Answers that Have Integrity

Hendrik Decker*

Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, Spain

Abstract. Answers to queries in possibly inconsistent databases may not have integrity. We formalize ‘has integrity’ on the basis of a definition of ‘causes’. A cause of an answer is a minimal excerpt of the database that explains why the answer has been given. An answer has integrity if one of its causes does not overlap with any cause of integrity violation.

1 Introduction

We continue the development of ‘answers that have integrity’ (in short, AHI) in databases that may suffer from extant violations of integrity. It has begun in [6], for databases, queries, constraints and answers without negation. In this paper, definitions and results are generalized to be applicable also if there is negation.

Consistent query answering (CQA) [1] is a popular approach to provide useful answers in inconsistent databases. Roughly, CQA defines an answer to be consistent if the answer is true in each minimal repair. Unfortunately, the consistency of answers is not invariant under different notions of minimality.

We elaborate the alternative idea of answers that ‘have integrity’, i.e, answers that are reasonably correct in the presence of integrity violation. This idea is based on ‘causes’, i.e., certain extracts of the database that explain why an answer is given, or why a constraint is violated. Intuitively, an answer has integrity if one of its causes does not overlap with any cause of integrity violation.

Arguably, AHI does not suffer from any ambivalence of minimality, nor from several other shortcomings associated to CQA. However, while computing AHI for definite databases and queries is very simple, it seems to be as complex as computing CQA in general.

Apart from some background of database logic, we broach, in Section 2.1 the only-if halves of predicate completions as a basis for defining causes of negative answers. Section 2.2 contains the main definitions for characterizing causes. In Section 4 we define and discuss how to compute AHI. In Section 5 we compare AHI to related work. In Section 6 we conclude with an outlook to further work.

2 Preliminaries

In 2.1 we address the foundations of the database logic upon which the remainder of the paper is built. In 2.2 we make explicit an implicit part of the database, viz. ground instances of the only-if halves of predicate definitions, since they may contribute to causes for explaining negative answers.

* Supported by FEDER and the Spanish grants TIN2009-14460-C03, TIN2010-17139.
2.1 Background

As a formal framework, we opt for *datalog*. We denote logical consequence by $\models$. We use the abbreviation iff for 'if and only if'.

We assume a universal language denoted by $\mathcal{L}$ that contains a finite universal domain $\mathcal{L}^c$ of constant terms over which each attribute variable in each database ranges. W.l.o.g., we represent the elements in $\mathcal{L}^c$ by natural numbers. By overloading, we use $=$ as the identity predicate in $\mathcal{L}$, as the assignment symbol in substitutions of variables with terms, and as meta-level equality. Since $,,$ is used as the conjunction operator between literals in the body of clauses, we use ';,' as the delimiter between elements of sets of clauses.

As in [21], we call each database without any occurrence of negative literals in the body of its clauses a *definite* database, and each database without self-recursive definitions of predicates a *hierarchical* database. As opposed to definite databases, which may contain self-recursive definitions of predicates, hierarchical database clauses may contain negative literals in their body. Unless mentioned otherwise, each database considered in this paper is assumed to be hierarchical. Answers that have integrity in definite databases are studied in [6].

We say that a formula $F$ is a *conjunctive sentence* if $F$ is a universally closed non-empty conjunction of literals. If all literals in $F$ are positive, we also say that $F$ is a *definite sentence*.

For each database $D$ and each conjunctive sentence $F$, we write $D(F) = \text{true}$ and say ‘$F$ is true in $D$’ if $F$ is a logical consequence of the theory associated to $D$ (e.g., the completion of $D$, or some standard model theory such as the set of stable models of $D$). Otherwise, we write $D(F) = \text{false}$ and say ‘$F$ is false in $D$’.

Let $D$ be a database and $L$ a ground literal such that $D(L) = \text{true}$. We say that $L$ is *terminal* in $D$ if the atom of $L$ does not match the head of any clause with non-empty body in $D$. For instance, $\sim p(1, 2)$ is terminal in $\{ p(x, 1) \leftarrow q(x): q(1) \}$. If the predicates in $\mathcal{L}$ are partitioned, as usual, into extensional and intensional ones, then each extensional fact is terminal.

We assume, w.l.o.g., that each query is a conjunctive query, possibly with negative literals, where the predicates in the query are defined by clauses in the database. Similarly, we assume that each integrity constraint (shortly, constraint) is a *denial*, i.e., a clause with empty head and a conjunction of (possibly negative) literals as its body. Each finite set of constraints is called an *integrity theory*.

We assume each clause $C$ to be *range-restricted*, i.e., each variable in $C$ occurs in a positive literal in the body of $C$.

We do not flatten databases by materializing the definition of predicates or by representing the database by some standard model, since such an unfolding of database clauses with non-empty body may lead to a loss of causal information.

**Example 1.** The two databases $D_1 = \{ p \leftarrow r; \ r; \ s \}$ and $D_2 = \{ p \leftarrow s; \ r; \ s \}$ have the same flattened version: $\{ p; \ r; \ s \}$. That version, however, does not provide to identify that $r$ is part of the cause of the truth of $p$ in $D_1$ and $s$ is not, nor that $s$ is part of the cause of the truth of $p$ in $D_2$ and $r$ is not.