Database validity and completeness: another approach and its formalisation in modal logic

R. Demolombe
ONERA/DTIM
2 Avenue E. Belin BP 4025, 31055 Toulouse Cedex, France.
e-mail: Robert.Demolombe@cert.fr

1 Introduction

In database context users are concerned by completenesss of data because they want to have an answer to every questions. Of course, they also want to have a consistent database, because if it is inconsistent there is no doubt that there is at least one piece of information in the database which is false in the world.

In this abstract a database DB will be understood in a very general sense, and it will be assumed that it is represented by a theory in Fisrt Order Logic. In the particular case of a relational database, this theory is represented by a set of ground atomic facts, in the case of a deductive database, the theory contains, in addition, rules represented by definite Horn clauses, but our analysis can be applied as well to more general cases.

To be slightly more formal, we say that DB is **self-complete** [1] (s-complete for short) iff for every sentence f we have either $\vdash DB \rightarrow f$ or $\vdash DB \rightarrow \neg f$, and we say that it is **self-incomplete** iff there exists some sentence f such that $\not\vdash DB \rightarrow f$ and $\not\vdash DB \rightarrow \neg f$.

In the same formalism we say that DB is **consistent** iff there exists no sentence f such that $\vdash DB \rightarrow f$ and $\vdash DB \rightarrow \neg f$, and we say that it is **inconsistent** iff there exists some sentence f such that $\vdash DB \rightarrow f$ and $\vdash DB \rightarrow \neg f$.

A very intuitive reformulation of these definitions is that in an incomplete database there are some missing data, while in an inconsitent one there are too many data. This shows that these properties refer to the database alone and justifies the term "self-completeness" we have introduced.

Another approach, which is presented in this paper, is to consider the links between the database and the world. In many cases it is implicitly assumed that these links are on-to-one links, but in most real application that is just an ideal view, which is far to fit actual situation. Nevertheless, even if these one-to-one links are not guaranteed to hold for every piece of information, it can be assumed that for some of them this property holds. This observation has motivated our approach [Dem96, Mot89], where we distinguish the links from the world to the database, and the links from the database to the world. The formers correspond to the property we call "completeness", and the latters to what we call "validity".

To be a bit more formal, we consider a theory W, which is assumed to be s-complete (in the sense defined at the begining of this section) and to be an exact representation of the world, in the same language as DB. We do not claim that theory W is known, but we can talk about it.

Then, we say that DB is **complete** with regard to W for f iff $\vdash W \rightarrow f$ implies $\vdash DB \rightarrow f$, and we say that DB is **valid** with regard to W for f iff $\vdash DB \rightarrow f$ implies $\vdash W \rightarrow f$.

In the next section we shall recall how s-completeness can be formalised. In section 3, we shall give more formal definition for validity and completeness, we shall see properties of these notions, and we shall sketch how they can be used to qualify answers to queries, or to define constraints.

2 Standard approach: inconsistency and self-completeness

If database DB is not consistent, to make it consistent some facts have to be removed from DB. The big issue is to minimise DB changes. There is a huge number of techniques presented in the literature to do that (see for instance [Win90]), and we are not analysing this issue here.

If DB is incomplete the standard method to make it complete is to accept Reiter's Closed World Assumption (CWA) [Rei83]. The intuitive idea is that facts that are not represented in the database are false in the world. For instance, if we have $DB = \{p(a), p(b)\}$, we accept the Completion Axiom : $\forall x(p(x) \rightarrow x = a \vee x = b)$, which is equivalent to $\forall x(\neg(x = a) \wedge \neg(x = b) \rightarrow \neg p(x))$. The meaning of this axiom is that for every x different of a

[1] We introduce this term to avoid confusions. In particular, in formal logic the term completeness may also refer to completeness of the axiomatics with respect to the semantics.

and of b we can infer $\neg p(x)$.

If we accept CWA, for every f we have $\vdash DB \wedge CWA \rightarrow f$ or $\vdash DB \wedge CWA \rightarrow \neg f$, then DB is s-complete. In some sense we have a "global" s-completeness. In [Rei92] Reiter has proposed a formalisation of what can be called "local" s-completeness. The idea is to characterise s-completeness not for every sentence, but only for sentences of a certain type.

For instance, we might want to know, or to impose, that for every employee in a database either telephone number or fax number or email address is explicitly represented in the database. Notice that the following sentence does not represent what we have in mind:
$\forall x(emp(x) \rightarrow (\exists y\ tel(x,y) \vee \exists z\ fax(x,z) \vee \exists t\ email(x,t)))$.
Indeed, if, for instance, $emp(a)$ is in DB, we can only infer that a has, in the world, a telephone number or a fax number or an email address, but these numbers are not necessarily in DB.

To formally represent what we want to represent we need to distinguish what is true in the world and what is in DB. For that purpose Reiter uses an epistemic logic defined by Levesque: KFOPC. In this logic, sentences of the form $B(f)$, or Bf for short, mean that the database DB "believes" f, or that f is "in" DB. Then, the above property can be expressed by:
$\forall x(Bemp(x) \rightarrow (\exists y\ Btel(x,y) \vee \exists z\ Bfax(x,z) \vee \exists t\ Bemail(x,t)))$.

This sentence means that if $emp(a)$ is in DB there exists a y value such that $tel(a,y)$ is in DB, or there exists a z value such that $fax(a,z)$ is in DB, or there exists a t value such that $email(a,t)$ is in DB.

3 New approach: validity and completeness

We first give more formal definitions of the notions of validity and completeness presented in the introduction [DJ94, Dem96, DJ94]. For these definitions the database content is represented by the set of beliefs DBB such that:
$DBB = \{Bf\ :\ \vdash DB \rightarrow f\} \cup \{\neg Bf\ :\ \not\vdash DB \rightarrow f\}$.
For instance, if we have $DB = \{p(a), p(b)\}$, we have $DBB = \{Bp(a), Bp(b), \neg Bp(c), \neg Bp(d), \ldots\}$. For reasoning about beliefs we use a modal logic of type (KD) very close to KFOPC. Theorems in this logic are denoted by $\vdash_B$.

We say that DB is complete with respect to W for f iff we have
$\vdash_B W \wedge DBB \rightarrow (f \rightarrow Bf)$. Property of completeness is represented by the sentence $f \rightarrow Bf$, and we use the notation $comp(f) \stackrel{\text{def}}{=} f \rightarrow Bf$. This definition is generalised to represent completeness for all the sentences of the form $f(x)$. This is represented by $comp(f(x)) \stackrel{\text{def}}{=} \forall x(f(x) \rightarrow Bf(x))$.

In the same example, if we have $W = \{p(a), \neg p(b), \neg p(c), \neg p(d), \ldots\}$, we can check that we have $\vdash_B W \wedge DBB \rightarrow comp(p(x))$.

Validity is defined in the same way. We say that DB is valid with resepct to W for f iff we have $\vdash_B W \wedge DBB \rightarrow (Bf \rightarrow f)$. For validity we adopt the notation $val(f) \stackrel{\text{def}}{=} Bf \rightarrow f$, and $val(f(x)) \stackrel{\text{def}}{=} \forall x(Bf(x) \rightarrow f(x))$. In the same example we do **not** have $\vdash_B W \wedge DBB \rightarrow val(p(x))$.

Now, we can analyse how validity and completeness are related to consistency and s-completeness. We have the property:
$\vdash_B comp(f(x)) \wedge comp(\neg f(x)) \rightarrow \forall x(Bf(x) \vee B\neg f(x))$.
That is, from completeness for $f(x)$ and for $\neg f(x)$ we can infer that, for every x, we have either $\vdash DB \rightarrow f(x)$ or $\vdash DB \rightarrow \neg f(x)$, which is some kind of local s-completeness for sentences of the form $f(x)$.

We also have the similar property :
$\vdash_B val(f(x)) \wedge val(\neg f(x)) \rightarrow \neg\exists x(Bf(x) \wedge B\neg f(x))$.
That is, from validity for $f(x)$ and $\neg f(x)$ we can infer that there is no x such that $\vdash DB \rightarrow f(x)$ and $\vdash DB \rightarrow \neg f(x)$, which is some kind of local consistency. Morevoer, if DB is consistent we have $\vdash_B comp(\neg f) \rightarrow val(f)$, and, if DB is s-complete we have $\vdash_B val(f) \rightarrow comp(\neg f)$. Reasoning about val and $comp$ is not trivial. For instance, we have neither $\vdash_B val(f \wedge g) \rightarrow val(f)$, nor $\vdash_B comp(f \vee g) \rightarrow comp(f)$.

From these definitions, to know whether a database is valid or complete, we should have to know W. However, we may know, or at least we may assume, that the database is valid or complete for some kinds of sentences. These assumptions may be jusitified by the fact that we trust users who insert corresponding facts in DB [Dem98]. If $val(f(x))$ is assumed, from $Bf(a)$ (i.e. $f(a)$ is "in" DB) we can infer $f(a)$ (i.e. $f(a)$ is true in the world), and, if $comp(f(x))$ is assumed, from $\neg Bf(a)$ we can infer $\neg f(a)$. These assumptions precisely define to what extend DB content informs us about the world.

For instance, it could be assumed that the database is complete for employees and telephone numbers, and valid for engineers' email addresses. In formal terms it is **assumed** that we have:
$\vdash_B W \wedge DBB \rightarrow comp(emp(x)) \wedge comp(tel(x,y)) \wedge val(eng(x) \wedge email(x,y))$.

We have defined, and implemented [Dem97], an automated deduction technique to characterise the subsets of answers that are guaranteed to be valid or complete. In this example, if we have the query : *what are employees' telephone numbers or email addresses?*, i.e. $emp(x) \wedge (tel(x,y) \vee email(x,y))$?, we can infer, from the above assumptions, that the answer is complete for the subset of tuples $< x, y >$ that correspond to employees' telephone numbers, and that it is valid for engineers' email addresses, i.e. we have $comp(emp(x) \wedge tel(x, y))$ and $val(eng(x) \wedge email(x, y))$.

Instead of assuming that the database is valid or complete for some sentences, we might want to impose, as a constraint, that the database **should** be valid or complete for some kinds of sentences. That leads to another approach of what is usually called "integrity constraints". In [DJ96, DJC96] this new view of constraints has been analysed using deontic logic. To check violations of these constraints we need to assume that some parts of DB is valid or complete for other kinds of sentences.

References

[Dem96] R. Demolombe. Validity Queries and Completeness Queries. In *Proc. of 9th International Symposium on Methodologies for Intelligent Systems*, 1996.

[Dem97] R. Demolombe. Answering queries about validity and completeness of data: from modal logic to relational algebra. In T. Andreasen, H. Christiansen, and H. L. Larsen, editors, *Flexible Query Answering Systems*. Kluwer Academic Publishers, 1997.

[Dem98] R. Demolombe. To trust information sources: a proposal for a modal logical framework. In C. Castelfranchi and Y-H. Tan, editor, *Proc. of the Workshop on Deception, Fraud and Trust in Agent Societies*, 1998.

[DJ94] R. Demolombe and A.J.I. Jones. Deriving answers to safety queries. In R. Demolombe and T. Imielinski, editors, *Nonstandard queries and nonstandard answers*. Oxford University Press, 1994.

[DJ96] R. Demolombe and A.J.I. Jones. Integrity Constraints Revisited. *Journal of the Interest Group in Pure and Applied Logics*, 4(3), 1996.

[DJC96] R. Demolombe, A. J. I. Jones, and J. Carmo. Toward a uniform logical representation of different kinds of integrity constraints. In S. Conrad, H-J. Klein, and K-D. Schewe, editors, *Integrity in Databases*. Otto von Guericke Universitat Magdeburg, 1996.

[Mot89] A. Motro. Integrity = validity + completeness. *ACM TODS*, 14(4), 1989.

[Rei83] 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.

[Rei92] R. Reiter. What Should a Database Know? *Journal of Logic Programming*, 14(2,3), 1992.

[Win90] M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.