(a)$, the argument applying to (i) also applies here, and we see that one of the last two members of IC must be violated. The difference between (i) and (iii) is that in (iii) the violation is known by the system, because it can be derived from SAF and DB. A practical consequence is that in (iii) there is a choice for the system to enforce or not insertion of p(a), while in (i) the insertion is always performed because the system cannot detect the violation.

Finally, as regards cases of type (iv), the outcome is as it was for (ii): the first member of IC, but none of the other can be shown to be violated, but the difference between (ii) an (iv), is that in (iv) the system can detect the violation.

If we now change the basic assumptions, and add the supposition that $SAF \subseteq DB$, we may ask whether this will have any consequences for cases (iii) and (iv). (It clearly will not for (i) and (ii)). As regards case (iii), we still get violation of the last two members of IC (and no others), so the situation is unchanged with respect to violation. It is, however, worth noting that the supposition that $SAF \subseteq DB$ yields the following:
$DBB \vdash B(p(a) \rightarrow m(a) \vee w(a)) \wedge Bp(a) \wedge B(\neg m(a) \vee \neg w(a))$.

Hence (from KD45) $DBB \vdash B(m(a) \vee w(a)) \wedge B(\neg m(a) \vee \neg w(a))$. This is perfectly compatible with the fact that we also have: $DBB \vdash \neg Bm(a) \wedge \neg Bw(a)$. The situation is that the DB is aware that a is either a man or a woman, is aware that a is not both a man and a woman, but yet is not aware of which sex a is. Since in fact a is either m or w (because a is a person), the DB violates ICs requirement of completeness vis -à-vis sentences "m(a)", "w(a)".

As regards situations of type (iv), the consequence of accepting $SAF \subseteq DB$ will simply be that the database contains $\neg p(a)$, and thus the insertion of p(a) would produce an inconsistent database.

Assumptions and conclusions corresponding to the four cases we have analysed are summarised below.

For every cases:
IC = { Val(p(x)), Val(m(x)), Val(w(x)), Comp(m(x)), Comp(w(x)) }.

- **Case (i):**
  - $W \vdash p(a)$.
  - DB={p(a)}
  - SAF = { $\forall x(p(x) \rightarrow m(x) \vee w(x))$, $\forall x(m(x) \wedge w(x) \rightarrow)$, $\forall x(m(x) \rightarrow p(x))$ }.
  - Violation of Comp(m(a)) or Comp(w(a)).
  - The violation cannot be detected by the system.
- **Case (ii):**
  - $W \vdash \neg p(a)$.
  - DB={p(a)}
  - SAF = { $\forall x(p(x) \rightarrow m(x) \vee w(x))$, $\forall x(m(x) \wedge w(x) \rightarrow)$, $\forall x(m(x) \rightarrow p(x))$ }.
  - Violation of Val(p(a)).
  - The violation cannot be detected by the system.
- **Case (iii):**
  - $W \vdash p(a)$.
  - DB={p(a)}
  - SAF = { p(a), $\forall x(p(x) \rightarrow m(x) \vee w(x))$, $\forall x(m(x) \wedge w(x) \rightarrow)$, $\forall x(m(x) \rightarrow p(x))$ }.
  - Violation of Comp(m(a)) or Comp(w(a)).
  - The violation can be detected by the system.
  - The system has the choice to enforce or not the insertion of p(a).
- **Case (iv):**
  - $W \vdash \neg p(a)$.
  - DB={p(a)}
  - SAF = { $\neg p(a)$, $\forall x(p(x) \rightarrow m(x) \vee w(x))$, $\forall x(m(x) \wedge w(x) \rightarrow)$, $\forall x(m(x) \rightarrow p(x))$ }.
  - Violation of Val(p(a)).
  - The violation can be detected by the system.
  - The insertion of p(a) has to be rejected.

7 Conclusion

We have presented a new approach to Integrity Constraints that allows explicit representation of the properties of Validity and Completeness that have to be enforced for some parts of the Database. These properties are formally defined in the framework of a Doxastic Logic, and we have presented formal results that allow IC violations to be interpreted as violations of Validity, or as violations of Completeness. The first kind of violation indicates the false sentences that have to be removed, and the second indicates the missing sentences that have to be inserted.

We have shown that the explicit distinction between beliefs that are guaranteed to be true, represented by SAF, and other beliefs that are not guaranteed to be true, represented by DB, allows a system that knows both SAF and DB to detect some Integrity Constraint violations. The distinction between information that is true of the world, represented by W, and SAF, allows one to distinguish between, on the one hand, DB beliefs that are also true of the world, formally represented by DBB $\vdash$ Bp and W $\vdash$ p, and, on the other hand, DB beliefs that are known by the system to be true of the world, formally represented by DBB $\vdash$ Bp and SAF $\vdash$ p.

However one might wish for a more precise definition of the epistemic status of the SAF content. We think that an answer to this question can be found in our work on safe information [?]. In this work a safe sentence p is, by definition, a sentence such that a particular agent, called the administrator, knows that if p has been inserted in DB by a reliable agent, then p is true of the world.In future work we intend to follow up this line of investigation.

In section 6, we have also shown that in some circumstances the system may know that an insertion leads to an Integrity Constraint violation, and it has the choice, in the case of Completeness violation, as to whether or not to enforce the insertion. The possibility that the system may have a choice of response to IC violation calls for a distinction to be made between "hard" ICs, violation of which will never be allowed by the system, and "soft" ICs, for which the system might tolerate violation.For example, in a particular application, it might be sensible to define a constraint on completeness regarding specification of level of salary (for a given class of agents) as a hard constraint, and a constraint on completeness about telephone numbers (of the same agents) as a soft constraint. A soft constraint would then be a constraint that it would be desirable for the system to satisfy, under ideal circumstances. From the practical point of view, the notion of soft constraint would be likely to be of value only if the system was required to react in specific ways to instances of violation. There is a potential

role here for the application of deontic logic, in characterising more precisely the concept of soft IC. Deontic logic is particularly concerned with the representation of reasoning about the distinction between actual and ideal situations, and thus also with "reparational" norms which come into force when actual circumstances deviate from the ideal. We intend to pursue these matters too in the next stages of this research. The present paper, then, has laid out the groundwork, by giving a framework in terms of which ICs and their violation may be described.

Acknowledgement: This work has been partially suported by the CEC, in the context of the Basic Research Action, MEDLAR.

Appendix

Lemma 1: DBB $\nvdash \neg Bp$ iff DBB $\vdash Bp$.

Proof: From the definition of DBB we have:

$$DBB \vdash \neg Bp \quad \text{iff} \quad DB \nvdash p$$

If we take the negation of both propositions we get:

$$DBB \nvdash \neg Bp \quad \text{iff} \quad DB \vdash p$$

Again from the definition of DBB we have:

$$DBB \vdash Bp \quad \text{iff} \quad DB \vdash p$$

Therefore we have: DBB $\nvdash \neg Bp$ iff DBB $\vdash Bp$.

Lemma 2: W $\cup$ DBB is consistent.

Proof: a sentence p is, or is not, derivable from DB. Therefore either Bp or $\neg Bp$ is in DBB, but DBB cannot contain both of them. Then DBB is consistent, and, from properties of the logic KD45, there exists a Kripke structure M, and a world ω in M, such that any formula of DBB is true in ω (this is denoted by $M, \omega \models$ DBB), and ω is not accessible from itself.

Let's change the assignment of truth values of propositional variables in ω, and only in ω, in such a way that W is true in ω. This is possible because W is consistent, and the theory W does not contain formulas with the modal operator B, except tautologies of KD45. That means that the truth value of any formula in W depends only on ω.

Let's call M' this new Kripke structure. By definition of M' we have $M', \omega \models W$, and since the assignment of truth values of propositional variables in other worlds than ω is unchanged, we have: $M', \omega \models DBB$. This shows that $W \cup DB$ is satisfiable, and therefore it is consistent.

Property 1: Inval(W,DB,p) iff not Val(W,DB,p).

Proof: from the definitions of Validity and Invalidity we have:

Inval(W,DB,p) = (1) $W, DBB \vdash Bp \wedge \neg p$.
not Val(W,DB,p) = (2) $W, DBB \nvdash Bp \rightarrow p$.

We first prove that (1) implies (2).

Since $Bp \rightarrow p$ is equivalent to $\neg(Bp \wedge \neg p)$, (2) is equivalent to (3) $W, DBB \nvdash \neg(Bp \wedge \neg p)$. From Lemma 2 $W \cup DBB$ is consistent, therefore (1) implies (3), and then (1) implies (2).

We now prove that (2) implies (1).

Since $Bp \rightarrow p$ is equivalent to $\neg Bp \vee p$, (2) implies (4) $W, DBB \nvdash \neg Bp \vee p$, and (4) implies (5) $W, DBB \nvdash \neg Bp$ and (6) $W, DBB \nvdash p$.

Due to monotonicity of our consequence relation, $DBB \vdash \neg Bp$ implies $W, DBB \vdash \neg Bp$, then, by contraposition, (5) implies (7) $DBB \nvdash \neg Bp$, and, from Lemma 1, (7) implies (8) $DBB \vdash Bp$, and, due to monotonicity, (8) implies (9) $W, DBB \vdash Bp$.

Due to monotonicity we also have (6) implies (10) $W \nvdash p$, and (10) implies (11) $W \vdash \neg p$, because W is a complete theory. From monotonicity (11) implies (12) $W, DBB \vdash \neg p$, and from (9) and (12) we have (1).

Property 2: Incomp(W,DB,p) iff not Comp(W,DB,p).

Proof: The proof is quite similar to the proof of Property 1. We have:

Incomp(W,DB,p) = (1) $W, DBB \vdash p \wedge \neg Bp$.
not Comp(W,DB,p) = (2) $W, DBB \nvdash p \rightarrow Bp$.

We first prove that (1) implies (2).

(2) is equivalent to (3) $W, DBB \nvdash \neg(p \wedge \neg Bp)$. From Lemma 2, (1) implies (3), and then (1) implies (2).

We now prove that (2) implies (1).

We have (2) implies (4) $W, DBB \nvdash \neg p$ and (5) $W, DBB \nvdash Bp$. Since W is complete, (4) implies (6) $W, DBB \vdash p$, and from Lemma 1, (5) implies (7) $W, DBB \vdash \neg Bp$. Then, (2) directly follows from (6) and (7).

Property 3: $DBB \vdash Bp$ and $SAF \vdash \neg p$ implies $\neg Val(p)$.

Proof: from the property of SAF we have: $W \vdash SAF$, then $SAF \vdash \neg p$ implies $W \vdash \neg p$, and $W, DBB \vdash \neg p$.

Since we also have $DBB \vdash Bp$, we can infer $W, DBB \vdash (Bp) \wedge \neg p$, that is Inval(p), and from Property 1 we have $\neg Val(p)$.

Property 4: $DBB \vdash \neg Bp$ and $SAF \vdash p$ implies $\neg Comp(p)$.

Proof: the proof is very similiar to the proof of Property 3.

From $SAF \vdash p$ we infer $W \vdash p$, and $W, DBB \vdash p$. Then we have $W, DBB \vdash p \wedge \neg Bp$, which means Incomp(p). Then we have $\neg Comp(p)$.

Property 5: $DBB \vdash Bp \wedge \neg Bq$ and $SAF \vdash p \rightarrow q$ implies $\neg Val(p) \vee \neg Comp(q)$.

Proof: from $SAF \vdash p \rightarrow q$ we infer $W \vdash p \rightarrow q$, and $W \vdash \neg p \vee q$. Since W is a complete theory we have (1) $W \vdash \neg p$ or (2) $W \vdash q$.

From $DBB \vdash Bp \wedge \neg Bq$ we have (3) $DBB \vdash Bp$ and (4) $DBB \vdash \neg Bq$. From (1) and (3) we infer $\neg Val(p)$, and from (2) and (4) we infer $\neg Comp(q)$. Therefore we have $\neg Val(p) \vee \neg Comp(q)$.

Property 6: $DBB \vdash B(p_1 \wedge \ldots \wedge p_m) \wedge \neg B(q_1 \vee \ldots \vee q_n)$ and $SAF \vdash (p_1 \wedge \ldots p_m) \rightarrow (q_1 \vee \ldots \vee q_n)$ implies $\neg Val(p_1) \vee \ldots \vee \neg Val(p_m) \vee \neg Comp(q_1) \vee \ldots \vee \neg Comp(q_n)$.

Proof: let's call p the sentence $p_1 \wedge \ldots \wedge p_m$, and q the sentence $q_1 \vee \ldots \vee q_n$. From Property 5 we have $\neg Val(p) \vee \neg Comp(q)$. From Property V.3 we have $\neg Val(p)$ implies $\neg Val(p_1) \vee \ldots \vee \neg Val(p_m)$, and from Property C.3 we have $\neg Comp(q)$ implies $\neg Comp(q_1) \vee \ldots \vee \neg Comp(q_n)$. Therefore we have $\neg Val(p_1) \vee \ldots \vee \neg Val(p_m) \vee \neg Comp(q_1) \vee \ldots \vee \neg Comp(q_n)$.

Property 7: $DBB \vdash B(p_1 \wedge \ldots \wedge p_m) \wedge \neg B(q_1) \wedge \ldots \wedge \neg B(q_n)$ and $SAF \vdash (p_1 \wedge \ldots p_m) \rightarrow (q_1 \vee \ldots \vee q_n)$ implies $\neg Val(p_1) \vee \ldots \vee \neg Val(p_m) \vee \neg Comp(q_1) \vee \ldots \vee \neg Comp(q_n)$.

Proof: from $DBB \vdash B(p_1 \wedge \ldots \wedge p_m) \wedge \neg B(q_1) \wedge \ldots \wedge \neg B(q_n)$ we have for i in [1,m] $DBB \vdash B(p_i)$, and for j in [1,n] $DBB \vdash \neg B(q_j)$.

From $SAF \vdash (p_1 \wedge \ldots p_m) \rightarrow (q_1 \vee \ldots \vee q_n)$ we have: $W \vdash \neg p_1$ or ... or $W \vdash \neg p_m$ or $W \vdash q_1$ or ... or $W \vdash q_n$.

Then the proof follows as for Property 5.

Property V.1: $Val(p) \wedge Val(q) \Rightarrow Val(p \wedge q)$.

Proof : let's call C the theory $W \cup DBB$.

From the definition of Validity we have:

$Val(p) = (1)\ C \vdash Bp \rightarrow p$
$Val(q) = (2)\ C \vdash Bq \rightarrow q$.

We have to prove that, in the context of C, (1), (2) and $B(p \wedge q)$ implies $p \wedge q$. Since $B(p \wedge q)$ implies Bp, from (1) we have p. In the same way $B(p \wedge q)$ implies Bq, and from (2) we have q. Therefore we have $p \wedge q$.

Property V.2: $Val(p \vee q) \Rightarrow Val(p) \vee Val(q)$.

Proof: by contraposition V.2 is equivalent to: $\neg Val(p) \wedge \neg Val(q) \Rightarrow \neg Val(p \vee q)$.

From the antecedent and from the definition of Invalidity and Property 1 we have (1) $C \vdash Bp \wedge \neg p$ and (2) $C \vdash Bq \wedge \neg q$. Then (1) and (2) implies (3) $C \vdash Bp \wedge Bq \wedge \neg p \wedge \neg q$, and (3) implies (4) $C \vdash B(p \vee q) \wedge \neg(p \vee q)$, which is $Invalid(p \vee q)$. Again, from Property 1, we have $\neg Val(p \vee q)$.

Property V.3: $\text{Inval}(p \wedge q) \Rightarrow \text{Inval}(p) \vee \text{Inval}(q)$.

Proof: by contraposition V.3 is equivalent to : $\neg\text{Inval}(p) \wedge \neg\text{Inval}(q) \Rightarrow \neg\text{Inval}(p \wedge q)$.

From Property 1, this is equivalent to: $\text{Val}(p) \wedge \text{Val}(q) \Rightarrow \text{Val}(p \wedge q)$, which is the Property V.1.

Property V.4: $\text{Val}(p) \wedge \text{Val}(q) \not\Rightarrow \text{Val}(p \vee q)$.

Proof: we exhibit a counterexample. Let's consider W and DB defined by: $W = \{\neg p, \neg q\}$ $DB = \{p \vee q\}$.

In that case we have $DBB \vdash \neg Bp$. Then we have $DBB \vdash \neg Bp \vee p$ and $DBB \vdash Bp \rightarrow p$. Then $\text{Val}(p)$ holds. In the same way we can show that $\text{Val}(q)$ holds.

From the definition of DB we have $DBB \vdash B(p \vee q)$; but $W \vdash \neg p \wedge \neg q$, so we have $W, DBB \vdash B(p \vee q) \wedge \neg(p \vee q)$. This shows that $\text{Inval}(p \vee q)$ holds, and, from Property 1, $\text{Val}(p \vee q)$ does not hold.

Property V.5: $\text{Val}(p) \not\Rightarrow \text{Val}(\neg p)$.

Proof: we can easily show that the following situation is a counterexample: $W = \{p\}$ $DB = \{\neg p\}$. Indeed we have $DBB \vdash B\neg p$, $DBB \vdash \neg Bp$, and $W \vdash p$.

Property V.6: $\text{Val}(p \vee q) \not\Rightarrow \text{Val}(p)$.

Proof: we can easily show that the following situation is a counterexample: $W = \{\neg p, q\}$ $DB = \{p\}$. Indeed we have $DBB \vdash Bp$, $DBB \vdash B(p \vee q)$, $W \vdash \neg p$, and $W \vdash p \vee q$.

Property V.7: $\text{Val}(p \wedge q) \not\Rightarrow \text{Val}(p)$.

Proof: we can easily show that the following situation is a counterexample: $W = \{\neg p, q\}$ $DB = \{p\}$. Indeed we have $DBB \vdash Bp$, $DBB \vdash \neg B(p \wedge q)$, and $W \vdash \neg p$.

Property C.1: $\text{Comp}(p) \wedge \text{Comp}(q) \Rightarrow \text{Comp}(p \wedge q)$.

Proof: the proof is very similar to that for Property V.1. From the definition

of Completeness we have:

$\text{Comp}(p) = (1)\ C \vdash p \rightarrow Bp$
$\text{Comp}(q) = (2)\ C \vdash q \rightarrow Bq$
$\text{Comp}(p \wedge q) = (3)\ C \vdash (p \wedge q) \rightarrow B(p \wedge q)$.

If, in the context of C, we assume $p \wedge q$, from (1) we have Bp, and from (2) we have Bq. Then we have $B(p \wedge q)$.

Property C.2: $\text{Comp}(p) \wedge \text{Comp}(q) \Rightarrow \text{Comp}(p \vee q)$.

Proof: from the definition of Completeness we have:

$\text{Comp}(p) = (1)\ C \vdash p \rightarrow Bp$
$\text{Comp}(q) = (2)\ C \vdash q \rightarrow Bq$
$\text{Comp}(p \vee q) = (3)\ C \vdash (p \vee q) \rightarrow B(p \vee q)$.

If, in the context of C, we assume $p \vee q$, since W is a complete theory, we have either $W \vdash p$ or $W \vdash q$; then we have either $C \vdash p$ or $C \vdash q$, and from (1) and (2) we have either Bp or Bq. Since Bp implies $B(p \vee q)$, and Bq also implies $B(p \vee q)$, we have $B(p \vee q)$.

Comment: the basic reason why for Completeness we don't have the same property as for Validity (see V.4) is that from the fact that $p \vee q$ holds in the world, represented by W, we can infer that either p or q holds in W, while from the fact that $p \vee q$ is a DB belief, we cannot infer that either p or q are DB beliefs; so there is no perfect duality between Completeness and Validity.

Property C.3: $\text{Incomp}(p \vee q) \Rightarrow \text{Incomp}(p) \vee \text{Incomp}(q)$.

Proof: this is direct consequence of Property 1 and of the contrapositive form of C.2.

Property C.4: $\text{Incomp}(p \wedge q) \Rightarrow \text{Incomp}(p) \vee \text{Incomp}(q)$.

Proof: this is direct consequence of Property 1 and of the contrapositive form of C.1.

Property C.5: $\text{Comp}(p \wedge q) \not\Rightarrow \text{Comp}(p)$.

Proof: we can easily show that the following situation is a counterexample: $W = \{p, \neg q\}$ $DB = \emptyset$. Indeed we have $DBB \vdash B\neg p$, $W \vdash p$, and

$W \vdash \neg(p \wedge q)$.

Property C.6: $\mathrm{Comp}(p) \not\Rightarrow \mathrm{Comp}(\neg p)$.

Proof: we can easily show that the following situation is a counterexample: $W = \{\neg p\}$ $DB = \emptyset$. Indeed we have $DBB \vdash \neg B \neg p$ and $W \vdash \neg p$.

Property C.7: $\mathrm{Comp}(p \vee q) \not\Rightarrow \mathrm{Comp}(p)$.

Proof: we can easily show that the following situation is a counterexample: $W = \{p, \neg q\}$ $DB = \{p \vee q\}$. Indeed we have $DBB \vdash B(p \vee q)$, $DBB \vdash \neg Bp$, $W \vdash p$, and $W \vdash p \vee q$.

Property C.8: $\mathrm{Comp}(p \vee q) \not\Rightarrow \mathrm{Comp}(p) \vee \mathrm{Comp}(q)$.

Proof: we can easily show that the following situation is a counterexample: $W = \{p, q\}$ $DB = \{p \vee q\}$. Indeed we have $\mathrm{Comp}(p \vee q)$, because we have $W \vdash p \vee q$ and $DBB \vdash B(p \vee q)$, and we have neither $\mathrm{Comp}(p)$, because we have $W \vdash p$ and $DBB \vdash \neg Bp$, nor $\mathrm{Comp}(q)$, because we have $W \vdash q$ and $DBB \vdash \neg Bq$.

Property VC.1: If DB is consistent: $\mathrm{Comp}(\neg p) \Rightarrow \mathrm{Val}(p)$.

Proof: we prove the contrapositive form of the proposition: $\neg\mathrm{Val}(p) \Rightarrow \neg\mathrm{Comp}(\neg p)$.

From Validity definition and Property 1, we have:
$\neg\mathrm{Val}(p) = W, DBB \vdash (Bp) \wedge \neg p$, and $W, DBB \vdash Bp$. We know from Lemma 2 that $W \cup DBB$ is consistent, then we have $W, DBB \nvdash \neg Bp$. Due to monotonicity property of our logic we have $DBB \nvdash \neg Bp$, and from Lemma 1 we have $DBB \vdash Bp$. From the assumption that DB is consistent we have: $DBB \vdash \neg B \neg p$. Therefore we have: $W, DBB \vdash \neg p \wedge \neg B(\neg p)$, that is $\neg\mathrm{Comp}(\neg p)$.

Property VC.2: If DB is complete: $\mathrm{Val}(p) \Rightarrow \mathrm{Comp}(\neg p)$.

Proof: the proof is similar to that proof of Property VC.1. We consider the contrapositive form of the proposition: $\neg\mathrm{Comp}(\neg p) \Rightarrow \neg\mathrm{Val}(p)$.

From the definition of Incomp we have: $W, DBB \vdash \neg p \wedge \neg B(\neg p)$. From Lemma 2 we know that $W \cup DBB$ is consistent, then we have $W, DBB \nvdash B(\neg p)$, and from monotonicity $DBB \nvdash B(\neg p)$. From the assumption that DB is complete, we have either $DBB \vdash B(\neg p)$ or $DBB \vdash Bp$, then we have $DBB \vdash Bp$. Finally we have: $W, DBB \vdash (Bp) \wedge \neg p$, that is $\neg\mathrm{Val}(p)$.

References

[1] B. F. Chellas. *Modal Logic: An introduction.* Cambridge University Press, 1988.

[2] R. Demolombe and A. Jones. Deriving answers to safety queries. In R. Demolombe and L. Fariñas del Cerro and T. Imielinski, editor, *Workshop Nonstandard Queries and Answers*, Toulouse, 1991. ONERA-CERT.

[3] R. Kowalski. Logic for data description. In H.Gallaire and J.Minker, editors, *Logic and Data Bases.* Plenum Press, 1978.

[4] H. Levesque. *A formal treatment of incomplete Knowledge Bases.* PhD thesis, University of Toronto, 1981.

[5] J-M. Nicolas and K. Yazdanian. Integrity checking in Deductive Databases. In H. Gallaire and J. Minker, editors, *Logic and Databases.* Plenum, 1982.

[6] A. Olivé. Integrity checking in Deductive Databases. In *17th Int. Conf. on Very Large Data Bases*, 1991.

[7] 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.

[8] R. Reiter. What Should a Database Know? *Journal of Logic Programming*, To appear.

[9] E. Teniente. *El método dels esdeveniments per a l'actualització de vistes en Bases de Dades Deductives.* PhD thesis, Universitat Politécinica de Catalunya, 1992.

[10] E. Teniente and A. Olivé. The Events Method for View Updating in Deductive Databases. In *Int. Conf. on Extending Data Base Technology*, 1992.