Efficient Updates to Independent Schemes in the Weak Instance Model

Paolo Atzeni
Università di Napoli
Dipartimento di Informatica e Sistemistica
Via Claudio, 21
80125 Napoli, Italy
atzeni@irmiasi bitnet

Riccardo Torlone
IASI-CNR
Viale Manzoni, 30
00185 Roma, Italy
torlone@irmiasi bitnet

Abstract

The weak instance model is a framework to consider the relations in a database as a whole, regardless of the way attributes are grouped in the individual relations Queries and updates can be performed involving any set of attributes The management of updates is based on a lattice structure on the set of legal states, and inconsistencies and ambiguities can arise
In the general case, the test for inconsistency and determinism may involve the application of the chase algorithm to the whole database In this paper it is shown how, for the highly significant class of independent schemes, updates can be handled efficiently, considering only the relevant portion of the database

This work was partially supported by Ministero della Pubblica Istruzione, within the Project "Metodi formali e strumenti per basi di dati evolute" and by Consiglio Nazionale delle Ricerche, within "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Obiettivo Logidata+" Correspondence should be addressed to the first author at IASI-CNR, where he holds a cross appointment

1 Introduction

In a relational database, the universe of discourse is represented by means of a set of relations The *weak instance approach* [7,13,17,18,19] provides a framework to consider a database as a whole, regardless of the way attributes appear in the various relation schemes The information content of a database state is considered to be embodied in the *representative instance*, a sort of relation with variables, obtained by extending all relations to the global set of attributes (called the *universe*) and then by *chasing* [16] their union Query answering is performed by first computing the *total projection* [21] of the representative instance on the set of attributes involved in the query, and then executing whatever further operations are needed Any subset of the universe can in principle be queried

Example 1 *Figures 1, 2, and 3, respectively, show a consistent database state, the corresponding representative instance, and a total projection, which refer to a database scheme with the relation schemes* $R_1(EDP)$, $R_2(DM)$, $R_3(PM)$, *and the functional dependencies* $E \rightarrow D$, $D \rightarrow M$ *as constraints*

Recently, following a new interest on the theory of database updates [1], we proposed a formal approach to updates in the weak instance model [9], coherently with the management of queries, which allows the retrieval of tuples over any subset of the universe, insertions and deletions over any subset of the universe are allowed Problems of consistency and determinism arise, and have been completely characterized

Example 2 *If we want to insert in the state in Figure 1 a tuple defined on* EM, *with values* Jim *for* E *and* White *for* M, *we can consistently add a tuple to the second relation, with values* MS *for* D *and* White

Employee	Dept	Project
John	CS	A
John	CS	B
Bob	EE	B
Jim	MS	C

Dept	Manager
CS	Smith
IE	White
EE	Jones

Project	Manager
A	Smith
B	Smith
B	Jones

Figure 1 A database state

Employee	Dept	Project	Manager
John	CS	A	Smith
John	CS	B	Smith
Bob	EE	B	Jones
Jim	MS	C	v_1
v_2	CS	v_3	Smith
v_4	IE	v_5	White
v_6	EE	v_7	Jones
v_8	v_9	A	Smith
v_{10}	v_{11}	B	Smith
v_{12}	v_{13}	B	Jones

Figure 2 A representative instance

Employee	Manager
John	Smith
Bob	Jones

Figure 3 A total projection of the representative instance

for M because of the dependency $D \rightarrow M$, the chase would combine the tuple < Jim, MS, C >, *already in* r_1, *with this tuple, generating a tuple in the representative instance with values* Jim *for E and* White *for M*

If we want to insert into the same state another tuple defined on EM, but with values Dan *for E and* Moore *for M, we obtain a potential result only if we add one tuple to* r_1 *and one tuple to* r_2, *with the same value for D and a value for P, whichever they be (provided that the constraints are not violated) In this case, some further piece of information has to be added, and there are several possible choices, this is a case of nondeterminism*

An inconsistency arises if we try to insert a tuple on EM with John *for E and* White *for M*

In the general case, the characterizations for consistency and determinism of insertions require the construction of the representative instance of the state, by means of the application of the chase procedure to a set of data that involves the whole database In this paper, we restrict our attention to the important class of *independent schemes* [12,14,20,21], for which it is known that query answering can be performed in an efficient way [6,14,22], and show that updates can be implemented efficiently

The paper is organized as follows In Section 2 we briefly review the needed background In Section 3 we review definitions and characterizations about updates in the weak instance model [9] In Section 4 we consider independent schemes, and present characterizations for consistency and determinism, specific for them On the basis of these results, efficient algorithms are shown in Section 5 Finally, in Section 6, we summarize the conclusions Because of space limitations, proofs of theorems are omitted and can be found in [8]

2 Background

For the sake of brevity, we only sketch most usual definitions, which can be found in detail in popular textbooks [15,23], concentrating on the nonstandard ones

2.1 Basics

Throughout the paper, we consider a generic *database scheme* $\mathbf{R} = \{R_1(X_1), \ , R_n(X_n)\}$, with a *universe* of *attributes* $U = X_1 \ X_n$ (following common practice, we denote the union of sets of attributes by means of the juxtaposition of their names) We assume all the attributes to have a common *domain D* that is the disjoint union of two countably infinite sets, the set of *constants* and the set of *variables* We consider *tuples*, *tableaux*

(finite sets of tuples over the universe), *relations* (finite sets of total tuples over a relation scheme), and *(database) states* (indicated as sets of relations, one for each relation scheme $\mathbf{r} = \{r_1, \ldots, r_n\}$)

The *total projection* of a tableau T on a set of attributes is the set of total tuples that are restrictions of tuples in T $\pi\downarrow_X(T) = \{t[X] | t \in T \text{ and } t[X] \text{ is total}\}$ We will also use a generalization of the total projection, that still operates on tableaux, but produces database states the *projection of a tableau T on a database scheme* $\mathbf{R}$ (indicated with $\pi\downarrow_{\mathbf{R}}(T)$) is the state obtained by totally projecting T on the various relation schemes

2.2 Constraints: local satisfaction and global consistency

Associated with a database scheme there is usually a set of *constraints*, that is, properties that are satisfied by the legal states There are two notions of satisfaction *local* satisfaction, defined on individual relations, and *global* satisfaction, or *consistency*, defined on the database state We introduce the two concepts in turn Various classes of constraints have been defined in the literature [15,23], here, we limit our attention to functional dependencies

Let $YZ \subseteq X \subseteq U$, a relation r over a scheme $R(X)$, *(locally) satisfies* the *functional dependency (FD)* $Y \rightarrow Z$, if, for every pair of tuples $t_1, t_2 \in r$ such that $t_1[Y] = t_2[Y]$, it is the case that $t_1[Z] = t_2[Z]$

Let $\mathbf{R} = \{R_1(X_1), \ldots, R_n(X_n)\}$ be a database scheme, we associate with $\mathbf{R}$ a set of FDs $F = \cup_{i=1}^{n} F_i$, where, for every $1 \leq i \leq n$, the FDs in F_i are defined on $R_i(X_i)$

A state $\mathbf{r} = \{r_1, \ldots, \mathbf{r}_n\}$ *globally satisfies* [13] a set of FDs F, if there is a relation w on the universe U (called a *weak instance* for $\mathbf{r}$ with respect to F) that (locally) satisfies F and contains the relations of $\mathbf{r}$ in its projections over the respective relation schemes $\pi_{X_i}(w) \supseteq r_i$, for $1 \leq i \leq n$ A state that globally satisfies the set of dependencies associated with the database scheme is also said *(globally) consistent*

We recall that the *closure* of a set of FDs F, denoted by F^+, is the set of dependencies that are logically implied by F, and the *closure* of a set of attributes X, denoted by X^+, is the set of attributes $\{A / X \rightarrow A \in F^+\}$

2.3 The Chase Procedure and the Representative Instance

The definition of global satisfaction is interesting, but not practical in general there may be many weak instances (often infinitely many), and there is no direct way to find any of them However, the existence of a weak instance (and some other interesting properties) can be studied by means of the notion of *representative instance*, a tableau over the universe U of the attributes, defined, for each database state $\mathbf{r}$, by means of the notions of *state tableau* and *chase*

For each database state $\mathbf{r}$, the *state tableau* for $\mathbf{r}$ is a tableau (indicated with $T_{\mathbf{r}}$) formed taking the union of all the relations in $\mathbf{r}$ extended to U by means of unique variables

The chase [16] is a procedure that receives in input a tableau T and generates a tableau $CHASE_F(T)$ that, if possible, satisfies the given dependencies F, if only functional dependencies are considered, the process modifies values in the tableau, by equating variables and "promoting" variables to constants If a chase step tries to equate two constants, then we say that the chase encounters a *contradiction*, and the process stops, generating a special tableau, which we call the *inconsistent tableau*, and indicate with T_∞

The *representative instance* for a state $\mathbf{r}$, indicated with $\mathrm{RI}_{\mathbf{r}}$, is the tableau obtained by chasing the state tableau $T_{\mathbf{r}}$ of $\mathbf{r}$ with respect to the dependencies associated with the database state

The main property of the representative instance is that a database state is consistent if and only if the corresponding representative instance is not the inconsistent tableau (that is, it is generated without encountering contradictions during the chase process) [13] Also, for every consistent state $\mathbf{r}$ and for every X, the X-total projection of the representative instance of $\mathbf{r}$ is equal to the set of tuples that appear in the projection on X of every weak instance of $\mathbf{r}$ [18]

The *weak instance approach* to query answering allows queries to be formulated on databases as if they were composed of just one relation over the universe U For every query, being $X \subseteq U$ the set of attributes involved, the evaluation requires a first step that computes the relation over X implied by the current state for the above consideration, it follows that the X-total projection of the representative instance is the natural content of this relation

We say that a tuple t over a set of attributes X *x-belongs* (in symbols $t \hat{\in} \mathbf{r}$) to a consistent state $\mathbf{r}$ of a database scheme $\mathbf{R}$ with a universe $U \supseteq X$ if t belongs to the X-total projection of the representative instance of $\mathbf{r}$

2.4 Tableau Containment and Equivalence

Given two tableau T_1, T_2, we say that T_1 *is contained* in T_2 (in symbols $T_1 \leq T_2$) if there is a partial function ψ (called *containment mapping*) from D to D that *(i)* is defined on all the symbols appearing in T_1, *(ii)* is the identity on constants, and *(iii)* if extended to rows

and tableaux, maps T_1 into a subset of T_2 (that is, for each row $t_1 \in T_1$, there is a row $t_2 \in T_2$, such that $\psi(t_1[A]) = t_2[A]$, for every $A \in X$) If both $T_1 \leq T_2$ and $T_2 \leq T_1$, the two tableau are *equivalent* [1] We assume that the inconsistent tableau properly contains every other tableau

2.5 Independent schemes

A scheme is *independent* [12,20] if, for all its states, local satisfaction implies global satisfaction

Independent schemes are clearly important from the pratical point of view, because the global consistency of their states can be verified in a local manner, looking at the individual relations, without having to build and chase the state tableau Graham and Yannakakis [12] showed an efficient test for independence, later improved by other authors [14,22]

Independent schemes are also important in the weak instance approach to query answering, because they guarantee the efficient computation of total projections of the representative instance Atzeni and Chan [6] and Ito et al [14] showed that for every independent scheme, for every subset X of its universe, there is a relational algebra expression E_X that computes the total projection of the representative instance for every state of the scheme In fact, E_X is the union of *simple chase join expression* (scje's), a restricted form of project-join expression, defined as follows [5,6,11]

A preliminary concept is needed a *derivation sequence (ds)* of some relation scheme R_{i_0} is a finite sequence of FDs $\sigma =< Y_1 \rightarrow Z_1, \ , Y_m \rightarrow Z_m >$ from F^+ such that, for all $1 \leq j \leq m$ $Y_j \subseteq X_{i_0} Z_1 \ Z_{j-1}$ and $Z_j \cap X_{i_0} Z_1 \ Z_{j-1} = \emptyset$ We say that σ *covers* a set of attributes X if $X_{i_0} Z_1 \ Z_j \supseteq X$ Essentially, a ds of R_{i_0} is a sequence of FDs used in computing (part of) the closure of X_{i_0}

Simple chase join expressions (scje's) are defined on the basis of ds's Given the ds σ above, assuming it covers X, the scje for σ *with target* X is the expression

$$\pi_X(R_{i_0} \bowtie \pi_{Y_1 Z_1}(R_{i_1}) \bowtie \ \bowtie \pi_{Y_m Z_m}(R_{i_m}))$$

3 Updates in the Weak Instance model

In this section we review definitions and characterizations about updates in the weak instance model [9]

[1] Different, but equivalent definitions of containment and equivalence of tableaux can be found in the literature [2,3]

3.1 A lattice on states

In the weak instance approach, two states $\mathbf{r}_1$, $\mathbf{r}_2$ are *equivalent* ($\mathbf{r}_1 \sim \mathbf{r}_2$) if they have the same set of weak instances [19], this holds if and only if their representative instances are (tableau) equivalent Therefore, two states are equivalent if and only if, for every X, their X-total projections are equal, that is if they have identical query answering behaviour Therefore, it makes sense to consider equivalence classes of states, as representatives of the various classes, we will often use states that enjoy the interesting property of completeness, as follows

A consistent state $\mathbf{r}$ is *complete* if it coincides with the projection of its representative instance on the database scheme [19], that is if $\mathbf{r} = \pi^{\downarrow}_{\mathbf{R}}(\mathrm{RI}_{\mathbf{r}})$ It is known that each consistent state is equivalent to one and only one complete state [19, Section 3]

Example 3 *The state in Figure 1 is a complete state, but if we remove, for instance, the tuple* < B, Smith > *from* r_3, *we obtain a state that is not complete, since such a tuple belongs to the projection on PM of its representative instance*

Now we say that a state $\mathbf{r}_1$ is *weaker* than a state $\mathbf{r}_2$ ($\mathbf{r}_1 \preceq \mathbf{r}_2$) if every weak instance of $\mathbf{r}_2$ is also a weak instance of $\mathbf{r}_1$ The relation $\preceq$ is a partial order on the set of the complete states, since it is reflexive, antisymmetric, and transitive (Note that it is not antisymmetric on the set of consistent states) In [9] we showed that the partial order $\preceq$ extended to the inconsistent state induces a complete lattice on the set of complete states, that is, every pair of complete states $\mathbf{r}_1$ and $\mathbf{r}_2$ has both a greatest lower bound and a least upper bound

A *lower bound* of $\mathbf{r}_1$ and $\mathbf{r}_2$ is a state that is weaker than both $\mathbf{r}_1$ and $\mathbf{r}_2$, the *greatest lower bound (glb)* of $\mathbf{r}_1$ and $\mathbf{r}_2$ is a lower bound of $\mathbf{r}_1$ and $\mathbf{r}_2$ such that every other lower bound is weaker than it It follows that, if there is a glb, it is unique The *least upper bound (lub)* of $\mathbf{r}_1$ and $\mathbf{r}_2$ is defined dually an *upper bound* is a state such that both $\mathbf{r}_1$ and $\mathbf{r}_2$ are weaker than it, the lub is an upper bound that is weaker than all other upper bounds A known property of lattices is the associativity of both the glb and lub operations therefore, we can speak of glb and lub of finite sets of states A lattice over a domain $\mathcal{D}$ is *complete* [10] if each (finite or infinite) subset of $\mathcal{D}$ has both a glb and a lub

3.2 Insertions

Let $\mathbf{R} = \{R_1, \ , R_k\}$ be a database scheme, with $U = X_1 X_2 \ X_k$ Given a state $\mathbf{r}$ of $\mathbf{R}$ and a tuple t over a set of attributes $X \subseteq U$, we consider the *insertion* of t into $\mathbf{r}$ defined through the following notion of result

A state $\mathbf{r}_p$ is a *potential result* for the insertion of t into $\mathbf{r}$ if $\mathbf{r} \preceq \mathbf{r}_p$ and $t \widehat{\in} \mathbf{r}_p$ Various cases for an insertion of a tuple in a consistent state exists the insertion of a tuple t in a state is *possible* if there is a consistent state $\mathbf{r}'$ such that $t \widehat{\in} \mathbf{r}'$, a possible insertion is *consistent* if it has a consistent potential result, and a possible and consistent insertion is *deterministic* if the glb of the potential results is a potential result

In [9] we showed necessary and sufficient conditions for possibility, consistency, and determinism

Theorem 1 *[9, Theorem 1] The insertion of t in a state is possible if and only if there is a relation scheme $R_i(X_i) \in \mathbf{R}$ such that F implies the FD $X_i \rightarrow X$*

Let $\mathrm{RI}_{\mathbf{r}}$ be the representative instance of $\mathbf{r}$ The characterization of both consistency and determinism is based on the construction of a particular tableau obtained by adding to $\mathrm{RI}_{\mathbf{r}}$ a tuple t_o obtained by extending t to the universe U by means of unique variables Let $T_t\,\mathbf{r}$ be such a tableau

Theorem 2 *[9, Theorem 2] Let the insertion of t in $\mathbf{r}$ be possible It is consistent if and only if it is the case that $\mathcal{CHASE}_F(T_t\,\mathbf{r}) \neq T_\infty$*

Let $\mathbf{r}_+$ be the state obtained by (totally) projecting $\mathcal{CHASE}_F(T_t\,\mathbf{r})$ on the database scheme $\mathbf{r}_+ = \pi^{\downarrow}_{\mathbf{R}}(\mathcal{CHASE}_F(T_t\,\mathbf{r}))$

Lemma 1 *[9, Lemma 4] Let the insertion of t in $\mathbf{r}$ be possible and consistent Then, $\mathbf{r}_+$ is the glb of the potential results*

Theorem 3 *[9, Theorem 9] Let the insertion of t in $\mathbf{r}$ be possible and consistent It is deterministic if and only if $\mathcal{CHASE}_F(T_t\,\mathbf{r}) \equiv \mathrm{RI}_{\mathbf{r}_+}$*

Corollary 1 *[9, Corollary 1] Let the insertion of t in $\mathbf{r}$ be possible and consistent, it is deterministic if and only if $t \in \pi^{\downarrow}_X(\mathrm{RI}_{\mathbf{r}_+})$*

Corollary 1 gives an effective characterization of insertibility given $\mathbf{r}$ and t, we can build $T_t\,\mathbf{r}$, chase it with respect to the given constraints, then generate $\mathbf{r}_+$ and compute its representative instance $\mathrm{RI}_{\mathbf{r}_+}$, and finally check whether the total projection $\pi^{\downarrow}_X(\mathrm{RI}_{\mathbf{r}_+})$ contains t

Example 4 *Consider the first insertion in Example 2 Following the definitions, we could build the tableau $T_t\,\mathbf{r}$, chase it and project the result on the database scheme we obtain the state we suggested as a result We just note that beside the tuple $<$ MS, White $>$ that is added to r_2, the tuple $<$ C, White $>$ is also added to r_3, and the state remains complete*

3.3 Deletions

The definitions are somehow symmetric with respect to those concerning insertions However, the case is easier, because no problem arises regarding consistency and possibility

A state $\mathbf{r}_p$ is a *potential result* for the deletion of a tuple t from a state $\mathbf{r}$ if $\mathbf{r}_p \preceq \mathbf{r}$ and $t \widehat{\notin} \mathbf{r}_p$ A deletion is *deterministic* if the lub of the potential results is a potential result

It turns out that deletions are deterministic only in very restricted cases

Lemma 2 *[9, Lemma 5] Let $\mathbf{r}$ be a consistent state and t be a tuple on X that x-belong to $\mathbf{r}$ The deletion of t from $\mathbf{r}$ is deterministic only if there is a relation scheme $R_i(X_i)$ such that $X \subseteq X_i$*

Let $\mathbf{r}_-$ be the state obtained by removing, from each relation r_i such that $X \subseteq X_i$, each tuple t' such that $t'[X] = t[X]$

Theorem 4 *[9, Theorem 4] Let $\mathbf{r}$ be a consistent state and t be a tuple on X that x-belong to $\mathbf{r}$ The deletion of t from $\mathbf{r}$ is deterministic if and only if (i) there is a relation scheme $R_i(X_i)$ such that $X \subseteq X_i$ and (ii) t does not x-belong to $\mathbf{r}_-$*

4 Efficient Insertions for Independent Schemes

In the same way as query answering can be efficiently performed for independent schemes, we want to show that, for this meaningful class of schemes, updates can be managed efficiently

With regard to deletions, Theorem 4 already gives an efficient way of checking for determinism and for performing the update With respect to insertions, things are more complex in general, because the chase of a tableau involving the whole database state is needed

In this section, we show that, for independent schemes, there is an efficient method for checking the possibility, consistency, and determinism of the insertions

With respect to possibility, Theorem 1 gives a complete characterization at the scheme level, which can be verified very efficiently, by using known algorithms for the implication of FDs [15,23]

4.1 Consistency

Throughout this section we consider a consistent state $\mathbf{r}$ for an independent scheme $\mathbf{R}$ and the insertion of a tuple t over $X \subseteq U$ in $\mathbf{r}$, assuming it is possible

Algorithm 1
begin
$W = X$,
$t_W = t$,
while *exists* $V \rightarrow A \in F_j$ *for some* $1 \leq j \leq n$ *such that* $V \subseteq W, A \notin W$
and *exists* $t' \in r_j$ *such that* $t'[V] = t_W[V]$
do begin
$W = W \cup A$,
$t_W[A] = t'[A]$
end;
$\bar{t} = t_W$,
$\overline{X} = W$
end

Figure 4 Algorithm 1

Let $\bar{t}$ be the "extension" of t generated by Algorithm 1 (shown in Figure 4)

It turns out that $\bar{t}$ has interesting properties, which make it fundamental in the efficient check of both consistency and determinism Let us introduce a property to be used shortly

Condition 1 *There is no* $V \rightarrow A \in F_j$ *such that*

- $VA \subseteq \overline{X}$ *and*
- *there exists* $t' \in r_j$ *such that* $t'[V] = \bar{t}[V]$ *and* $t'[A] \neq \bar{t}[A]$

Condition 1 is clearly sufficient to guarantee that $\bar{t}$ is uniquely defined The next theorem shows that it characterizes consistency

Theorem 5 *The insertion of* t *in* $\mathbf{r}$ *is consistent if and only if condition 1 holds*

This theorem gives an effective and efficient method to check for consistency of insertions in independent schemes Instead of performing the chase of $T_t\,\mathbf{r}$ (as required by Theorem 2), it is sufficient to apply Algorithm 1 and to check for violations of FD's involving $\bar{t}$

Example 5 *The scheme in Example 1 is clearly independent Now consider the first insertions in Example 2 We have* $\bar{t} =<$ Jim, MS, White $>$, *so the insertion is consistent, since such a tuple does not violate the FDs that involves On the contrary, if we want to insert the tuple* $<$ John, White $>$, *we would have an inconsistent insertion since* $\bar{t} =<$ John, CS, White $>$, *and Condition 1 does not hold for the FD* $D \rightarrow M$ *and the tuple* $<$ CS, Smith $>$ *of* r_2

4.2 Determinism

The characterization of deterministic insertions (Theorem 3) is based on the generation of $\mathbf{r}_+$ by means of the chase of a tableau involving the whole database In this section, we present a simpler method for generating the minimum result for an insertion in independent schemes, which does not require the generation of $\mathbf{r}_+$

Throughout this section we consider a consistent state $\mathbf{r}$ for an independent scheme $\mathbf{R}$ and the insertion of a tuple t over $X \subseteq U$ in $\mathbf{r}$, assuming it is possible and consistent Moreover let $\bar{t}$ be the tuple obtained from $\mathbf{r}$ by means of Algorithm 1 and $\bar{\mathbf{r}}$ be the state obtained by adding the tuple $\bar{t}[X_j]$ to each relation $r_j \in \mathbf{r}$ such that $X_j \subseteq \overline{X}$ We will show that $\bar{\mathbf{r}}$ is strongly related to $\mathbf{r}_+$

By definition, $\mathbf{r}_+$ is obtained by (totally) projecting $CHASE_F(T_t\,\mathbf{r})$ on the database scheme Then, as the next step in finding the relationship between $\bar{\mathbf{r}}$ and $\mathbf{r}_+$, using Algorithm 2 (in Figure 5) we show that, for independent schemes, the chase of $T_t\,\mathbf{r} = \mathrm{RI}_{\mathbf{r}} \cup \{t_e\}$ can be performed in a particular way

It is immediate that $CHASE_F(T_2^*) = CHASE_F(T_t\,\mathbf{r})$ because both phase 1 and phase 2 consists in chase steps, and the order in which they are applied is not relevant [15]

Now, we note that phase 1 essential coincides with the construction of $\bar{t}$ (Algorithm 1) Then let $\mathbf{r}_1 = \pi^{\downarrow}_{\mathbf{R}}(T_1^*)$, it is easy to show that $\bar{\mathbf{r}} \sim \mathbf{r}_1$

The next lemma states a fundamental property of phase 3

Lemma 3 *Let the insertion of* t *in* $\mathbf{r}$ *be deterministic and consider chasing* $T_t\,\mathbf{r}$ *using Algorithm 2 In phase 3 of Algorithm, only variables are equated*

Let $\mathbf{r}_2 = \pi^{\downarrow}_{\mathbf{R}}(T_2^*)$, from the above lemma it follows

Algorithm 2
begin
/*phase 1*/
$t_W = t_e$,
$W = X$,
while *exists* $V \rightarrow A \in F_j$ *for some* $1 \leq j \leq n$ *such that* $V \subseteq W, A \notin W$
and *exists* $t' \in r_j$ *such that* $t'[V] = t_W[V]$
do begin
$W = W \cup A$,
$t_W[A] = t'[A]$
end, /*let $T_1^* = T_t \mathbf{r} \cup \{t_W\}$*/
/*phase 2*/
while *exists* $V \rightarrow A \in F_j$ *for some* $1 \leq j \leq n$ **and**
exist $t_1, t_2 \in T_1^*$ *such that* $t_1[YA]$ *is total and* $t_1[V] = t_2[V]$
do $t_2[A] = t_1[A]$, /*let T_2^* the obtained tableau*/
/*phase 3*/
Apply the chase to T_2^*
end

Figure 5 Algorithm 2

directly that if the insertion is deterministic then $\mathbf{r}_2 = \mathbf{r}_+$

Now, it is possible to prove that $\mathbf{r}_1$ and $\mathbf{r}_2$ are equivalent, this is a major step in the proof of the following result

Lemma 4 *If the insertion is deterministic then* $\bar{\mathbf{r}} \sim \mathbf{r}_+$

Example 6 *Consider again the first insertion in Example 2, we stated that it is a deterministic insertion Now we have* $\bar{t} =<$ Jim, MS, White $>$, *so* $\bar{\mathbf{r}}$ *is obtained from* $\mathbf{r}$ *by simply adding to* r_2 *the tuple* $<$ MS, White $>$ *Such a state does not coincide with* $\mathbf{r}_+$ *but it is equivalent to it, since the tuple* $<$ C, White $>$, *that is not in* $\bar{\mathbf{r}}$, *belongs to the projection on PM of its representative instance*

As a consequence of Lemma 4, we have that if the insertion is deterministic, then $t \hat{\in} \bar{\mathbf{r}}$ The converse of this claim is simple to prove, and therefore we obtain the following main theorem

Theorem 6 *The insertion of* t *in* $\mathbf{r}$ *is deterministic if and only if* $t \hat{\in} \bar{\mathbf{r}}$

Theorem 6 gives an alternative method to check for determinism that does not involve a chase of a state tableau In the next section we present an Algorithm to verify efficiently whether $t \hat{\in} \bar{\mathbf{r}}$

5 Check for determinism

Let $\mathbf{r}$ be a consistent state for an independent scheme $\mathbf{R}$, and consider the insertion of a tuple t over $X \subseteq U$ in $\mathbf{r}$, assuming that it is possible and consistent Moreover, let $\bar{t}$ be the tuple obtained from $\mathbf{r}$ by Algorithm 1

Atzeni and Chan [6] proved that total projections of representative instances can be computed by means of unions of scje's and proposed an algorithm to compute and optimize this expression in polynomial time Let $E_X = \cup E_i$ be the expression obtained for X Algorithm 3 (shown in Figure 6) checks if $t \hat{\in} \bar{\mathbf{r}}$ making use of E_X

Before arguing for the correctness of the algorithm we informally describe it by means of an example

Example 7 *Algorithm 3 tries to produce the tuple of* $\mathrm{RI}_{\bar{\mathbf{r}}}$ *that, for deterministic insertions, contains t (Theorem 6) The expression* E_X *gives us the relations involved in the construction of the total projection on X of the representative instance The tuple is generated joining tuples of* $\mathbf{r}$, *from relations* r_{i_j}, *for schemes* R_{i_j} *such that* $\overline{X} \not\supseteq X_{i_j}$, *and* $\bar{t}$ *for schemes* R_{i_k} *such that* $\overline{X} \supseteq X_{i_k}$, *selecting the tuples that coincide with* $\bar{t}$ *on the attributes of* $\overline{X}$ *If we want to insert in the state of Figure 1 the tuple* $<$ Jim, White $>$, *we have* $\bar{t} =<$ Jim, MS, White $>$ *and* $E_X = \pi_{EM}(R_1 \bowtie \pi_{DM}(R_2))$ *the algorithm returns* true *with* $q_1 =<$ Jim, MS, C, White $>$ *This tuple belongs to* $\mathrm{RI}_{\bar{\mathbf{r}}}$, *and, since it contains t, by Theorem 6, it confirms the determinism of the insertion*

The proof of correctness of Algorithm 3 is based on the following lemma

Algorithm 3
Input r, $\bar{t}(\overline{X})$ *and* E_X *as above,*
Output **true** *or* **false,**
begin
repeat
select a scje $E_i = \pi_X(R_{i_0} \bowtie \pi_{Y_1 Z_1}(R_{i_1}) \bowtie \quad \bowtie \pi_{Y_m Z_m}(R_{i_m}))$ *from* E_X,
determinıstıc = **true,**
ıf $\overline{X} \supseteq X_{i_0}$
then $q_0 = \bar{t}[X_{i_0}]$
else begin
$X_0 = X_{i_0} \cap \overline{X}$,
$q_0 = r_{i_0} \bowtie \{\bar{t}[X_0]\}$, /*$q_0 = r_{i_0}$ *for* $X_0 = \emptyset$*/
if $q_0 = \{\}$ **then** *determınıstıc* = **false,**
end,
$j = 0$,
while *determınıstıc* **and** $j < m$
do begin
$j = j+1$,
if $\overline{X} \supseteq X_{i_j}$
then $q_j = q_{j-1} \bowtie \{\bar{t}[Y_j Z_j]\}$
else begin
$X_j = Y_j Z_j \cap \overline{X}$,
$q_j = q_{j-1} \bowtie \pi_{Y_j Z_j}(r_{i_j}) \bowtie \{\bar{t}[X_j]\}$,
/*$q_j = q_{j-1} \bowtie \pi_{Y_j Z_j}(r_{i_j})$ *for* $X_j = \emptyset$*/
if $q_j = \{\}$ **then** *determınıstıc* = **false**
end
end
untıl *determınıstıc* **or** *the scje are fınıshed,*
return *determınıstıc*
end.

Figure 6 Algorithm 3

Lemma 5 *If the insertion of t in $\mathbf{r}$ is deterministic then there exists a tuple $\hat{t}$ in $\mathrm{RI}_{\mathbf{F}}$ such that $t \leq \bar{t} \leq \hat{t}$*

Algorithm 3 tries to construct the tuple $\hat{t}$ mentioned in Lemma 5, that is the tuple of $\mathrm{RI}_{\mathbf{F}}$ that, if the insertion is deterministic, contains t This is shown by next lemma

Lemma 6 *Let E_X and $\hat{t}$ as above and q_j be the relation produced in step j of Algorithm 3 If the insertion is deterministic it is the case that, for some $E_i \in E_X$, for each $0 \leq j \leq m$ $\hat{t}[R_i, Z_1 \quad Z_j] \in q_j$*

The final theorem follows as a consequence

Theorem 7 *Algorithm 3 is correct*

6 Conclusions

The results in Sections 4 and 5 can be used to obtain an efficient implementation of insertions for independent schemes Possibility can be tested efficiently at the scheme level, by Theorem 1 As regards consistency, Algorithm 1 constructs $\bar{t}$ in time polynomial in the size of the database scheme and the size of the set of FDs F, provided that the time needed to search for tuples given a value on the left hand side of an FD is constant (this can be guaranteed by an index), for the same reason, Condition 1 can be tested efficiently As regards determinism, for any given X, there are at most as many scje's in E_X as relation schemes in $\mathbf{R}$ [4,6], therefore the test can be reconducted to the execution of a bounded number of relational algebra expressions (which can be optimized efficiently [5]) Finally, the result of the insertion is obtained by simply adding to the state the projections of $\bar{t}$ on the schemes on which it is total

References

[1] S Abiteboul Updates, a new frontier In *ICDT'88 (Second International Conference on Data Base Theory), Bruges, Lecture Notes in Computer Science 326*, pages 1–18, Springer-Verlag, 1988

[2] A V Aho, Y Sagiv, and J D Ullman Efficient optimization of a class of relational expressions *ACM Trans on Database Syst*, 4(4) 435–454, 1979

[3] A V Aho, Y Sagiv, and J D Ullman Equivalence of relational expressions *SIAM Journal on Computing*, 8(2) 218–246, 1979

[4] P Atzeni and E P F Chan Efficient and optimal query answering on independent schemes *Theoretical Computer Science*, 71(2), March 1990 To appear

[5] P Atzeni and E P F Chan Efficient optimization of simple chase join expressions *ACM Trans on Database Syst*, 14(2) 212–230, June 1989

[6] P Atzeni and E P F Chan Efficient query answering in the representative instance approach In *Fourth ACM SIGACT SIGMOD Symp on Principles of Database Systems*, pages 181–188, 1985

[7] P Atzeni and M C De Bernardis A new basis for the weak instance model In *Sixth ACM SIGACT SIGMOD SIGART Symp on Principles of Database Systems*, pages 79–86, 1987

[8] P Atzeni and R Torlone *Efficient Updates to Independent Schemes in the Weak Instance Model* Rapporto R 281, IASI-CNR, Roma, 1989

[9] P Atzeni and R Torlone Updating databases in the weak instance model In *Eigth ACM SIGACT SIGMOD SIGART Symp on Principles of Database Systems*, pages 101–109, 1989

[10] G Birkhoff *Lattice Theory Colloquium Publications, Volume XXV*, American Mathematical Society, third edition, 1967

[11] E P F Chan Optimal computation of total projections with unions of simple chase join expressions In *ACM SIGMOD International Conf on Management of Data*, pages 149–163, 1984

[12] M H Graham and M Yannakakis Independent database schemas *Journal of Comp and System Sc*, 28(1) 121–141, 1984

[13] P Honeyman Testing satisfaction of functional dependencies *Journal of the ACM*, 29(3) 668–677, 1982

[14] M Ito, M Iwasaki, and T Kasami Some results on the representative instance in relational databases *SIAM Journal on Computing*, 14(2) 334–354, 1985

[15] D Maier *The Theory of Relational Databases* Computer Science Press, Potomac, Maryland, 1983

[16] D Maier, A O Mendelzon, and Y Sagiv Testing implications of data dependencies *ACM Trans on Database Syst*, 4(4) 455–468, 1979

[17] D Maier, D Rozenshtein, and D S Warren Window functions In P C Kanellakis and F Preparata, editors, *Advances in Computing Research, Vol 3*, pages 213–246, JAI Press, 1986

[18] D Maier, J D Ullman, and M Vardi On the foundations of the universal relation model *ACM Trans on Database Syst*, 9(2) 283–308, 1984

[19] A O Mendelzon Database states and their tableaux *ACM Trans on Database Syst*, 9(2) 264–282, 1984

[20] Y Sagiv Can we use the universal instance assumption without using nulls? In *ACM SIGMOD International Conf on Management of Data*, pages 108–120, 1981

[21] Y Sagiv A characterization of globally consistent databases and their correct access paths *ACM Trans on Database Syst*, 8(2) 266–286, 1983

[22] Y Sagiv On computing restricted projections of the representative instance In *Fourth ACM SIGACT SIGMOD Symp on Principles of Database Systems*, pages 173–180, 1985

[23] J D Ullman *Principles of Database Systems* Computer Science Press, Potomac, Maryland, second edition, 1982