Foundations of Simplified Integrity Checking Reviewed

M. Celma L. Mota

DSIC, Universidad Politécnica de Valencia (Spain), post box 22.012, mati@dsc.upv.es

ABSTRACT

We review some fundamental concepts of simplified integrity checking in deductive databases. This is done on a sufficiently abstract level such that we do not have to depend on any particular method. Our main focus is on concepts of soundness and completeness of simplified integrity checking. These two concepts relate the declarative and procedural concepts of satisfaction. The former is defined by well-established views of integrity satisfaction and the latter by the methods. Also, we distinguish between generation and evaluation phases of integrity checking, and apply the concepts of soundness and completeness to each of the two phases.

1 INTRODUCTION

Integrity checking is an important issue in database systems. An integrity constraint is a statement that a database must satisfy at any time in order to faithfully describe the real world represented by the database. The evolution through time of a database can be described by a sequence of states where transitions from one state to the next are accomplished by database transactions. According to this evolution scheme, static and dynamic constraints can be distinguished; the former restrict the validity of each state on its own, while the latter relate the validity of a sequence of consecutive states. In this paper, we only deal with static constraints, and therefore, integrity checking is reduced to determine, with some method, if a database state satisfies or violates the integrity constraints.

Given a state D and an integrity constraint W, by method we mean a decision procedure such that decides with a binary result (yes/no) whether D satisfies or does not satisfy W.

We rely on the nowadays classical formalization of deductive databases in logic, i.e., a database state is represented by a first order theory Tho. The evolution of the database can be described by a sequence of states related by deductive transactions. Integrity constraints by closed well-formed formulas of the underlying language of that theory. Using this formalization, integrity checking in deductive databases can be stated as follows:

Given:
- a deductive database scheme (L, IC), where
  - L is a first order language with infinitely countable sets of constant, function and predicate symbols, and
  - IC is the set of integrity constraints (closed wff of L).
- a database state D
  \[ D = \{ A \rightarrow B : A\text{ is an atom and } B\text{ is a wff}\}. \]
- a transaction T formed by two sets of clauses
  \[ T_{\text{ins}}: \text{facts and rules to be inserted} \]
  \[ T_{\text{del}}: \text{facts and rules to be deleted} \]
- a database state D', obtained by applying the transaction T to D:
  \[ D' = (D \cup T_{\text{ins}}) \setminus T_{\text{del}} \] (D and D' are states related by T).

Checking: D' satisfies W (for all W ∈ IC).

To ensure that each integrity constraint is satisfied in the new state, all of them must be checked after every transaction.

However, this checking can be very costly if they are evaluated as queries, particularly in large databases. In spite of these difficulties, it is possible to reduce the amount of computation if advantage is taken of the fact that, before the transaction was made, the database was known to satisfy its integrity constraints and hence, any violation of them is due to an update induced by the transaction.

Following Nicolas’s proposal for relational databases [19], many methods for simplified integrity checking in deductive databases have been proposed in the last ten years; all of them are based on the idea of evaluating simplified instances of the integrity constraints generated by the updates induced by the transaction. They differ mainly in the way these induced updates are determined and in the strategy used to instantiate and simplify the constraints.

Unfortunately, the work in this field often suffers from shortcomings in terms of:
- conceptual foundations,
- soundness and completeness results, and
- formal criteria to analyse various methods.

Thus, the aim of this paper is to reassess the foundations of simplified integrity checking independently of the particular strategy used.

Several methods proposed for simplified integrity checking in deductive databases have been presented in the literature; [10], [18], [28], [2], [12], [21], [5] and others. The outline of this paper is as follows: In section 2 we review two common points of view of integrity satisfaction used in the literature. In section 3, two phases of simplified integrity checking are reviewed. In section 4, we define and discuss concepts of soundness and completeness of simplified integrity checking. We finish with some conclusive remarks in section 5.

2 POINTS OF VIEW OF INTEGRITY SATISFACTION

Let (L,IC) be a deductive database scheme and D be a database state represented, as said, by a first order theory Th0.

This theory is, depending on the intended semantics, the following:

a) If the semantics of the completion is assumed [8]:
\[ Th_0 = \text{comp}(D) = D \cup \{ \text{completion axioms of each predicate of } L \} \cup \{ \text{equality theory axioms} \}. \]

The declarative semantics, in this case is defined by the set:
\[ L_0 = L \text{ is a ground literal such that } Th_0 = L_0. \]

b) If the semantics of a distinguished minimal model, M0, of D is assumed:
\[ Th_{M_0} = \text{comp}(Th_0) \cup \text{DCA} \]

where Th_{M_0} = \{ A: A is a ground atom and \forall M_0 \} and DCA is the Domain Closure Axiom [23]. Roughly speaking, this
the declarative semantics, in this case is defined by the set:
\[ \text{ground literal } L : \text{Th}_D \]

Depending on the minimal model considered, it is possible to distinguish the following semantics:
- iterated fixpoint semantics \[1\],
- perfect model semantics \[22\],
- stable model semantics \[13, 29\],
- well-founded model semantics \[30\].

Now we present two different points of view of integrity satisfaction \[28\] independently of the intended semantics. Let \( W \) be an integrity constraint:

\( D \) satisfies \( W \) iff \( \text{Th}_D \models W \).

\( D \) violates \( W \) iff \( \neg (D \text{ satisfies } W) \).

In both cases, constraint violation is defined in terms of constraint satisfaction:

\( D \) satisfies \( W \) iff \( \text{Th}_D \cup \{W\} \) is consistent.

The methods for simplified integrity checking mentioned above can be classified according to the point of view of the integrity satisfaction they consider:
- theoremhood view \(\{18, 12, 21, 10\}\),
- consistency view \(\{2, 28, 5\}\).

When the theory that represents a database state is consistent, the theoremhood view implies the consistency view. Both points of view are equivalent if the theory \( \text{Th}_D \) is categorical (for each closed wff \( W \) either \( \text{Th}_D \models W \) or \( \text{Th}_D \models \neg W \)).

It is important to notice that in the semantics of a minimal model, the theory which represents the state \( D \) is categorical.

Both points of view are problematic because, in general, determining if a closed wff \( W \) is a logical consequence of a theory \( \text{Th}_D \) is an undecidable problem in first order logic. Thus:

- in the theoremhood view \( D \) satisfies \( W \) iff \( \text{Th}_D \models W \),
- in the consistency view \( D \) satisfies \( W \) iff \( \text{Th}_D \models \neg W \),

violation of an integrity constraint \( W \) such that \( \text{Th}_D \models \neg W \) but also \( \text{Th}_D \models \neg W \) cannot be detected, in general.

The problems of incompleteness that we have just identified can be solved if the database and the constraints have syntactic properties which ensure that, for every database state \( D \) the theory \( \text{Th}_D \) representing the state \( D \) is categorical.

More advanced points of view of integrity satisfaction are introduced in \[25\] but discussion of these is beyond the scope of this paper.

3 SIMPLIFIED INTEGRITY CHECKING

Let \( (L, IC) \) be a deductive database scheme where the integrity constraints are represented by closed wff in prenex normal form: \( W = \forall x_1 \forall x_2 \ldots \forall x_n \neg W' \) (where \( x_1, x_2, \ldots, x_n \) are universally quantified variables not governed by an existential quantifier), and let \( D \) and \( D' \) be two deductive database states related by the transaction \( T \) such that \( D \) is known to satisfy any integrity constraint.

Independently of the particular strategy used, most methods simplify the integrity checking according to the following scheme:

PHASE 1: Generation Phase
Step 1: computation of the sets of literals which "capture" the difference between the consequent states \( D \) and \( D' \).
Step 2: identification of relevant constraints.
Step 3: instantiation of relevant constraints.
Step 4: simplification of instances of relevant constraints.

PHASE 2: Evaluation Phase
Step 5: evaluation of the simplified instances of relevant constraints in \( D' \).

In the generation phase, following the ideas of Nicolas [19], simplified instances of the constraints are obtained using the updates induced by the transaction. In the evaluation phase these instances are evaluated in the new state.

Now, we are going to present each step in more detail.

Step 1 consists of computing sets of literals which "capture" the difference between \( D \) and \( D' \). This difference is defined as follows:

\( \text{INS}_{D,D} = \{ L : L \text{ is a ground literal, } \text{Th}_D \models L, \text{Th}_D \models \neg L \} \)
\( \text{DELD}_{D,D} = \{ L : L \text{ is a ground literal, } \text{Th}_D \models L, \text{Th}_D \models \neg L \} \)

where \( \text{Th}_D \) (resp. \( \text{Th}_D \)) is the theory which represents the state \( D \) (resp. \( D' \)) in the intended semantics.

The sets \( \text{INS}_{D,D} \) and \( \text{DELD}_{D,D} \) represent the impact of the transaction. In order to analyse the meaning of the elements of these sets, we define the following subsets:

\( \text{INS}_{D,D}^+ \) (resp. \( \text{DELD}_{D,D}^+ \)) = \( \{ A: A \text{ is a ground atom, } \neg A \in \text{INS}_{D,D} \text{ (resp. } \text{DELD}_{D,D} \} \)
\( \text{INS}_{D,D}^- \) (resp. \( \text{DELD}_{D,D}^- \)) = \( \{ A: A \text{ is a ground atom, } A \in \text{INS}_{D,D} \text{ (resp. } \text{DELD}_{D,D} \} \)

The set \( \text{INS}_{D,D} \cup \text{DELD}_{D,D} \) represents the information added by the transaction and the set \( \text{INS}_{D,D}^+ \cup \text{DELD}_{D,D}^+ \) the information deleted by the transaction.

If \( \text{Th}_D \) is a categorical theory then the following holds:

\( \text{INS}_{D,D} = \text{DELD}_{D,D} \)
\( \text{DELD}_{D,D} = \text{INS}_{D,D} \)

and the difference can be defined without losing information in the following way:

\( \text{INS}_{D,D} = \{ A: A \text{ is a ground atom, } \text{Th}_D \models A \} \) (insertions)
\( \text{DELD}_{D,D} = \{ A: A \text{ is a ground atom, } \text{Th}_D \models \neg A \} \) (deletions)

Because computing this difference can be very costly, some methods define some kind of sets which "capture" this difference.

Step 2 consists of determining, by unification, the integrity constraints which are relevant w.r.t. the elements of the sets obtained in the previous step. An integrity constraint \( W \) is said to be relevant w.r.t. a transaction \( T \) iff there exists an element in \( \text{INS}_{D,D}^+ \cup \text{DELD}_{D,D}^- \) (resp. \( \text{INS}_{D,D}^- \cup \text{DELD}_{D,D}^+ \)) that unifies with an atom occurring negatively (resp. positively) in \( W \).

Step 3 consists of instantiating the relevant constraints by using the substitutions obtained in Step 2. These substitutions belong to the following two sets:

\( \Theta = \{ \theta : \theta \text{ is the restriction to } x_1, x_2, \ldots, x_n \text{ of an mgu of an atom occurring positively in } W \text{ and an atom in } \text{INS}_{D,D} \cup \text{DELD}_{D,D} \} \)
\( \Psi = \{ \psi : \psi \text{ is the restriction to } x_1, x_2, \ldots, x_n \text{ of an mgu of an atom occurring negatively in } W \text{ and an atom in } \text{INS}_{D,D} \cup \text{DELD}_{D,D} \} \)

Then, for every substitution \( \phi \in \Psi \cup \Theta \) there exists an instance of \( W \) defined as \( W_\phi \). This means that for every integrity constraint there can be several instances (one for each substitution in \( \Psi \cup \Theta \)).

Step 4 consists of simplifying the instances of the integrity constraints by applying absorption rules.