Database Repairing Using Updates

JEF WIJSEN
Université de Mons-Hainaut

Repairing a database means bringing the database in accordance with a given set of integrity constraints by applying some minimal change. If a database can be repaired in more than one way, then the consistent answer to a query is defined as the intersection of the query answers on all repaired versions of the database.

Earlier approaches have confined the repair work to deletions and insertions of entire tuples. We propose a theoretical framework that also covers updates as a repair primitive. Update-based repairing is interesting in that it allows rectifying an error within a tuple without deleting the tuple, thereby preserving consistent values in the tuple. Another novel idea is the construct of nucleus: a single database that yields consistent answers to a class of queries, without the need for query rewriting. We show the construction of nuclei for full dependencies and conjunctive queries. Consistent query answering and constructing nuclei is generally intractable under update-based repairing. Nevertheless, we also show some tractable cases of practical interest.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Relational databases*; *Query processing*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Theory

Additional Key Words and Phrases: Consistent query answering, database repairing

1. GENERAL PROBLEM DESCRIPTION

Database textbooks generally explain that integrity constraints are used for capturing the set of all "legal" databases and hence should be satisfied at all times. Nevertheless, many real-life databases contain data that is known or suspected to be inconsistent. Inconsistency may be caused, among other reasons, by data integration and underspecified constraints. For example, the rule "No employee has more than one contact address," gives rise to an error if two databases to be integrated store different addresses for the same employee. A `FIRSTNAME CHAR(20)` declaration in SQL does not protect us from inputting illegal first names like "Louis14." When later on we specify that first names cannot contain numbers, the database may already turn out to be inconsistent.

Author's address: Université de Mons-Hainaut, Avenue du Champs de Mars 6, B-7000 Mons, Belgium; email: wijsen@umh.ac.be.

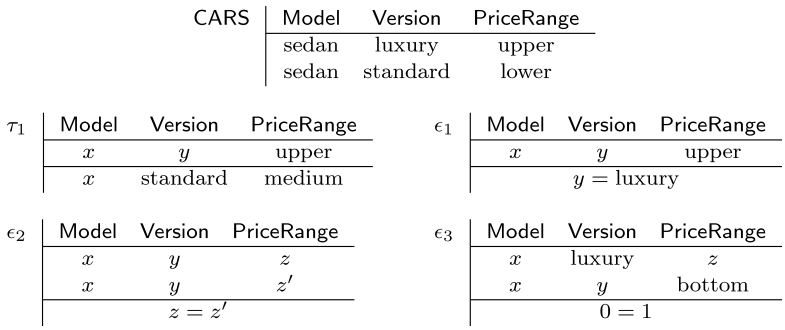


Fig. 1. Example relation with four full dependencies.

Since database inconsistency is a widespread phenomenon, it is important to understand how to react to it. The seminal work of Arenas et al. [1999] has roused much research in the construct of *repair* for dealing with inconsistency. In broad outline, a repair of an inconsistent database I is a database J that is consistent and "as close as possible" to I. Closeness can be captured in different ways and this has given rise to a variety of repair notions. Whatever repair notion is used, a repair of a given database I need not generally be unique. When there are multiple repairs, the question that arises is which repair to use for answering queries. The generally accepted query semantics is to execute the query on each repair and return the intersection of all answers, the so-called *consistent query answer*. Intuitively, all repairs are equally possible and only tuples that appear in all answers are certainly true.

Nearly all approaches so far have assumed that databases are repaired by deleting and inserting entire tuples. A problem with deletion/insertion-based repairing is that a single tuple may contain both correct and erroneous components. When we delete a tuple because it contains an error, we also lose the correct components as an undesirable side effect. This effect may be dramatic in real-life databases containing relations with many error-free attributes along with one or two error-prone attributes. To overcome this problem, we propose a notion of repair that allows updating erroneous components in place, while keeping the consistent ones. We refer to our approach as "updated-based repairing."

For example, the relation CARS of Figure 1 stores price ranges of different car variants; possible price ranges are: bottom, lower, medium, upper. Integrity constraints are expressed in the tableau formalism [Abiteboul et al. 1995], using column names for readability. The following constraints apply:

—Every model in the upper price range also exists in a medium priced standard version (τ_1).

—The upper price range only contains luxury cars (ϵ_1). On the other hand, luxury cars need not be upper priced.

—The price range of a car is determined by its model and version (ϵ_2). This is a *key dependency*.

—Models that exist in a luxury version are never available at bottom prices (ϵ_3). The constants 0 and 1 can be replaced by any pair of distinct constants.

F_1

Model	Version	PriceRange
x	luxury	upper
sedan	standard	lower

T_1

Model	Version	PriceRange
x	luxury	upper
x	standard	medium
sedan	standard	lower

F_2

Model	Version	PriceRange
sedan	luxury	z
sedan	standard	lower

$T_2 = F_2$

F_3

Model	Version	PriceRange
sedan	luxury	upper
x	standard	lower

T_3

Model	Version	PriceRange
sedan	luxury	upper
sedan	standard	medium
x	standard	lower

F_4

Model	Version	PriceRange
sedan	luxury	upper
sedan	y	lower

T_4

Model	Version	PriceRange
sedan	luxury	upper
sedan	standard	medium
sedan	y	lower

F_5

Model	Version	PriceRange
sedan	luxury	upper
sedan	standard	z

T_5

Model	Version	PriceRange
sedan	luxury	upper
sedan	standard	medium

Fig. 2. F_1 is a fix; the corresponding uprepairs are obtained from T_1 by substituting for x any value distinct from "sedan." Likewise for F_2–F_5.

This is an example of a *contradiction-generating dependency* (cgd) and can be stated as: $\forall x \forall y \forall z(\neg R(x, \text{luxury}, z) \vee \neg R(x, y, \text{bottom}))$.

Note that because of ϵ_1, we can substitute "luxury" for the variable y in τ_1 without changing the meaning of the constraints.

The relation CARS falsifies τ_1, because it does not contain the tuple ⟨sedan, standard, medium⟩. Adding that tuple results in a violation of constraint ϵ_2. In deletion/insertion-based repairing, we can delete the first tuple of CARS; alternatively, we can delete the second tuple and insert ⟨sedan, standard, medium⟩. In update-based repairing, erroneous values are replaced by variables in so-called *fixes*. Five fixes F_1–F_5 of CARS are shown in Figure 2. The fix F_1, for example, assumes that the value "sedan" in the first tuple of CARS is mistaken. Each fix is homomorphic to the original relation CARS: one can find substitutions for the variables x, y, z that map each fix into CARS. Moreover, each fix is homomorphic to a consistent relation. In fact, F_1 is homomorphic to T_1, and to every relation obtained from T_1 by substituting a constant for x; if we choose a constant distinct from "sedan," then the relation obtained will be consistent and called an *uprepair*. The prefix "up" distinguishes uprepairs from other repair notions in the literature; it may be read as "update" or it can refer to the fact that passing from fixes to uprepairs corresponds to "moving up" in a homomorphism lattice (see later). Significantly, uprepairs, unlike fixes, do not contain variables. To make F_5 consistent, we first add the tuple ⟨sedan, standard, medium⟩ because of τ_1, and then we identify z and "medium" because of ϵ_2. In this way, we obtain the uprepair T_5. Intuitively, the repair work that modifies CARS into T_5 updates "lower" into "medium".

In general, the number of uprepairs can be large or even infinite. The consistent answer to a query is defined as the intersection of the query answers on all uprepairs. In the running example, T_1 gives rise to as many uprepairs as there are possible Model-values distinct from "sedan." The question "Give all versions" yields "luxury" and "standard." The question "Give all lower priced versions" yields the empty answer, because the uprepair T_5 contains no lower priced cars.

The definition of consistent answer is conceptually clean but, since the number of uprepairs can be infinite, it is not clear whether consistent answers can be effectively computed and if so, whether they can be computed in tractable time. In this article, all complexity results refer to data complexity: complexity is in terms of the cardinality of the input relation.

A well-studied strategy for computing consistent query answers, known as *query rewriting,* consists in "pushing" integrity constraints into queries so as to obtain new queries that are guaranteed to return the consistent answer on any, possibly inconsistent, database. Since query rewriting is independent of the underlying database, it gives us tractability provided that the modified queries execute in polynomial time.

In this article, we explore a different route. We show that for full dependencies and conjunctive queries, all uprepairs can be "summarized" into a single tableau G such that the consistent answer to any conjunctive query can be obtained by executing the query on G. The tableau G, called *nucleus*, will be homomorphic to all uprepairs and will be maximal in the sense that any other tableau that is homomorphic to all uprepairs, is also homomorphic to G. A nucleus for the running example is the following tableau G:

G

Model	Version	PriceRange
x	luxury	u
x	standard	v
sedan	standard	w

The queries "Give all versions," and "Give all lower priced versions" on G, indeed yield the same consistent answers as before. Note incidentally the difference with deletion/insertion-based repairing, where the repair obtained by deleting the first tuple of CARS contains no luxury cars.

Rewriting conjunctive queries takes time, and more significantly, usually results in nonconjunctive queries that are more time-consuming. The use of nuclei eliminates the need for query rewriting. A nucleus could be computed once and then used to compute consistent answers to any conjunctive query. The drawback is that the nucleus has to be recomputed when the underlying database is modified. Of course, for classes of queries and constraints for which consistent query answering is tractable, the use of nuclei is only meaningful if they can be computed in polynomial time. In this article, we identify cases where the nucleus construction needs no more time than the time needed to answer a single query obtained from rewriting.

The rest of this article is organized as follows. Section 2 recalls some database constructs and introduces the notion of uprepair. In general, the set of uprepairs can be infinite. Section 3 shows how to effectively query infinitely many

uprepairs in the case of full dependencies and conjunctive queries. Section 4 shows the existence of nuclei and provides an effective way to compute them. The results in Section 2–4 are general in the sense that they do not depend on a particular way of fixing, but simply assume that some set of fixes is given. Section 5 settles the construct of fix and then explores an upper bound for the time complexity of consistent query answering. Section 6 provides results on the nucleus size and the time complexity for computing it. Two classes of dependencies are investigated in detail: key dependencies (Section 7) and contradiction-generating dependencies (Section 8). For both classes, we clarify the frontier between tractability and intractability. Section 9 shows the problems arising when we add inequalities or disjunctions. Finally, Section 10 contains a comparison with existing work.

2. FROM SUBCONSISTENT TABLEAUX TO CONSISTENT RELATIONS

2.1 Preliminaries

We recall the definition of tableau. To simplify the notation, we will assume a unirelational database containing a single relation of arity n.

Definition 2.1. We assume two disjoint, infinite sets **dom** and **var** of *constants* and *variables*. A *symbol* is either a constant or a variable.

We assume an arity n. A *tuple* is a sequence $\langle p_1, \ldots, p_n \rangle$ of symbols. The ith coordinate of tuple t is denoted $t(i)$ ($i \in \{1, 2, \ldots, n\}$).

A *tableau* is a finite set of tuples. A tuple or tableau without variables is called *ground*. A ground tableau is also called a *relation*. If F is a tableau, then $\mathbf{grd}(F) := \{t \in F \mid t \text{ is ground}\}$. The set of all tableaux (of fixed arity n) is denoted $\mathfrak{T}$.

We will use the letters I, J, K to denote relations; F, G, H, T denote tableaux; $\mathbf{I}$ denotes a set of relations; $\mathbf{F}$, $\mathbf{G}$ denote sets of tableaux. We now recall the definition of substitution and its extension to tableaux.

Definition 2.2. A *substitution* is a mapping θ from variables to symbols, extended to be the identity on constants. Substitutions naturally extend to tuples and tableaux: first, $\theta(\langle p_1, \ldots, p_n \rangle) = \langle \theta(p_1), \ldots, \theta(p_n) \rangle$, and second, if F is a tableau, then $\theta(F) = \{\theta(t) \mid t \in F\}$.

We write id for the identity function on symbols, and write $id_{p=q}$, where p and q are not two distinct constants, for a substitution that identifies p and q and that is the identity otherwise. That is, if p is a variable and q a constant, then $id_{p=q} = \{p/q\}$; similarly, mutatis mutandis, if p is a constant and q a variable. If p and q are variables, then $id_{p=q}$ can be either $\{p/q\}$ or $\{q/p\}$; the choice will be immaterial in the technical treatment.

The following definition introduces an order $\succeq$ on $\mathfrak{T}$ based on tableau homomorphism.

Definition 2.3. Let F, G be two tableaux (i.e. $F, G \in \mathfrak{T}$). A *homomorphism* from F to G is a substitution θ for the variables in F such that $\theta(F) \subseteq G$. If such a homomorphism from F to G exists, then F is said to be *homomorphic* to

G, denoted $G \succeq F$. Two tableaux are said to be *equivalent*, denoted $F \sim G$, if and only if $F \succeq G$ and $G \succeq F$. We write $G \succ F$ if and only if $G \succeq F$ and $G \not\sim F$. The relation $\sim$ is an equivalence relation (Lemma 2.4); we write $[F]_\sim$ for the equivalence class of $\sim$ that contains F. A tableau is *reduced* if it is equivalent to no tableau of smaller cardinality.

The relation $\sim$ on tableaux naturally extends to sets of tableaux: two sets $\mathbf{F}$ and $\mathbf{G}$ of tableaux are said to be *equivalent*, denoted $\mathbf{F} \sim \mathbf{G}$, if and only if for every tableau in either set, there exists an equivalent tableau in the other set.

It is well-known [Abiteboul et al. 1995], and significant for the results to follow, that if a tableau F is not reduced, then there exists a substitution θ such that $\theta(F) \subsetneq F$. Also, homomorphisms are known to be intimately related to logical implication. Each tableau G can be associated to a first-order logic sentence, denoted $wff(G)$, defined as $wff(G) := \exists^*(\bigwedge_{t \in G} R(t))$. Then, $G \succeq F$ if and only if $wff(G)$ logically implies $wff(F)$. Consequently, $G \sim F$ if and only if $wff(G)$ and $wff(F)$ are logically equivalent. The following properties are straightforward:

LEMMA 2.4. *Let* $F, G \in \mathfrak{T}$.

(1) *If* $F \supseteq G$, *then* $F \succeq G$.
(2) *If* $F \succeq G$, *then* $\mathbf{grd}(F) \supseteq \mathbf{grd}(G)$.
(3) $\langle \mathfrak{T}, \succeq \rangle$ *is a quasi-order.*[1]
(4) $\sim$ *is an equivalence relation on* $\mathfrak{T}$.

PROOF. Easy. □

2.2 Uprepairs

Consistency of relations is defined relative to a set Σ of integrity constraints. A tableau will be called *subconsistent* if it is homomorphic to a consistent relation. Our approach to repairing an inconsistent relation I consists in finding subconsistent fixes that are one-one homomorphic to I and homomorphic to consistent uprepairs. The actual relationship between I and its fixes will be specified in Section 5. Up to that point, we will assume that a set $\mathbf{F}$ of subconsistent fixes is given and focus on the problem of querying the uprepairs generated by $\mathbf{F}$.

Definition 2.5. We assume arity n. A *constraint* is a closed first-order formula σ using a single predicate symbol of arity n. If J is a relation, then $J \models \sigma$ denotes that J satisfies σ (according to standard first-order logic semantics).

Let Σ be a set of constraints. We write $J \models \Sigma$ if and only if $J \models \sigma$ for each $\sigma \in \Sigma$. The relation J is called *consistent* (with respect to Σ) if and only if $J \models \Sigma$; otherwise J is *inconsistent*. The set Σ is called *satisfiable* if it allows a consistent relation. The set Σ is called *coherent* if it allows a nonempty consistent relation. Two sets of constraints are *equivalent* if any relation that is consistent with respect to either set, is also consistent with respect to the other set.

A tableau F is *subconsistent* (with respect to Σ), denoted $F \Vdash \Sigma$, if and only if F is homomorphic to a consistent relation: $J \succeq F$ for some consistent relation J.

[1]A quasi-order satisfies reflexivity and transitivity.

Let F be a subconsistent tableau. An *uprepair* of F and Σ is a minimal (with respect to $\subseteq$) consistent relation J satisfying $J \succeq F$. In general, there will be more than one uprepair of F and Σ; the set of all uprepairs of F and Σ is denoted $F^{\uparrow\Sigma}$. The latter operator naturally extends to a set $\mathbf{F}$ of subconsistent tableaux:

$$\mathbf{F}^{\uparrow\Sigma} := \bigcup_{F \in \mathbf{F}} F^{\uparrow\Sigma} \ .$$

If $J \in \mathbf{F}^{\uparrow\Sigma}$, then we also say that J is *generated* by $\mathbf{F}$ and Σ.

Note that $F^{\uparrow\Sigma}$ need not be finite. For example, for the subconsistent tableau F_1 of Figure 2, ${F_1}^{\uparrow\{\tau_1,\epsilon_1,\epsilon_2,\epsilon_3\}}$ contains every relation obtained from T_1 by replacing x by a constant distinct from "sedan."

Equivalent sets of subconsistent tableaux generate the same uprepairs.

LEMMA 2.6. *Let Σ be a set of constraints. Let $\mathbf{F}$, $\mathbf{G}$ be sets of subconsistent tableaux such that $\mathbf{F} \sim \mathbf{G}$. Then, $\mathbf{F}^{\uparrow\Sigma} = \mathbf{G}^{\uparrow\Sigma}$.*

PROOF. Assume $J \in \mathbf{F}^{\uparrow\Sigma}$. We can assume the existence of $F \in \mathbf{F}$ such that $J \succeq F$ and for each relation J', $J \supset J' \succeq F$ implies $J' \not\models \Sigma$. Since $\mathbf{F} \sim \mathbf{G}$, we can assume the existence of $G \in \mathbf{G}$ such that $F \sim G$, hence $F \succeq G$ and $G \succeq F$. By transitivity, $J \succeq G$. Assume a relation J' such that $J \supset J' \succeq G$. By transitivity, $J \supset J' \succeq F$, hence $J' \not\models \Sigma$. It is correct to conclude $J \in \mathbf{G}^{\uparrow\Sigma}$. It follows $\mathbf{F}^{\uparrow\Sigma} \subseteq \mathbf{G}^{\uparrow\Sigma}$. By symmetry, $\mathbf{G}^{\uparrow\Sigma} \subseteq \mathbf{F}^{\uparrow\Sigma}$. □

3. QUERYING THE UPREPAIRS

In this section, we study an initial problem underlying consistent query answering: given a finite set $\mathbf{F}$ of subconsistent tableaux and a query τ, compute the intersection of the query answers on each $J \in \mathbf{F}^{\uparrow\Sigma}$, i.e. compute $\bigcap_{J \in \mathbf{F}^{\uparrow\Sigma}} \tau(J)$.

3.1 Tableau Queries and Full Dependencies

We use the tableau formalism to express conjunctive queries [Abiteboul et al. 1995]. The answer to a tableau query on a set of relations is defined as the intersection of the query answers on each relation of the set.

Definition 3.1. A *tableau query* is a pair (B, h) where B is a tableau (called *body*) and h is a tuple (called *summary* or *head*) such that every variable in h also occurs in B; B and h need not have the same arity. The class of tableau queries is denoted CQ (acronym for Conjunctive Queries).

Let $\tau = (B, h)$ be a tableau query, and F a tableau of the same arity as B. The *answer* to τ on input F, denoted $\tau(F)$, is the tableau defined as $\tau(F) := \{\theta(h) \mid \theta \text{ is a substitution and } \theta(B) \subseteq F\}$. Obviously, if F is ground, then so is $\tau(F)$.

A tableau query $\tau = (B, h)$ is called *Boolean* if h contains no variables; we will often choose $h = \langle 1 \rangle$, where $1 \in \mathbf{dom}$. Intuitively, the answer $\{\langle 1 \rangle\}$ can be thought of as **true**, and the answer $\{\}$ as **false**.

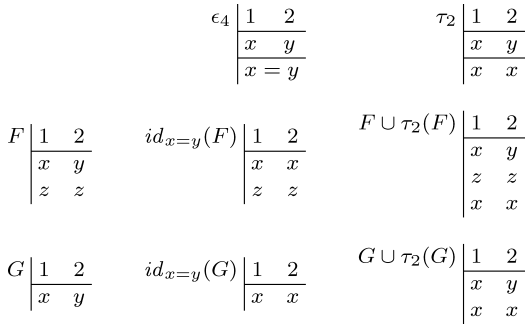


Fig. 3.

For a set $\mathbf{I}$ of relations and a tableau query τ, we define:

$$\tau(\mathbf{I}) := \bigcap_{I \in \mathbf{I}} \tau(I) \ .$$

Certain results in this article are limited to full dependencies, a restricted but interesting class of database constraints [Abiteboul et al. 1995]. Unlike the seminal paper by Beeri and Vardi [1984], our full dependencies need not be typed and can contain constants. All constraints of our running example (see Figure 1) are full dependencies.

Satisfaction of full dependencies by relations is borrowed from first-order logic semantics. We also define what it means for a tableau to satisfy a set of full dependencies. We are not aware of existing work introducing the same concept.

Definition 3.2. A *full dependency* is either a *full tuple-generating dependency* (ftgd) or a *full equality-generating dependency* (fegd).

An ftgd takes the form of a tableau query (B, h) where B and h have the same arity. The ftgd $\tau = (B, h)$ is *satisfied* by a tableau F, denoted $F \models \tau$, if and only if $F \cup \tau(F) \sim F$.

An fegd is of the form $(B, p = q)$ where B is a tableau and p, q are symbols such that every variable in $\{p, q\}$ also occurs in B. The fegd $\epsilon = (B, p = q)$ is *satisfied* by a tableau F, denoted $F \models \epsilon$, if and only if for every substitution θ, if $\theta(B) \subseteq F$, then $\theta(p), \theta(q)$ are not two distinct constants and $F \sim id_{\theta(p) = \theta(q)}(F)$.

Consistency (see Definition 2.5) carries over from relations to tableaux: a tableau F is called *consistent* with respect to a set Σ of full dependencies if and only if $F \models \Sigma$.

If P is a tableau, a tableau query, an ftgd, or an fegd, then the *active symbol domain* of P, denoted $\mathbf{asd}(P)$, is the set of symbols occurring in P.

Satisfaction of full dependencies by tableaux needs special attention, as illustrated by the tableaux F and G, and the dependencies ϵ_4 and τ_2 in Figure 3. Since $F \sim id_{x = y}(F) \sim F \cup \tau_2(F)$, F satisfies both ϵ_4 and τ_2. Note that in verifying whether F satisfies ϵ_4, one must not consider F as a relation by interpreting x and y as distinct constants. Since $G \not\sim id_{x = y}(G)$ and $G \not\sim G \cup \tau_2(G)$, G satisfies neither ϵ_4 nor τ_2.

Clearly, if a relation I is consistent with respect to a set of fegd's, then so is every subset of I. This is no longer true for tableaux, since $G \subseteq F$, $F \models \epsilon_4$, but $G \not\models \epsilon_4$. Although this behavior may be somewhat counterintuitive at first sight, we can give two arguments in favor of it. First, our definition of satisfaction ensures that equivalent tableaux satisfy the same full dependencies (see Theorem 3.7), which is obviously a nice property. Note that $F \sim \{\langle z, z\rangle\}$: the tuple $\langle x, y\rangle$ is redundant in F. In first-order logic terms, $\exists x \exists y \exists z(R(x, y) \wedge R(z, z))$ is logically equivalent to $\exists z(R(z, z))$. Second, this behavior is coherent with querying. Consider the Boolean tableau query:

$$\begin{array}{c|cc} \tau & 1 & 2 \\ \hline & v & v \\ \hline & 1 & \end{array} \quad .$$

Intuitively, since ϵ_4 expresses $\forall x \forall y(R(x, y) \rightarrow x = y)$, every nonempty tableau H satisfying ϵ_4 necessarily contains a tuple t such that $t(1) = t(2)$, hence $\tau(H) = \{\langle 1\rangle\}$. Since G is nonempty but $\tau(G) = \{\}$, we conclude that G must falsify ϵ_4. On the other hand, $\tau(F) = \{\langle 1\rangle\}$. Note incidentally that no tableau query can differentiate between F and $\{\langle z, z\rangle\}$: for every tableau query τ, $\tau(F) \sim \tau(\{\langle z, z\rangle\})$.

We now prove three lemmas that are needed in the main theorem to follow, which expresses that equivalent tableaux satisfy the same full dependencies. The operator $\circ$ denotes function composition.

LEMMA 3.3. *Let F be a tableau, and θ a substitution for the variables in F. Let $\tau = (B, h)$ be a tableau query. Then $\theta(\tau(F)) \subseteq \tau(\theta(F))$.*

PROOF. Let $t \in \theta(\tau(F))$. We can assume $s \in \tau(F)$ such that $\theta(s) = t$. There exists a substitution ω such that $\omega(B) \subseteq F$ and $\omega(h) = s$. Hence, $\theta \circ \omega(B) \subseteq \theta(F)$. Hence, $\theta \circ \omega(h) = t \in \tau(\theta(F))$. □

LEMMA 3.4. *Let F, G be tableaux and τ a tableau query. If $F \succeq G$, then $\tau(F) \succeq \tau(G)$ and $F \cup \tau(F) \succeq G \cup \tau(G)$.*

PROOF. Assume $F \succeq G$. We can assume a substitution θ such that $\theta(G) \subseteq F$. It is easy to see $\tau(\theta(G)) \subseteq \tau(F)$. By Lemma 3.3, it follows $\theta(\tau(G)) \subseteq \tau(F)$. Hence, $\tau(F) \succeq \tau(G)$. From $\theta(G) \subseteq F$ and $\theta(\tau(G)) \subseteq \tau(F)$, it follows $\theta(G \cup \tau(G)) \subseteq F \cup \tau(F)$. Hence, $F \cup \tau(F) \succeq G \cup \tau(G)$. □

COROLLARY 3.5. *Let F, G be tableaux and τ a tableau query. If $F \sim G$, then $\tau(F) \sim \tau(G)$ and $F \cup \tau(F) \sim G \cup \tau(G)$.*

LEMMA 3.6. *Let F, G be tableaux, and p, q symbols. Let θ be a substitution such that $\theta(p), \theta(q)$ are not two distinct constants. If $\theta(G) \subseteq F$, then $id_{\theta(p) = \theta(q)}(F) \succeq id_{p = q}(G)$.*

PROOF. Since $\theta(p)$ and $\theta(q)$ are not two distinct constants, p and q are not two distinct constants either. From the premise $\theta(G) \subseteq F$, it follows $id_{\theta(p) = \theta(q)}(\theta(G)) \subseteq id_{\theta(p) = \theta(q)}(F)$. It is easy to verify that $id_{\theta(p) = \theta(q)} \circ \theta = id_{\theta(p) = \theta(q)} \circ \theta \circ id_{p = q}$. Hence, $id_{\theta(p) = \theta(q)} \circ \theta(id_{p = q}(G)) \subseteq id_{\theta(p) = \theta(q)}(F)$. Hence, $id_{\theta(p) = \theta(q)}(F) \succeq id_{p = q}(G)$. □

THEOREM 3.7. *Equivalent tableaux satisfy the same full dependencies.*

PROOF. Let F, G be two tableaux such that $F \sim G$: $F \succeq G$ and $G \succeq F$. Assume $F \models \tau$ for some ftgd τ: $F \succeq F \cup \tau(F)$. By Lemma 3.4, $F \cup \tau(F) \succeq G \cup \tau(G)$. By transitivity, $G \succeq G \cup \tau(G)$. It follows $G \models \tau$.

Assume $F \models \epsilon$ for some fegd $\epsilon = (B, p = q)$. Since $F \succeq G$, we can assume a substitution θ such that $\theta(G) \subseteq F$. Let ω be any substitution with $\omega(B) \subseteq G$. Hence, $\theta \circ \omega(B) \subseteq F$. Since $F \models \epsilon$, $\theta \circ \omega(p)$ and $\theta \circ \omega(q)$ are not two distinct constants, and $F \succeq id_{\theta\circ\omega(p) = \theta\circ\omega(q)}(F)$. Clearly, $\omega(p)$ and $\omega(q)$ are not two distinct constants either. By Lemma 3.6, $id_{\theta\circ\omega(p) = \theta\circ\omega(q)}(F) \succeq id_{\omega(p) = \omega(q)}(G)$. Since $G \succeq F$, it follows $G \succeq id_{\omega(p) = \omega(q)}(G)$ by transitivity. Since $id_{\omega(p) = \omega(q)}(G) \succeq G$ is obvious, it follows $G \models \epsilon$. □

3.2 The Chase

The chase was originally introduced as a tool for deciding logical implication; here we use it for repairing databases. We generalize some results of Beeri and Vardi [1984] to tableaux that can contain constants and need not be typed, replacing equality of tableaux by equivalence. A consequence is that a chase can terminate "unsuccessfully" when it tries to identify two distinct constants. New results include decidability of subconsistency (Theorem 3.14), which will be needed for determining fixes.

Definition 3.8. We introduce an artificial top element, denoted □, to the quasi-order $\langle \mathfrak{T}, \succeq \rangle$. Let $F \neq \square$ and G be tableaux, and Σ a set of full dependencies. We write $F \vdash_\Sigma G$ if G can be obtained from F by a single application of one of the following *chase rules*:

(1) If $\tau = (B, h)$ is an ftgd of Σ, then $F \vdash_\Sigma F \cup \tau(F)$.
(2) Let $(B, p = q)$ be an fegd of Σ, and θ a substitution such that $\theta(B) \subseteq F$. If $\theta(p)$ and $\theta(q)$ are two distinct constants, then $F \vdash_\Sigma \square$; otherwise, $F \vdash_\Sigma id_{\theta(p) = \theta(q)}(F)$.

A *chase* of F by Σ is a maximal (with respect to length) sequence $F = F_0, F_1, \ldots, F_n$ of tableaux such that for every $i \in \{1, \ldots, n\}$, $F_{i-1} \vdash_\Sigma F_i$ and $F_{i-1} \neq F_i$.

Requiring that chases be maximal tacitly assumes that chases are finite, which is expressed by Lemma 3.9. This lemma and some of the lemmas to follow imitate some known results (see for example Lemmas 8.4.3 and 8.4.4 in Abiteboul et al. [1995]). Since our semantics are slightly different (distinct variables should not be treated as distinguished constants in general), the results are stated explicitly.

LEMMA 3.9. *Let $F \neq \square$ be a tableau and Σ a set of full dependencies.*

(1) *If G is a tableau in a chase of F by Σ, then $G \succeq F$.*
(2) *Each chase of F by Σ is finite.*
(3) *If $G \neq \square$ is the last element of a chase of F by Σ, then $G \models \Sigma$.*
(4) *If $G \neq \square$ is the last element of a chase of F by Σ, and θ is a substitution mapping distinct variables to distinct constants not occurring elsewhere, then $\theta(G) \models \Sigma$.*

PROOF. Easy. □

The following two lemmas are needed for the main theorems to follow, expressing that the chase is Church-Rosser up to $\sim$ and that subconsistency is decidable for full dependencies. Lemma 3.10 implies that all tableaux in a successful chase of F by Σ are homomorphic to each consistent tableau H satisfying $H \succeq F$.

LEMMA 3.10. *Let F, H be tableaux, both distinct from $\square$, and Σ a set of full dependencies. Let $G \neq \square$ be a tableau in a chase of F by Σ. If $H \succeq F$ and $H \models \Sigma$, then $H \succeq G$.*

PROOF. Assume $H \succeq F$ and $H \models \Sigma$. Let $F = F_0, F_1, \ldots, F_n = G \neq \square$ be the initial sequence of a chase of F by Σ. We prove by induction that for each $k \in \{0, \ldots, n\}$, $H \succeq F_k$. *Base:* $H \succeq F = F_0$ is given. *Step:* The induction hypothesis is $H \succeq F_{k-1}$. The tableau F_k can be obtained by an ftgd or an fegd:

(1) F_k is obtained from F_{k-1} by an application of $\tau = (B, h)$ of Σ; that is, $F_k = F_{k-1} \cup \tau(F_{k-1})$. Since $H \succeq F_{k-1}$, $H \cup \tau(H) \succeq F_{k-1} \cup \tau(F_{k-1})$ by Lemma 3.4. Since $H \models \tau$, $H \succeq H \cup \tau(H)$. It follows $H \succeq F_k$.
(2) F_k is obtained from F_{k-1} by an application of $\epsilon = (B, p = q)$ of Σ. As $G \neq \square$, $F_k \neq \square$. We can assume a substitution ψ such that $\psi(B) \subseteq F_{k-1}$, $\psi(p)$ and $\psi(q)$ are not distinct constants, and $F_k = id_{\psi(p) = \psi(q)}(F_{k-1})$. Since $H \succeq F_{k-1}$, we can assume a substitution θ such that $\theta(F_{k-1}) \subseteq H$. It follows $\theta(\psi(B)) \subseteq H$, and, since $H \models \epsilon$, $\theta \circ \psi(p)$ and $\theta \circ \psi(q)$ are not two distinct constants and $H \sim id_{\theta\circ\psi(p) = \theta\circ\psi(q)}(H)$. By Lemma 3.6, $id_{\theta\circ\psi(p) = \theta\circ\psi(q)}(H) \succeq id_{\psi(p) = \psi(q)}(F_{k-1})$. By transitivity, $H \succeq F_k$.

It is correct to conclude $H \succeq G$. □

LEMMA 3.11. *Let F, H be tableaux, both distinct from $\square$, and Σ a set of full dependencies. If $\square$ is the last element of a chase of F by Σ and $H \succeq F$, then $H \not\models \Sigma$.*

PROOF. Let G be the last but one element in the chase of F by Σ that ends with $\square$: $G \vdash_\Sigma \square$. Assume $H \succeq F$. If $H \not\succeq G$, then $H \not\models \Sigma$ by Lemma 3.10. Next assume $H \succeq G$, hence we can assume a substitution θ such that $\theta(G) \subseteq H$. Since $G \vdash_\Sigma \square$, Σ contains an fegd $\epsilon = (B, p = q)$ such that there exists a substitution ψ satisfying $\psi(B) \subseteq G$ and $\psi(p) \neq \psi(q)$ are distinct constants. Then $\theta \circ \psi(B) \subseteq H$, but $\theta \circ \psi(p) \neq \theta \circ \psi(q)$ are distinct constants. We conclude $H \not\models \epsilon$, hence $H \not\models \Sigma$. □

THEOREM 3.12. *Let $F \neq \square$ be a tableau and Σ a set of full dependencies. If two chases of F by Σ end with G_1 and G_2 respectively, then $G_1 \sim G_2$.*

PROOF. This is the Church-Rosser property. Assume $G_2 \neq \square$ and $G_1 \neq \square$. Since $G_2 \succeq F$ and $G_2 \models \Sigma$ by Lemma 3.9, $G_2 \succeq G_1$ by Lemma 3.10. Similarly, $G_1 \succeq G_2$. It follows $G_1 \sim G_2$.

Next assume $G_2 = \square$. Then $G_1 = \square$, or else $G_1 \succeq F$ and $G_1 \models \Sigma$ by Lemma 3.9, but $G_1 \not\models \Sigma$ by Lemma 3.11, a contradiction. □

Theorem 3.12 motivates the following definition.

Definition 3.13. Let $F \neq \square$ be a tableau and Σ a set of full dependencies. We write $\mathbf{chase}(F, \Sigma)$ for the equivalence class $[G]_\sim$ if the last element of a chase of F by Σ is G. The singleton $[\square]_\sim$ is also written $\square$.

Given a set Σ of full dependencies, it is decidable whether a given tableau F is subconsistent.

THEOREM 3.14. *Let Σ be a set of full dependencies and $F \neq \square$ a tableau. Then, F is subconsistent (i.e. $F \Vdash \Sigma$) if and only if $\mathbf{chase}(F, \Sigma) \neq \square$.*

PROOF. The *only-if part* is an immediate corollary of Lemma 3.11. The *if part* follows from Lemma 3.9. □

Since the empty tableau is homomorphic to any relation, Σ is satisfiable if and only if the empty tableau is subconsistent. Likewise, since the tableau $\{\langle x_1, x_2, \ldots, x_n\rangle\}$, where $x_1, x_2, \ldots, x_n$ are distinct constants, is homomorphic to each nonempty relation, Σ is coherent (Σ allows a nonempty relation) if and only if $\{\langle x_1, x_2, \ldots, x_n\rangle\}$ is subconsistent. So we obtain the following result:

COROLLARY 3.15. *Assume arity n and let $x_1, x_2, \ldots, x_n$ be distinct constants. Let Σ be a set of full dependencies. Then,*

(1) *Σ is satisfiable if and only if $\mathbf{chase}(\{\}, \Sigma) \neq \square$;*
(2) *Σ is coherent if and only if $\mathbf{chase}(\{\langle x_1, x_2, \ldots, x_n\rangle\}, \Sigma) \neq \square$.*

Finally, the chase preserves the order $\succeq$ on tableaux.

THEOREM 3.16. *Let F, G be tableaux, both distinct from $\square$, and Σ a set of full dependencies. If $F \succeq G$, $F' \in \mathbf{chase}(F, \Sigma)$, and $G' \in \mathbf{chase}(G, \Sigma)$, then $F' \succeq G'$.*

PROOF. Assume $F \succeq G$. The proof is trivial if $\mathbf{chase}(F, \Sigma) = \square$. Next assume that $\mathbf{chase}(F, \Sigma)$ is distinct from $\square$. Assume without loss of generality that $F' \neq \square$ is the last element in a chase of F by Σ. By Lemma 3.9, $F' \models \Sigma$ and $F' \succeq F$, hence $F' \succeq G$. Then $\mathbf{chase}(G, \Sigma) \neq \square$, or else $F' \not\models \Sigma$ by Lemma 3.11, a contradiction. Let $G' \neq \square$ be the last element in a chase of G by Σ. By Lemma 3.10, $F' \succeq G'$. □

COROLLARY 3.17. *Let F, G be tableaux, both distinct from $\square$, and Σ a set of full dependencies. If $F \sim G$, then $\mathbf{chase}(F, \Sigma) = \mathbf{chase}(G, \Sigma)$.*

3.3 Replacing Uprepairs by Chase Results

After our digression on the chase, we return to the problem raised at the beginning of Section 3: given a satisfiable set Σ of full dependencies, a query τ, and a finite set $\mathbf{F}$ of subconsistent tableaux, compute $\tau(\mathbf{F}^{\uparrow\Sigma})$. Recall that $\tau(\mathbf{F}^{\uparrow\Sigma})$ denotes the intersection of the answers to τ on all (possibly infinitely many) uprepairs generated by $\mathbf{F}$ and Σ. In the results to follow, $\mathbf{F}$ can be any set of subconsistent tableaux. Later on in Section 5, $\mathbf{F}$ will be restricted to be a set of fixes. Theorem 3.22 implies that in order to compute $\tau(\mathbf{F}^{\uparrow\Sigma})$, it suffices to query the tableaux obtained by chasing each $F \in \mathbf{F}$ by Σ.

For example, let $\mathbf{F} = \{F_1, \ldots, F_5\}$, the set of tableaux shown in Figure 2, and let $\Sigma = \{\tau_1, \epsilon_1, \epsilon_2, \epsilon_3\}$, the full dependencies in Figure 1. A chase of F_i by Σ ends

with T_i $(1 \leq i \leq 5)$. Then, Theorem 3.22 will tell us that for any tableau query τ, $\tau(\mathbf{F}^{\uparrow\Sigma})$ is equal to $\mathbf{grd}(\tau(T_1)) \cap \cdots \cap \mathbf{grd}(\tau(T_5))$.

The following shorthand notation will be used for convenience.

Definition 3.18. Let G be a tableau and $\mathbf{F}$ a set of tableaux. We write $\mathbf{F} \stackrel{*}{\succeq} G$ if and only if $F \succeq G$ for each $F \in \mathbf{F}$.

LEMMA 3.19. *Let Σ be a satisfiable set of full dependencies and F a subconsistent tableau. For each tableau H, $F^{\uparrow\Sigma} \stackrel{*}{\succeq} H$ if and only if* $\mathbf{chase}(F, \Sigma) \stackrel{*}{\succeq} H$.

PROOF. Note that since $F \Vdash \Sigma$, $\mathbf{chase}(F, \Sigma) \neq \square$ by Theorem 3.14. *If part.* Let $G \neq \square$ be the last element in a chase of F by Σ. Assume $G \succeq H$ and $J \in F^{\uparrow\Sigma}$. It suffices to show $J \succeq H$. Since $J \succeq F$ and $J \models \Sigma$, $J \succeq G$ by Lemma 3.10. By transitivity, $J \succeq H$. *Only-if part.* Assume $\mathbf{chase}(F, \Sigma) \stackrel{*}{\not\succeq} H$. Let G be the last element in a chase of F by Σ. From $\mathbf{chase}(F, \Sigma) \stackrel{*}{\not\succeq} H$, it follows $G \not\succeq H$. Let V be the set of variables occurring in G. Let $\theta : V \xrightarrow{inj} \mathbf{dom}$ be a substitution that maps distinct variables to distinct constants not in $\mathbf{asd}(G) \cup \mathbf{asd}(H)$. By Lemma 3.9, $\theta(G) \models \Sigma$. Assume $\theta(G) \succeq H$. Then, we can assume a substitution μ such that $\mu(H) \subseteq \theta(G)$, hence $\theta^{-1} \circ \mu(H) \subseteq G$. It follows $G \succeq H$, a contradiction. We conclude by contradiction $\theta(G) \not\succeq H$. From $\theta(G) \succeq G$ and $G \succeq F$ (Lemma 3.9), $\theta(G) \succeq F$ by transitivity. We can assume a minimal (with respect to $\subseteq$) relation J satisfying $\theta(G) \supseteq J \succeq F$ and $J \models \Sigma$. Then $J \in F^{\uparrow\Sigma}$ and $J \not\succeq H$. Hence, $F^{\uparrow\Sigma} \stackrel{*}{\not\succeq} H$. $\square$

Since all tableaux in $\mathbf{chase}(F, \Sigma)$ are equivalent, in order to verify $\mathbf{chase}(F, \Sigma) \stackrel{*}{\succeq} H$ in the preceding lemma (and the following theorem), it suffices to chase F by Σ and check whether H is homomorphic to the final tableau in the chase.

THEOREM 3.20. *Let Σ be a satisfiable set of full dependencies and $\mathbf{F}$ a finite set of subconsistent tableaux. For each tableau H, $\mathbf{F}^{\uparrow\Sigma} \stackrel{*}{\succeq} H$ if and only if for each $F \in \mathbf{F}$,* $\mathbf{chase}(F, \Sigma) \stackrel{*}{\succeq} H$.

PROOF. This follows immediately from Lemma 3.19. $\square$

LEMMA 3.21. *Let τ be a tableau query. Let Σ be a satisfiable set of full dependencies. Let F be a subconsistent tableau. For each $G \in \mathbf{chase}(F, \Sigma)$, $\tau(F^{\uparrow\Sigma}) = \mathbf{grd}(\tau(G))$.*

PROOF. Let $\tau = (B, h)$. Since $F \Vdash \Sigma$, $\mathbf{chase}(F, \Sigma) \neq \square$ by Theorem 3.14. Let $G \in \mathbf{chase}(F, \Sigma)$.

To prove the inclusion $\mathbf{grd}(\tau(G)) \subseteq \tau(F^{\uparrow\Sigma})$, assume $t \in \mathbf{grd}(\tau(G))$. Let $J \in F^{\uparrow\Sigma}$. Since $J \succeq F$ and $J \models \Sigma$, $J \succeq G$ by Lemma 3.10. By Lemma 3.4, $\tau(J) \succeq \tau(G)$. Since t is a ground tuple in $\tau(G)$, it follows $t \in \tau(J)$. Since J is an arbitrary element of $F^{\uparrow\Sigma}$, it follows $t \in \tau(F^{\uparrow\Sigma})$.

To prove the inclusion $\tau(F^{\uparrow\Sigma}) \subseteq \mathbf{grd}(\tau(G))$, let $t \in \tau(F^{\uparrow\Sigma})$. Let θ be a substitution for the symbols in h such that $\theta(h) = t$, extended to be the identity otherwise. Obviously, $t \in \tau(F^{\uparrow\Sigma})$ implies $F^{\uparrow\Sigma} \stackrel{*}{\succeq} \theta(B)$. By Lemma 3.19, $G \succeq \theta(B)$. Since $\theta(h)$ is ground, $\theta(h) = t \in \mathbf{grd}(\tau(G))$. $\square$

THEOREM 3.22. *Let τ be a tableau query. Let Σ be a satisfiable set of full dependencies and $\mathbf{F}$ a finite set of subconsistent tableaux. Let $\mathbf{G}$ be a set of tableaux such that $\mathbf{G} \sim \bigcup_{F \in \mathbf{F}} \mathbf{chase}(F, \Sigma)$. Then, $\tau(\mathbf{F}^{\uparrow\Sigma}) = \bigcap_{G \in \mathbf{G}} \mathbf{grd}(\tau(G))$.*

PROOF. This follows immediately from Lemma 3.21. □

In practice, the set $\mathbf{G}$ in the preceding theorem will be computed by collecting the terminal tableaux that result from chasing each tableau of $\mathbf{F}$ by Σ.

4. CONSTRUCTING THE NUCLEUS

Given a satisfiable set Σ of full dependencies, a tableau query τ, and a set $\mathbf{F}$ of subconsistent tableaux, Theorem 3.22 gives us an effective algorithm to compute $\tau(\mathbf{F}^{\uparrow\Sigma})$: chase each tableau of $\mathbf{F}$ by Σ, query the last tableau of each chase by τ, and return the ground tuples common to all query answers. We now show that $\mathbf{F}^{\uparrow\Sigma}$ has a nucleus: that there exists a *single* tableau G such that for *every* tableau query τ, $\tau(\mathbf{F}^{\uparrow\Sigma}) = \mathbf{grd}(\tau(G))$.

4.1 Nucleus Definition

A *nucleus* of a set $\mathbf{I}$ of relations is defined relative to a class of queries; intuitively, it is a single tableau that can replace $\mathbf{I}$ for the purpose of query answering.

Definition 4.1. Let $\mathbf{I}$ be a (possibly infinite) set of relations. Let Q be a subclass of CQ ($\mathsf{Q} \subseteq \mathsf{CQ}$). The tableau F is called a Q-*nucleus* of $\mathbf{I}$ if and only if for every query $\tau \in \mathsf{Q}$, $\mathbf{grd}(\tau(F)) = \tau(\mathbf{I})$. The term CQ-nucleus is often abbreviated as *nucleus*—CQ can be omitted.

In the preceding definition, the function $\mathbf{grd}(\cdot)$ serves to eliminate from $\tau(F)$ tuples that contain variables. The following theorem implies that a nucleus must be unique up to $\sim$.

THEOREM 4.2. *Let $\mathbf{I}$ be a set of relations. If G_1 and G_2 are CQ-nuclei of $\mathbf{I}$, then $G_1 \sim G_2$.*

PROOF. Let G_1 and G_2 be CQ-nuclei of $\mathbf{I}$. Consider the Boolean tableau query $\tau = (G_1, \langle 1 \rangle)$. Since $\tau(G_1) = \{\langle 1 \rangle\}$, we have $\tau(\mathbf{I}) = \{\langle 1 \rangle\}$, hence $\tau(G_2) = \{\langle 1 \rangle\}$. Hence, there exists a substitution θ that maps the body of τ into G_2: $\theta(G_1) \subseteq G_2$. It follows $G_2 \succeq G_1$. By the same reasoning, $G_1 \succeq G_2$. Hence, $G_1 \sim G_2$. □

4.2 Infimum

We define the greatest lower bound (or infimum) of a set of tableaux.

Definition 4.3. Let $\mathbf{F}$ be a finite set of tableaux. An *infimum* (with respect to $\succeq$) of $\mathbf{F}$ is a tableau G satisfying:

(1) $\mathbf{F} \overset{*}{\succeq} G$, and

(2) for every tableau G', $\mathbf{F} \overset{*}{\succeq} G'$ implies $G \succeq G'$.

It is easy to see that all infimums of a set $\mathbf{F}$ of tableaux must be mutually equivalent. We assume there is an arbitrary selection rule that picks a representative of this $\sim$-equivalence class and denote it $\inf \mathbf{F}$.

There is an effective procedure to construct an infimum of any finite set of tableaux; the construction is borrowed from infimums of sets of clauses [Plotkin 1969]. Practically, to construct an infimum of $\{F_1, F_2\}$, every tuple of F_1 is combined columnwise with every tuple of F_2. The combination of two symbols p and q, denoted $\mu(p,q)$, is a variable unless p and q are the same constant. The combination of two times the same constant (say a) yields that same constant: $\mu(a,a) = a$.

Definition 4.4. An *inf-mapping* is a partial one-one function $\mu : (\mathbf{dom} \cup \mathbf{var}) \times (\mathbf{dom} \cup \mathbf{var}) \xrightarrow{inj} \mathbf{dom} \cup \mathbf{var}$ such that $\mu(p,q) = p$ if p and q are the same constant, and $\mu(p,q)$ is a new distinct variable otherwise.

Let F, G be tableaux and μ an inf-mapping defined over $\mathbf{asd}(F) \times \mathbf{asd}(G)$. The inf-mapping μ naturally extends to pairs of tuples and pairs of tableaux as follows: firstly, if $t \in F$ and $t' \in G$, then

$$\mu(t,t') := \langle \mu(t(1), t'(1)), \ldots, \mu(t(n), t'(n)) \rangle;$$

secondly, $\mu(F,G) := \{\mu(t,t') \mid t \in F, t' \in G\}$.

For example, for the following tableaux F_1 and F_2,

F_1

1	2	3
a	b	z
b	c	z

F_2

1	2	3
a	y	y
y	c	y

,

we obtain:

$\mu(F_1, F_2)$

1	2	3
$\mu(a,a)$	$\mu(b,y)$	$\mu(z,y)$
$\mu(b,a)$	$\mu(c,y)$	$\mu(z,y)$
$\mu(a,y)$	$\mu(b,c)$	$\mu(z,y)$
$\mu(b,y)$	$\mu(c,c)$	$\mu(z,y)$

$\sim$

1	2	3
a	u	v
w_1	w_2	v
w_3	w_4	v
u	c	v

$\sim$ G

1	2	3
a	u	v
u	c	v

It is easy to verify that G is homomorphic to both F_1 and F_2. It takes more effort to show that every tableau homomorphic to both F_1 and F_2 is also homomorphic to G. We show that the above construction is correct in general.

THEOREM 4.5. *Let F_1, F_2 be tableaux. If μ is an inf-mapping defined over $\mathbf{asd}(F_1) \times \mathbf{asd}(F_2)$, then $\mu(F_1, F_2)$ is an infimum of $\{F_1, F_2\}$.*

PROOF. Let $G = \mu(F_1, F_2)$. Let λ and ρ be substitutions for the symbols in the range of μ such that for all symbols $(p,q) \in \mathbf{asd}(F_1) \times \mathbf{asd}(F_2)$, $\lambda(\mu(p,q)) = p$ and $\rho(\mu(p,q)) = q$. The mappings λ and ρ are the identity on constants because $\mu(p,q) = a \in \mathbf{dom}$ implies $p = q = a$; they are well-defined because $\mu(p_1,q_1) = \mu(p_2,q_2)$ implies $p_1 = p_2$ and $q_1 = q_2$. Clearly, $\lambda(G) = F_1$ and $\rho(G) = F_2$. It follows $F_1 \succeq G$ and $F_2 \succeq G$, hence $\{F_1, F_2\} \overset{*}{\succeq} G$.

Assume a tableau H homomorphic to both F_1 and F_2, $\{F_1, F_2\} \overset{*}{\succeq} H$; it suffices to show $G \succeq H$. We can assume substitutions θ and ω such that $\theta(H) \subseteq F_1$ and $\omega(H) \subseteq F_2$. Let γ be a total function with domain $\mathbf{asd}(H)$ defined by $\gamma(p) = \mu(\theta(p), \omega(p))$. Note that γ is the identity on constants. Let $t \in H$. It can be easily verified that $\gamma(t) = \mu(\theta(t), \omega(t))$. From $\theta(t) \in F_1$ and $\omega(t) \in F_2$, it follows $\gamma(t) \in G$. Since t is an arbitrary tuple of H, $\gamma(H) \subseteq G$. Hence, $G \succeq H$. □

The following theorem expresses that the infimum of a set of tableaux preserves consistency.

THEOREM 4.6. *Let F_1, F_2 be tableaux and Σ a set of full dependencies. Let G be an infimum of $\{F_1, F_2\}$. If $F_1 \models \Sigma$ and $F_2 \models \Sigma$, then $G \models \Sigma$.*

PROOF. Assume $F_1 \models \Sigma$ and $F_2 \models \Sigma$. Let $\tau = (B, h)$ be an ftgd in Σ. Since $F_1 \models \tau$, $F_1 \succeq F_1 \cup \tau(F_1)$. From $F_1 \succeq G$, it follows $F_1 \cup \tau(F_1) \succeq G \cup \tau(G)$ by Lemma 3.4. Hence, $F_1 \succeq G \cup \tau(G)$. Likewise, $F_2 \succeq G \cup \tau(G)$. Hence, $\{F_1, F_2\} \overset{*}{\succeq} G \cup \tau(G)$. It follows $G \succeq G \cup \tau(G)$, hence $G \models \tau$.

Next, let $\epsilon = (B, p = q)$ be an fegd in Σ. Let μ be an inf-mapping defined over $\mathbf{asd}(F_1) \times \mathbf{asd}(F_2)$. By Theorem 4.5, $G' := \mu(F_1, F_2)$ is an infimum of $\{F_1, F_2\}$. Let λ and ρ be substitutions for the symbols in the range of μ such that for all symbols $(m, o) \in \mathbf{asd}(F_1) \times \mathbf{asd}(F_2)$, $\lambda(\mu(m, o)) = m$ and $\rho(\mu(m, o)) = o$. Clearly, $\lambda(G') = F_1$ and $\rho(G') = F_2$. Let θ be any substitution with $\theta(B) \subseteq G'$. It follows $\lambda \circ \theta(B) \subseteq F_1$ and $\rho \circ \theta(B) \subseteq F_2$. To ease the notation, define $p' := \theta(p)$ and $q' := \theta(q)$. Since $F_1 \models \epsilon$, it follows $\lambda(p'), \lambda(q')$ are not two distinct constants; likewise, $\rho(p'), \rho(q')$ are not two distinct constants. Since λ and ρ are the identity on constants, p', q' cannot be two distinct constants either. Let $F_1' := id_{\lambda(p') = \lambda(q')}(F_1)$ and $F_2' := id_{\rho(p') = \rho(q')}(F_2)$. Since $F_1 \models \epsilon$ and $F_2 \models \epsilon$, $F_1 \sim F_1'$ and $F_2 \sim F_2'$. Let $H := \mu(F_1', F_2')$. Let ϕ be a substitution for the variables in the range of μ defined by $\phi(\mu(x, y)) = \mu(id_{\lambda(p') = \lambda(q')}(x), id_{\rho(p') = \rho(q')}(y))$. Since $\phi(p') = \phi(q')$ is now easily verifiable, it follows $\phi = \phi \circ id_{p' = q'}$. We show $\phi(id_{p' = q'}(G')) \subseteq H$. To this extent, let $r \in id_{p' = q'}(G')$. Hence, there exist $t \in F_1$ and $s \in F_2$ such that $r = id_{p' = q'}(\mu(t, s))$. Since $t \in F_1$ and $s \in F_2$, $\mu(id_{\lambda(p') = \lambda(q')}(t), id_{\rho(p') = \rho(q')}(s)) \in H$. From $\mu(id_{\lambda(p') = \lambda(q')}(t), id_{\rho(p') = \rho(q')}(s)) = \phi(\mu(t, s)) = \phi \circ id_{p' = q'}(\mu(t, s)) = \phi(r)$, it follows $\phi(r) \in H$. Since r is an arbitrary tuple of $id_{p' = q'}(G')$, $\phi(id_{p' = q'}(G')) \subseteq H$, hence $H \succeq id_{p' = q'}(G')$. By Theorem 4.5, H is an infimum of $\{F_1', F_2'\}$. Since $F_1 \sim F_1'$ and $F_2 \sim F_2'$, H is an infimum of $\{F_1, F_2\}$. It follows $G' \succeq H \succeq id_{p' = q'}(G')$, hence $G' \models \epsilon$. Since $G' \sim G$, $G \models \epsilon$ by Theorem 3.7. It is correct to conclude $G \models \Sigma$. □

COROLLARY 4.7. *Let Σ a set of full dependencies. Every infimum of a finite set of consistent tableaux is itself consistent.*

4.3 Commutative Diagram

The construction of nuclei relies on Theorem 4.8, which states that the following commutative diagram is correct:

$$\begin{array}{ccc} \{F_1, F_2\} & \xrightarrow{\inf} & \inf\{F_1, F_2\} \\ {\scriptstyle\tau}\downarrow & & \downarrow{\scriptstyle\tau} \\ \{\tau(F_1), \tau(F_2)\} & \xrightarrow[\inf]{} & \inf\{\tau(F_1), \tau(F_2)\} \sim \tau(\inf\{F_1, F_2\}) \end{array}$$

THEOREM 4.8. *Let F_1, F_2 be tableaux and $\tau = (B, h)$ a tableau query. Let G be an infimum of $\{F_1, F_2\}$, and H an infimum of $\{\tau(F_1), \tau(F_2)\}$. Then $\tau(G) \sim H$.*

PROOF. Obviously, $\mathbf{asd}(\tau(F_1)) \subseteq \mathbf{asd}(F_1) \cup \mathbf{asd}(\tau)$ and $\mathbf{asd}(\tau(F_2)) \subseteq \mathbf{asd}(F_2) \cup \mathbf{asd}(\tau)$. Let μ be an inf-mapping defined over $(\mathbf{asd}(F_1) \cup \mathbf{asd}(\tau)) \times$

($\mathbf{asd}(F_2) \cup \mathbf{asd}(\tau)$). By Theorem 4.5, $G' := \mu(F_1, F_2)$ is an infimum of $\{F_1, F_2\}$ and $H' := \mu(\tau(F_1), \tau(F_2))$ is an infimum of $\{\tau(F_1), \tau(F_2)\}$.

From $F_1 \succeq G'$ and $F_2 \succeq G'$, it follows $\tau(F_1) \succeq \tau(G')$ and $\tau(F_2) \succeq \tau(G')$ by Lemma 3.4. Hence, $\{\tau(F_1), \tau(F_2)\} \overset{*}{\succeq} \tau(G')$. It follows $H' \succeq \tau(G')$.

We next show $\tau(G') \succeq H'$. Let r be an arbitrary tuple in H'. We can assume the existence of $t \in \tau(F_1)$ and $s \in \tau(F_2)$ such that $r = \mu(t, s)$. Since $t \in \tau(F_1)$, we can assume a substitution θ such that $\theta(B) \subseteq F_1$ and $\theta(h) = t$. Likewise, there exists a substitution ω such that $\omega(B) \subseteq F_2$ and $\omega(h) = s$. Let δ be a total function with domain $\mathbf{asd}(\tau)$ such that $\delta(p) = \mu(\theta(p), \omega(p))$. Let b be an arbitrary tuple of B: $b \in B$. It can be easily verified that $\delta(b) = \mu(\theta(b), \omega(b))$. Since $\theta(b) \in F_1$ and $\omega(b) \in F_2$, it follows $\delta(b) \in G'$. Since b is an arbitrary tuple of B, $\delta(B) \subseteq G'$. Hence, $\delta(h) \in \tau(G')$. From $\delta(h) = \mu(\theta(h), \omega(h)) = \mu(t, s) = r$, it follows $r \in \tau(G')$. Since r is an arbitrary tuple of H', $H' \subseteq \tau(G')$, hence $\tau(G') \succeq H'$.

It follows $\tau(G') \sim H'$. Since an infimum is unique up to $\sim$, $G' \sim G$ and $H' \sim H$. By Corollary 3.5, $\tau(G') \sim \tau(G)$. By transitivity, $\tau(G) \sim H$. This concludes the proof. □

COROLLARY 4.9. *Let* $\mathbf{F}$ *be a finite set of tableaux and* $\tau = (B, h)$ *a tableau query. Let* G *be an infimum of* $\mathbf{F}$*, and* H *an infimum of* $\{\tau(F) \mid F \in \mathbf{F}\}$*. Then* $\tau(G) \sim H$.

PROOF. Immediately from Theorem 4.8 and the following obvious property: if $\mathbf{F}_1, \mathbf{F}_2$ are finite sets of tableaux, G_1 an infimum of $\mathbf{F}_1$, and G_2 an infimum of $\mathbf{F}_2$, then each infimum of $\{G_1, G_2\}$ is an infimum of $\mathbf{F}_1 \cup \mathbf{F}_2$. □

4.4 Nucleus Construction

Let Σ be a satisfiable set of full dependencies and $\mathbf{F}$ a finite set of subconsistent tableaux. We show that the set of uprepairs generated by $\mathbf{F}$ and Σ (i.e. $\mathbf{F}^{\uparrow\Sigma}$) has a computable nucleus. Moreover, this nucleus is consistent.

THEOREM 4.10. *Let* Σ *be a satisfiable set of full dependencies, and* $\mathbf{F}$ *a finite set of subconsistent tableaux. Let* $\mathbf{G}$ *be a set of tableaux such that* $\mathbf{G} \sim \bigcup_{F \in \mathbf{F}} \mathbf{chase}(F, \Sigma)$*. Then, every infimum of* $\mathbf{G}$ *is (i) consistent, and (ii) a* CQ*-nucleus of* $\mathbf{F}^{\uparrow\Sigma}$.

PROOF. Let G be an infimum of $\mathbf{G}$. Let τ be a tableau query. Let H be an infimum of $\{\tau(F) \mid F \in \mathbf{G}\}$. By Corollary 4.9, $H \sim \tau(G)$. By Lemma 2.4, $\mathbf{grd}(\tau(G)) = \mathbf{grd}(H)$. Obviously, $\mathbf{grd}(H) = \bigcap_{F \in \mathbf{G}} \mathbf{grd}(\tau(F))$. By Theorem 3.22, $\bigcap_{F \in \mathbf{G}} \mathbf{grd}(\tau(F)) = \tau(\mathbf{F}^{\uparrow\Sigma})$. It follows $\mathbf{grd}(\tau(G)) = \tau(\mathbf{F}^{\uparrow\Sigma})$. Since τ is an arbitrary tableau query, G is a CQ-nucleus of $\mathbf{F}^{\uparrow\Sigma}$.

If $F \in \mathbf{F}$, then the last tableau in a chase of F by Σ is consistent (Lemma 3.9). By Theorem 3.7, each tableau in $\mathbf{G}$ is consistent. By Corollary 4.7, G is consistent. □

For a satisfiable set Σ of full dependencies and a finite set $\mathbf{F}$ of subconsistent tableaux, Theorem 4.10 gives us an effective procedure for computing a CQ-nucleus of $\mathbf{F}^{\uparrow\Sigma}$: chase each F in $\mathbf{F}$ by Σ, and compute an infimum of the last tableaux of these chases.

5. FIXES AND CONSISTENT QUERY ANSWERING

In the preceding lemmas and theorems, we start from any finite set **F** of subconsistent tableaux. We will now apply these results in situations where **F** results from "fixing" a relation I. We also provide upper bounds for the complexity of fix checking and consistent query answering.

5.1 Fixes

Intuitively, a fix of I is a subconsistent tableau obtained from I by replacing erroneous values by distinct variables. The formalization relies on homomorphisms that do not identify distinct tuples.

Definition 5.1. Let F, G be tableaux. A *one-one homomorphism* from F to G is a substitution θ for the variables in F such that $\theta(F) \subseteq G$ and θ identifies no two distinct tuples of F—for all $t, t' \in F$, $t \neq t'$ implies $\theta(t) \neq \theta(t')$. If such a one-one homomorphism from F to G exists, then F is said to be *one-one homomorphic* to G, denoted $G \sqsupseteq F$. We write $F \simeq G$ if and only if $F \sqsupseteq G$ and $G \sqsupseteq F$. We write $G \sqsupset F$ if and only if $G \sqsupseteq F$ and $G \not\simeq F$.

We illustrate the difference between $\succeq$ and $\sqsupseteq$ by our running example (see Figures 1 and 2). The tableau $F_1 \cup F_2$ is subconsistent and homomorphic to CARS: CARS $\succeq F_1 \cup F_2$ (use the substitution $\{x/\text{sedan}, z/\text{upper}\}$). However, CARS $\not\sqsupseteq F_1 \cup F_2$ because every homomorphism from $F_1 \cup F_2$ to CARS must identify the first and the third tuple of $F_1 \cup F_2$. We require fixes to be one-one homomorphic to the relation that has to be repaired, in order to avoid double repairing of the same tuple.

$F_1 \cup F_2$

Model	Version	PriceRange
x	luxury	upper
sedan	standard	lower
sedan	luxury	z

The following properties are straightforward.

LEMMA 5.2. *Let* $F, G \in \mathfrak{T}$.

(1) *If* $F \supseteq G$, *then* $F \sqsupseteq G$. *If* $F \sqsupseteq G$, *then* $F \succeq G$.
(2) $\langle \mathfrak{T}, \sqsupseteq \rangle$ *is a quasi-order.*
(3) $\simeq$ *is an equivalence relation on* $\mathfrak{T}$.

PROOF. Easy. □

Unlike earlier work [Wijsen 2002], we disallow multiple occurrences of the same variable in a fix, in order to exclude some unnatural uprepairs illustrated by Figure 4. The relation EMP stores employee salaries, and the fegd ϵ_5 expresses that no employee has more than one salary. Clearly, EMP $\not\models \epsilon_5$. We have EMP $\sqsupseteq F$ and F subconsistent. Note incidentally that F is not homomorphic to the tableau obtained from F by substituting 195 for the second occurrence of y. The relation J is the uprepair generated by F and ϵ_5. This uprepair is counterintuitive, however, as there seems to be no reason to assume that An's salary is mistaken in EMP. Intuitively, the fact that Ed and An earn the same

ϵ_5

Name	Sal
x	y
x	z
$y = z$	

EMP

Name	Sal
Ed	195
Ed	205
An	195

F

Name	Sal
Ed	y
Ed	205
An	y

J

Name	Sal
Ed	205
An	205

Fig. 4.

salary in EMP is just a coincident not to be encoded in fixes, as is done in F through the double occurrence of y.

Definition 5.3. A tuple or tableau without multiple occurrences of the same variable is called *linear*.

Let Σ be a satisfiable set of constraints and I a relation. A *fix* of I and Σ is a linear tableau F satisfying:

(1) $I \sqsupseteq F$ and F is subconsistent;
(2) *Maximality:* for every linear tableau G, if $I \sqsupseteq G \sqsupset F$, then G is not subconsistent.

Let $\mathbf{F}$ be the set of fixes of I and Σ. We define:

$$I^{\downarrow\uparrow\Sigma} := \mathbf{F}^{\uparrow\Sigma},$$

the set of uprepairs generated by the fixes of I and Σ. If $J \in I^{\downarrow\uparrow\Sigma}$, then we also say that J is an uprepair generated by I and Σ. If τ is a tableau query, then $\tau(I^{\downarrow\uparrow\Sigma})$ is called the *consistent query answer* to τ on input I (relative to Σ).

Given a linear tableau F, it is generally sufficient to know the entries of F up to a renaming of variables; if this is the case, then every variable can be replaced by a placeholder • without loss of information. So each occurrence of • in a tableau stands for a distinct variable.

Intuitively, the use of $\downarrow\uparrow$ in $I^{\downarrow\uparrow\Sigma}$ indicates that in repairing a relation I, we first go down the homomorphism lattice to find fixes, then go up to find uprepairs. Recall that $\tau(I^{\downarrow\uparrow\Sigma})$ is the intersection of the answers to τ on each uprepair generated by I and Σ. If Σ is a satisfiable set of constraints, then the empty tableau is subconsistent, so the set of fixes is nonempty. Since the cardinality of each fix of I is bounded by $|I|$, there can be at most finitely many nonequivalent fixes. This does not mean, however, that determining fixes is feasible in general. Obviously, the singleton tableau $\{\langle x_1, x_2, \ldots, x_n\rangle\}$ is subconsistent (with respect to Σ) if and only if Σ has a nonempty model (Σ is coherent). Since the latter property is generally undecidable, there is no algorithm to determine whether a given tableau F is subconsistent. When we limit ourselves to full dependencies, we obtain the following result.

COROLLARY 5.4. *For every relation I and satisfiable set Σ of full dependencies, there exists a computable* CQ*-nucleus of $I^{\downarrow\uparrow\Sigma}$.*

PROOF. Let $\mathbf{F}$ be the set of all fixes of I and Σ, hence $I^{\downarrow\uparrow\Sigma} = \mathbf{F}^{\uparrow\Sigma}$. Since the number of nonequivalent fixes is finite, we can compute a finite set $\mathbf{F}'$ such that $\mathbf{F}' \sim \mathbf{F}$. By Lemma 2.6, $\mathbf{F}'^{\uparrow\Sigma} = \mathbf{F}^{\uparrow\Sigma}$. Since $I^{\downarrow\uparrow\Sigma} = \mathbf{F}'^{\uparrow\Sigma}$ and $\mathbf{F}'$ is finite, the desired result follows from Theorem 4.10. □

The introductory example of Section 1 can illustrate the nucleus computation. The tableaux F_1–F_5 are five fixes of CARS and $\Sigma = \{\tau_1, \epsilon_1, \epsilon_2, \epsilon_3\}$, and all other fixes are equivalent to one of F_1–F_5. The chases of these fixes result in T_1–T_5. The infimum G of $\{T_1, \ldots, T_5\}$, which is shown in Section 1, is a CQ-nucleus of $I^{\downarrow\uparrow\Sigma}$. Note incidentally that this nucleus is not linear.

5.2 Fix Checking

Let $\mathfrak{L}$ denote the set of linear tableaux, and $\mathfrak{R}$ the set of relations (all of fixed arity n). Let Σ be a satisfiable set of constraints. Fix checking is the complexity of the set:

$$\mathsf{FC}(\Sigma) := \{(I, F) \in \mathfrak{R} \times \mathfrak{L} \mid F \text{ is a fix of } I \text{ and } \Sigma\}.$$

We will show that fix checking is in **P** for full dependencies. We need the construct of transformation, which allows modifying tableaux by specifying the new symbol to be put at each entry. Transformations differ from substitutions in that two occurrences of the same symbol in a tableau need not be transformed into the same symbol. Also, constants may be transformed into variables.

Definition 5.5. Let F be a tableau of arity n. Each element of $F \times \{1, 2, \ldots, n\}$ is called an *entry* of F (or *F-entry*). A *transformation* of F is a total function

$$\Delta : F \times \{1, 2, \ldots, n\} \to \mathbf{dom} \cup \mathbf{var}$$

A transformation Δ of F is applied to F in a natural way as follows. For each $t \in F$:

$$\Delta(t) := \langle \Delta(t, 1), \Delta(t, 2), \ldots, \Delta(t, n) \rangle.$$

Finally,

$$\Delta(F) := \{\Delta(t) \mid t \in F\}.$$

A special transformation of a relation I results from listing the entries of I that have to be replaced by new distinct variables.

Definition 5.6. Let I be a relation of arity n. Every set E of I-entries induces a transformation of I, denoted $\Delta^E : I \times \{1, 2, \ldots, n\} \to \mathbf{dom} \cup \mathbf{var}$, defined as follows: for each $t \in I$, $\Delta^E(t, i) = t(i)$ if $(t, i) \notin E$; otherwise $\Delta^E(t, i)$ is a new distinct variable not occurring elsewhere.

For example, let $I = \{s, t\}$ be as shown next. Let $E = \{(s, 2), (t, 1), (t, 2)\}$. Since $\Delta^E(I)$ is obviously a linear tableau, variables can be shown as • by our notational convention.

I	1	2	3	
	a	b	c	(s)
	b	c	d	(t)

$\Delta^E(I)$	1	2	3
	a	•	c
	•	•	d

Clearly, for every relation I and set E of I-entries, we have $\Delta^{\{\}}(I) = I$, $I \sqsupseteq \Delta^E(I)$, and $|I| = |\Delta^E(I)|$. The following lemma states that all fixes of a relation I and a coherent set Σ of constraints (Σ has a nonempty model) can be obtained by replacing certain occurrences of constants in I by distinct variables.

LEMMA 5.7. *Assume arity n. Let Σ be a coherent set of constraints. Let I be a relation and F a linear tableau.*

(1) *If F is a fix of I and Σ, then $|F| = |I|$.*
(2) *For every fix F of I and Σ, there exists a set E of I-entries such that $F \simeq \Delta^E(I)$.*

PROOF. For (1), let F be a fix of I and Σ. Clearly, $|F| \leq |I|$, or else $I \not\supseteq F$, a contradiction. Next assume $|F| < |I|$. Since Σ is coherent, $\{\}$ is not a fix, hence $|F| \geq 1$. Let $t = \langle x_1, x_2 \ldots, x_n \rangle$ where $x_1, \ldots, x_n$ are new distinct variables not occurring elsewhere. From $F \sim F \cup \{t\}$ and F subconsistent, it follows $F \cup \{t\}$ subconsistent. Since $I \sqsupseteq F \cup \{t\} \sqsupset F$ is obvious, F is not a fix, a contradiction. We conclude by contradiction that $|F| = |I|$.

For (2), let F be a fix of I and Σ. Since $I \sqsupseteq F$ and $|F| = |I|$, we can assume a substitution θ such that $\theta(F) = I$. Let E be the set of I-entries such that E contains $(\theta(t), i)$ if $t \in F$ and $t(i) \in \mathbf{var}$. Obviously, $\Delta^E(I) \simeq F$. □

LEMMA 5.8. *Let I be a relation and F a linear tableau such that $|I| = |F|$. It can be decided in polynomial time whether $I \sqsupseteq F$.*

PROOF. Define $\mathbf{S} := \{(s, t) \in I \times F \mid \{s\} \succeq \{t\}\}$. Clearly, $I \sqsupseteq F$ if and only if there exists $\mathbf{T} \subseteq \mathbf{S}$ such that $|\mathbf{T}| = |I|$ and no two distinct pairs of $\mathbf{T}$ agree on the first or the second coordinate. The latter problem is a two-dimensional matching problem, known to be in $\mathbf{P}$. □

THEOREM 5.9. *For a satisfiable set Σ of full dependencies, $\mathsf{FC}(\Sigma)$ is in $\mathbf{P}$.*

PROOF. Let I be a relation and F a linear tableau. If Σ is not coherent, then $(I, F) \in \mathsf{FC}(\Sigma)$ if and only if $F = \{\}$. If Σ is coherent, then $(I, F) \in \mathsf{FC}(\Sigma)$ if and only if the following three conditions are simultaneously satisfied:

(1) $|I| = |F|$ and $I \sqsupseteq F$. This can be tested in polynomial time by Lemma 5.8.
(2) F is subconsistent. By Theorem 3.14, this can be tested by the chase, which is known to have polynomial time data complexity [Beeri and Vardi 1984].
(3) If x is a variable occurring in the ith column of F, and a a constant occurring in the ith column of I, then either $I \not\supseteq id_{x=a}(F)$ or $id_{x=a}(F)$ is not subconsistent. For arity n, there are at most $n|I|^2$ possible substitutions of the form $id_{x=a}$.

We show that the condition (3) is sufficient to guarantee maximality (with respect to $\sqsupseteq$) of F. Assume that F satisfies conditions (1) and (2), but F is not a fix. Then, there is a linear tableau G such that $I \sqsupseteq G \sqsupset F$ and G is subconsistent. We can assume a substitution θ such that $\theta(F) = G$ and $|\theta(F)| = |F|$. Since G is linear, θ cannot map distinct variables of F to the same variable in G. So θ can only rename variables or replace variables by constants. Since $G \not\simeq F$, for some variable x in F, for some constant a, $\theta(x) = a$. Obviously, $I \sqsupseteq G \sqsupseteq id_{x=a}(F) \sqsupset F$, and since G is subconsistent, $id_{x=a}(F)$ is subconsistent. So we showed that if F satisfies conditions (1) and (2), but F is not a fix, then for some variable x in F, for some constant a, $I \sqsupseteq id_{x=a}(F)$ and $id_{x=a}(F)$ subconsistent. □

5.3 Consistent Query Answering

The problem of consistent query answering is defined relative to a fixed set Σ of constraints and a fixed tableau query $\tau = (B, h)$, where B and all tuples in Σ have the same fixed arity n. The problem is to decide membership of the set (recall that $\mathfrak{R}$ is the set of all relations):

$$\mathsf{CQA}(\Sigma, \tau) := \{I \in \mathfrak{R} \mid \tau(I^{\downarrow\uparrow\Sigma}) = \{\}\ \}.$$

That is, on input of a relation I of fixed arity, we have to decide whether the intersection of the answers to τ on all uprepairs is empty. A difference with the problem called *consistent query answers* in Chomicki and Marcinkowski [2005a], is that our definition is not restricted to Boolean queries. We show that consistent query answering is in **NP** for full dependencies and tableau queries.

THEOREM 5.10. *For a satisfiable set Σ of full dependencies and a Boolean tableau query τ,* $\mathsf{CQA}(\Sigma, \tau)$ *is in* **NP**.

PROOF. Let I be a relation of arity n. If Σ is not coherent, then $I^{\downarrow\uparrow\Sigma} = \{\{\}\}$, so it suffices to verify whether $\tau(\{\}) = \{\}$. Next assume Σ coherent. Since τ is Boolean, it follows from Lemma 2.6 and Theorem 3.22 that $\tau(I^{\downarrow\uparrow\Sigma}) = \{\}$ if and only if for some fix F of I and Σ, we have $\tau(G) = \{\}$ where G is the last tableau of a chase of F by Σ. From Lemma 5.7, it follows that every fix to consider can be encoded as a set E of I-entries, whose size is bounded by $n|I|$.

A nondeterministic polynomial time algorithm can guess some set E of I-entries, check in polynomial time whether $\Delta^E(I)$ is a fix (Theorem 5.9), and if so, compute the last tableau (call it G) of a chase of $\Delta^E(I)$ by Σ, and verify whether $\tau(G) = \{\}$. □

COROLLARY 5.11. *For a satisfiable set Σ of full dependencies and a tableau query τ,* $\mathsf{CQA}(\Sigma, \tau)$ *is in* **NP**.

PROOF. Assume I a relation of arity n. Let l be the length of Σ. Let $\tau = (B, h)$. Assume $\tau(I^{\downarrow\uparrow\Sigma}) \neq \{\}$; we can assume a ground tuple t such that $t \in \tau(I^{\downarrow\uparrow\Sigma})$. Clearly, all constants that occur in t but not in h must come from I or Σ. Let V be the set of variables that occur in h, and C the set of constants that occur in I or Σ. Then, for some substitution $\theta : V \to C$, extended to be the identity otherwise, $\theta(h) = t$. Let $\tau_\theta = (\theta(B), \theta(h))$, a Boolean tableau query. Clearly, $t \in \tau(I^{\downarrow\uparrow\Sigma})$ if and only if $t \in \tau_\theta(I^{\downarrow\uparrow\Sigma})$. There are at most $(n|I| + l)^{|V|}$ possibilities for θ, a number that is polynomial in $|I|$. Consequently, to decide whether $I \in \mathsf{CQA}(\Sigma, \tau)$, it suffices to decide $I \in \mathsf{CQA}(\Sigma, \tau_\theta)$ for polynomially many Boolean queries τ_θ. The desired result then follows from Theorem 5.10. □

6. NUCLEUS FOR UPDATE-BASED REPAIRING

Let Σ be a satisfiable set of full dependencies. On input of a relation I, we can effectively compute a CQ-nucleus of $I^{\downarrow\uparrow\Sigma}$ (Corollary 5.4). This nucleus allows us to compute consistent answers to any tableau query without query rewriting. However, the nucleus may not be practical because its construction takes exponential time, or even worse, its size is exponential. We show next that the

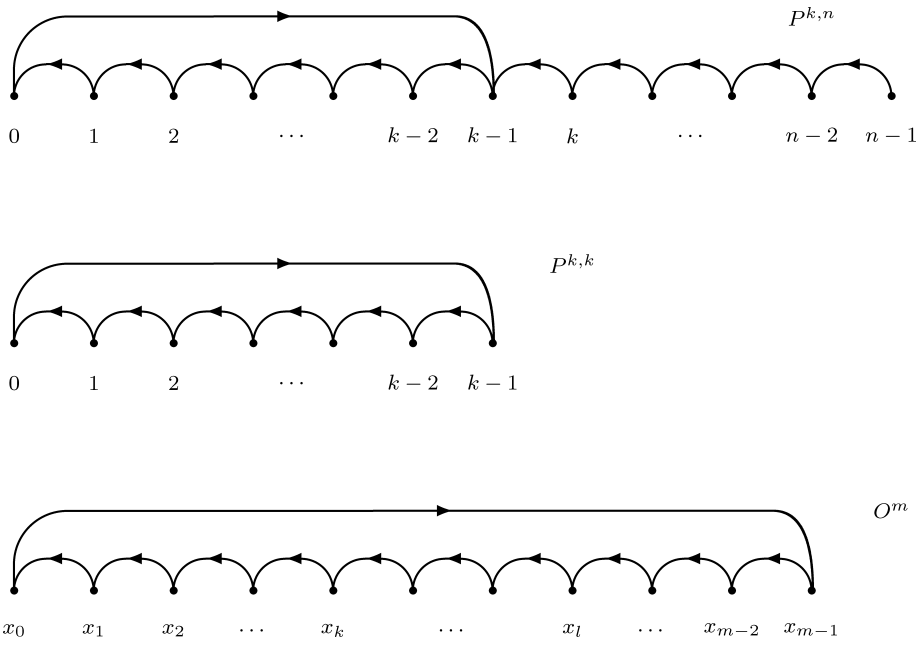


Fig. 5. Graphical representation of $P^{k,n}$, $P^{k,k}$, and O^m, $n \geq k > 1$, $m \geq 1$.

worst scenario cannot be ruled out. Clearly, if the complexity of CQA(Σ, τ) is **NP**-complete for some tableau query τ, then unless $\mathbf{P} = \mathbf{NP}$, the construction of a CQ-nucleus takes exponential time. In Sections 6.2 and 6.3, we introduce restricted query classes that allow nuclei of polynomial size, and we provide sufficient conditions under which these nuclei can be computed in polynomial time. These conditions will be used for identifying practical cases in Sections 7 and 8.

6.1 Nucleus of Exponential Size

We show that for full dependencies and conjunctive queries, the size of nuclei may not be polynomially bounded (Theorem 6.6).

Definition 6.1. For $n \geq k > 1$, define $P^{k,n}$ as the following relation:

$$P^{k,n} := \{\langle b, b, 0, k-1\rangle\} \cup \{\langle b, b, i+1, i\rangle \mid 0 \leq i \leq n-2\}.$$

For $m \geq 1$, define O^m as the following tableau:

$$O^m := \{\langle b, b, x_0, x_{m-1}\rangle\} \cup \{\langle b, b, x_{i+1}, x_i\rangle \mid 0 \leq i \leq m-2\}.$$

Since the constructs are technical, we give a graphical interpretation. Ignoring the first and the second column, which are filled by the constant b, the third and the fourth column can be represented as a graph. In particular, a tuple $\langle b, b, \alpha, \beta\rangle$ encodes an edge from node α to node β. The graph representations of $P^{k,n}$, $P^{k,k}$ and O^m are shown in Figure 5.

Obviously, if $m = k$, then O^m is homomorphic to $P^{k,n}$: simply use the substitution θ defined as $\theta(x_j) = j$, $j \in \{0, 1, \ldots, m-1\}$. The following lemma states

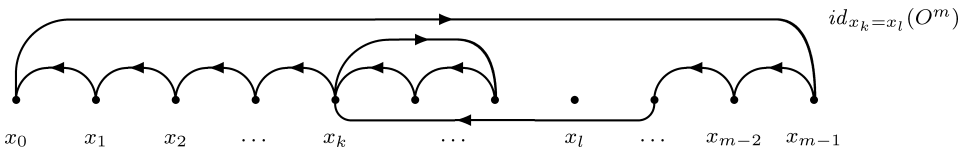


Fig. 6. Graphical representation of $id_{x_k = x_l}(O^m)$, illustrating Lemma 6.3.

that a necessary and sufficient condition for O^m to be homomorphic to $P^{k,n}$ is that m is a multiple of k.

LEMMA 6.2. *Let $n \geq k > 1$ and $m \geq 1$. Then $P^{k,n} \succeq O^m$ if and only if m is a multiple of k.*

PROOF. It can be easily verified that $O^m = \{\langle b, b, x_{(j+1) \bmod m}, x_{j \bmod m}\rangle \mid j \in \mathbb{N}_0\}$, where $\mathbb{N}_0 = \{0, 1, 2, \dots\}$. *If part.* Assume m is a multiple of k. Let θ be a substitution such that for each $j \in \{0, 1, \dots, m-1\}$, $\theta(x_j) = j \bmod k$. Let $t = \langle b, b, x_{(i+1) \bmod m}, x_{i \bmod m}\rangle$ be an arbitrary tuple of O^m ($i \in \mathbb{N}_0$). Then

$$\theta(t) = \langle b, b, ((i+1) \bmod m)) \bmod k, (i \bmod m) \bmod k\rangle.$$

Since m is a multiple of k,

$$\theta(t) = \langle b, b, (i+1) \bmod k, i \bmod k\rangle \in P^{k,n}.$$

It is correct to conclude $\theta(O^m) \subseteq P^{k,n}$, hence $P^{k,n} \succeq O^m$.
Only if part. Assume $P^{k,n} \succeq O^m$. Let θ be a substitution for $x_0, x_1, \dots, x_{m-1}$ such that $\theta(O^m) \subseteq P^{k,n}$. Assume for some $j \in \{0, 1, \dots, m-1\}$, $\theta(x_j) = l \geq k$. Then necessarily $\theta(x_{(j+1) \bmod m}) = l+1$, $\theta(x_{(j+2) \bmod m}) = l+2, \dots$ Eventually, $\theta(x_{p \bmod m}) = n-1$ for some $p \in \mathbb{N}_0$. Let $t = \langle b, b, x_{(p+1) \bmod m}, x_{p \bmod m}\rangle \in O^m$. Since $\theta(x_{p \bmod m}) = n-1$ does not occur in the rightmost column of $P^{k,n}$, $\theta(t) \notin P^{k,n}$, a contradiction. We conclude by contradiction that for each $j \in \{0, 1, \dots, m-1\}$, $\theta(x_j) < k$. That is, $\theta(O^m) \subseteq P^{k,k}$. Then θ is essentially a homomorphism from a cycle of length m to a cycle of length k. By the main result in Hell and Zhu [1995], it follows that m is a multiple of k. □

The following lemma states that identifying two distinct variables in O^m creates a cycle of smaller length. The graphical representation is shown in Figure 6.

LEMMA 6.3. *Let $m > l > k \geq 0$. There exists an integer p such that $m > p \geq 1$ and $id_{x_k = x_l}(O^m) \succeq O^p$.*

PROOF. Let $p = l - k$. Clearly, $m > p \geq 1$. Let θ be a substitution for $x_0, x_1, \dots, x_{p-1}$ such that $\theta(x_j) = x_{j+k}$ ($j \in \{0, 1, \dots, p-1\}$). Then $id_{x_k = x_l} \circ \theta(O^p) \subseteq id_{x_k = x_l}(O^m)$, hence $id_{x_k = x_l}(O^m) \succeq O^p$. □

In broad outline, the remainder goes as follows. We choose a database D^n of size $n > 1$ and a set Σ of full dependencies such that, first, each uprepair contains $P^{k,n}$ for some k with $n \geq k > 1$ (Lemma 6.5, first item), and second, for each k, there is an uprepair that contains exactly $P^{k,n}$ (Lemma 6.5, second item). We know that a nucleus of $(D^n)^{\downarrow\uparrow\Sigma}$ exists (Corollary 5.4), and that this nucleus is unique up to $\sim$ (Theorem 4.2). We then use Lemma 6.2 to show that some nucleus contains the tableau O^m with m a multiple of each positive integer k satisfying $n \geq k$, and we rely on Lemma 6.3 to show that this nucleus is reduced.

The desired result then follows from the fact that the least common multiple of the first n positive integers is greater than 2^n if $n \geq 7$.

Definition 6.4. For each $n > 1$, define D^n as follows:

D^n	1	2	3	4
	a	$1-n$	$n-1$	$n-2$
	a	$2-n$	$n-2$	$n-3$
			$\vdots$	
	a	-2	2	1
	a	-1	1	0

The second column contains negative values only; the third and the fourth columns contain nonnegative values only. Consider the following full dependencies:

ϵ_6	1	2	3	4
	x	y	u_1	v_1
	x	z	u_2	v_2
	$y = z$			

τ_3	1	2	3	4
	a	u	x	v
	b	b	0	x

τ_4	1	2	3	4
	u	v	x	y
	b	b	x	y

LEMMA 6.5. *Let* $\Sigma = \{\epsilon_6, \tau_3, \tau_4\}$ *and* $n > 1$.

(1) *If* F *is a fix of* D^n *and* Σ*, and* $F' \in \mathbf{chase}(F, \Sigma)$*, then for some* k *with* $n \geq k > 1$*,* $F' \geq P^{k,n}$.

(2) *For every* k *with* $n \geq k > 1$*, there exists a fix* F *of* D^n *and* Σ *such that for each* $F' \in \mathbf{chase}(F, \Sigma)$*,* $P^{k,n} = \{t \in F' \mid t(1) = t(2) = b\}$.

PROOF. It is easy to see that each fix is of the form F shown below, for some permutation $\langle i_1, i_2, \ldots, i_{n-1}\rangle$ of $\langle 1, 2, \ldots, n-1\rangle$, and for some l such that $1 \leq l \leq n-1$. A chase of F by Σ ends with F'.

F	1	2	3	4
	•	$-i_{n-1}$	i_{n-1}	$i_{n-1}-1$
			$\vdots$	
	•	$-i_{l+2}$	i_{l+2}	$i_{l+2}-1$
	•	$-i_{l+1}$	i_{l+1}	$i_{l+1}-1$
	a	•	i_l	i_l-1
			$\vdots$	
	a	•	i_3	i_3-1
	a	•	i_2	i_2-1
	a	$-i_1$	i_1	i_1-1

F'	1	2	3	4
	•	$-i_{n-1}$	i_{n-1}	$i_{n-1}-1$
			$\vdots$	
	•	$-i_{l+1}$	i_{l+1}	$i_{l+1}-1$
	a	$-i_1$	i_l	i_l-1
			$\vdots$	
	a	$-i_1$	i_1	i_1-1
	b	b	$n-1$	$n-2$
			$\vdots$	
	b	b	2	1
	b	b	1	0
	b	b	0	i_l
			$\vdots$	
	b	b	0	i_2
	b	b	0	i_1

The first part of the proof follows because $l \geq 1$. The second part is obtained by choosing $i_1 = k - 1$ and $l = 1$. □

THEOREM 6.6. *Let* $\Sigma = \{\epsilon_6, \tau_3, \tau_4\}$ *and* $n \geq 7$. *Every* CQ*-nucleus of* $(D^n)^{\downarrow\uparrow\Sigma}$ *contains at least* 2^n *tuples.*

PROOF. Let G be a CQ-nucleus of $(D^n)^{\downarrow\uparrow\Sigma}$. By Theorems 4.10 and 4.2, G is $\sim$-equivalent to an infimum of the set of terminal tableaux that result from chasing each fix of D^n and Σ by Σ. Let $m = \text{lcm}(2, 3, \ldots, n)$, where $\text{lcm}(\cdot)$ denotes the least common multiple.

We first show $G \succeq O^m$. Let F be a fix of D^n and Σ. Let $F' \in \mathbf{chase}(F, \Sigma)$. By Lemma 6.5 (first item), $F' \succeq P^{k,n}$ for some k with $n \geq k > 1$. Since m is a multiple of k, $P^{k,n} \succeq O^m$ by Lemma 6.2. Hence, $F' \succeq O^m$. Since F is an arbitrary fix, it follows $G \succeq O^m$.

Since $G \succeq O^m$, we can assume the existence of a substitution θ such that $\theta(O^m) \subseteq G$. We show that θ is injective. Assume θ identifies two variables of O^m, say $x_{k'}$ and x_l with $m > l > k' \geq 0$. Since $\theta = \theta \circ id_{x_{k'} = x_l}$ is obvious, $\theta(id_{x_{k'} = x_l}(O^m)) \subseteq G$, so $G \succeq id_{x_{k'} = x_l}(O^m)$. By Lemma 6.3, we can assume the existence of some p with $m > p \geq 1$ such that $G \succeq O^p$. Let $k \in \{2, 3, \ldots, n\}$. By Lemma 6.5 (second item), we can assume the existence of a fix F such that a chase of F by Σ results in a tableau F' with $\{t \in F' \mid t(1) = t(2) = b\} = P^{k,n}$. Since $F' \succeq G$ and $G \succeq O^p$, it follows $F' \succeq O^p$. Hence, there exists a substitution ω such that $\omega(O^p) \subseteq F'$. Moreover, since $t(1) = t(2) = b$ for every tuple $t \in O^p$, it follows $\omega(O^p) \subseteq P^{k,n}$, hence $P^{k,n} \succeq O^p$. By Lemma 6.2, p is a multiple of k. Since k is an arbitrary element of $\{2, 3, \ldots, n\}$, p is a multiple of $2, 3, \ldots, n$. Since $p < m$, $m \neq \text{lcm}(2, 3, \ldots, n)$, a contradiction. We conclude by contradiction that θ is injective, hence $m = |O^m| = |\theta(O^m)| \leq |G|$. Since it is known [Nair 1982a; Nair 1982b] that $\text{lcm}(1, 2, 3, \ldots, n) \geq 2^n$ if $n \geq 7$, $|G| \geq 2^n$. This concludes the proof. □

6.2 Linear Tableau Queries

We now turn to query classes that allow nuclei of polynomial size, and we give sufficient conditions under which these nuclei can be computed in polynomial time. The query classes are obtained from CQ by limiting the number of occurrences of quantified variables.

Definition 6.7. A tableau query $\tau = (B, h)$ is said to be *linear* if every variable that occurs more than once in B also occurs in h. The class of linear tableau queries is denoted linCQ.

Note that (B, h) linear does *not* imply B linear. For example, the following tableau query τ asking for models that exist in both upper and bottom priced versions, is linear:

τ

Model	Version	PriceRange
x	y	upper
x	z	bottom
x		

We now show that for any relation I and satisfiable set Σ of full dependencies, $I^{\downarrow\uparrow\Sigma}$ has a linCQ-nucleus the size of which is polynomially bounded in $|I|$. The proof relies on linearizations of tableaux, which are transformations that copy constants through and replace each variable occurrence by a new distinct variable.

Definition 6.8. Let F be a tableau of arity n. A *linearization* of F is a transformation Δ^{lin} of F such that $\Delta^{\mathsf{lin}}(t, j) = t(j)$ if $t(j) \in \mathbf{dom}$; otherwise $\Delta^{\mathsf{lin}}(t, j)$ is a new distinct variable not occurring elsewhere ($t \in F$, $j \in \{1, 2, \ldots, n\}$).

Obviously, if Δ^{lin} is a linearization of F, then $\Delta^{\mathsf{lin}}(F)$ is a linear tableau. For example,

F	1	2
	a	x
	b	x
	x	x

$\Delta^{\mathsf{lin}}(F)$	1	2
	a	y
	b	z
	v	w

$\Delta^{\mathsf{lin}}(F)$	1	2
	a	•
	b	•
	•	•

The linearization Δ^{lin} used is defined by $\Delta^{\mathsf{lin}}(\langle a, x\rangle, 1) = a$, $\Delta^{\mathsf{lin}}(\langle a, x\rangle, 2) = y$, $\Delta^{\mathsf{lin}}(\langle b, x\rangle, 1) = b$, $\Delta^{\mathsf{lin}}(\langle b, x\rangle, 2) = z$, $\Delta^{\mathsf{lin}}(\langle x, x\rangle, 1) = v$, and $\Delta^{\mathsf{lin}}(\langle x, x\rangle, 2) = w$. Since $\Delta^{\mathsf{lin}}(F)$ is linear, variables can be replaced by the placeholder • without loss of information.

Note that since no variable occurs more than once in a linear tableau G, an equivalent reduced tableau can be computed in quadratic time. The computation starts from G and repeatedly removes some tuple t if there is a tuple s left such that $\{s\} \succeq \{t\}$. In the above example, $\Delta^{\mathsf{lin}}(F)$ can be equivalently reduced by deleting its last tuple.

We show that the ground tuples in the answer to a linear tableau query are not affected by linearization.

LEMMA 6.9. *Let F be a tableau and Δ^{lin} a linearization of F. For every linear tableau query τ, $\mathbf{grd}(\tau(F)) = \mathbf{grd}(\tau(\Delta^{\mathsf{lin}}(F)))$.*

PROOF. Let $\tau = (B, h)$ be a linear tableau query. It suffices to show $\mathbf{grd}(\tau(F)) \subseteq \tau(\Delta^{\mathsf{lin}}(F))$ and $\mathbf{grd}(\tau(\Delta^{\mathsf{lin}}(F))) \subseteq \tau(F)$.

Since $F \succeq \Delta^{\mathsf{lin}}(F)$ is obvious, $\tau(F) \succeq \tau(\Delta^{\mathsf{lin}}(F))$ by Lemma 3.4. By Lemma 2.4, $\mathbf{grd}(\tau(\Delta^{\mathsf{lin}}(F))) \subseteq \tau(F)$.

For the inclusion $\mathbf{grd}(\tau(F)) \subseteq \tau(\Delta^{\mathsf{lin}}(F))$, assume $t \in \mathbf{grd}(\tau(F))$. Hence, there exists a substitution θ such that $\theta(B) \subseteq F$ and $\theta(h) = t$. Let δ be a substitution for the variables of B such that for every tuple $s \in B$, $\delta(s) = \Delta^{\mathsf{lin}}(\theta(s))$; $\delta(s(i)) = \Delta^{\mathsf{lin}}(\theta(s), i)$ for each $i \in \{1, 2, \ldots, n\}$.

To prove that δ is well-defined, we have two show two things: (i) that δ is the identity on constants, and (ii) that δ is never required to map two occurrences of the same variable to distinct symbols. Proving (i) is straightforward; for (ii), let t_1, t_2 be (not necessarily distinct) tuples of B and let $i, j \in \{1, \ldots, n\}$ such that $t_1(i) = t_2(j) \in \mathbf{var}$, and either $i \neq j$ or $t_1 \neq t_2$. Assume $t_1(i) = t_2(j) = x$ without loss of generality. That is, the same variable x occurs more than once in B. The definition of δ imposes $\delta(x) = \Delta^{\mathsf{lin}}(\theta(t_1), i)$ and $\delta(x) = \Delta^{\mathsf{lin}}(\theta(t_2), j)$, so we must show $\Delta^{\mathsf{lin}}(\theta(t_1), i) = \Delta^{\mathsf{lin}}(\theta(t_2), j)$. Since τ is linear, x occurs in h. Then $\theta(x)$ must be a constant, or else $\theta(h)$ would not be ground, a contradiction.

Since $\theta(x)$ is a constant and the linearization Δ^{lin} copies constants through, $\Delta^{\mathsf{lin}}(\theta(t_1), i) = \Delta^{\mathsf{lin}}(\theta(t_2), j) = \theta(x)$. It is correct to conclude that δ is well-defined.

We show that $\delta(h) = \theta(h)$: $\delta(h(i)) = \theta(h(i))$ for each $i \in \{1, \ldots, n\}$. Two cases can occur:

—$h(i)$ is a constant. Since δ and θ are the identity on constants, $\delta(h(i)) = \theta(h(i)) = h(i)$.

—$h(i)$ is variable. Since $h(i)$ must occur in B, we can assume $s \in B$ and $j \in \{1, \ldots, n\}$ such that $s(j) = h(i)$. Since $\theta(h)$ is ground, $\theta(h(i)) = \theta(s(j))$ is a constant. Since the linearization Δ^{lin} copies constants through, we obtain $\delta(h(i)) = \delta(s(j)) = \Delta^{\mathsf{lin}}(\theta(s), j) = \theta(s(j)) = \theta(h(i))$.

It is correct to conclude $\delta(h) = \theta(h) = t$.

From $\delta(B) = \Delta^{\mathsf{lin}}(\theta(B))$ and $\theta(B) \subseteq F$, it follows $\delta(B) \subseteq \Delta^{\mathsf{lin}}(F)$, hence $\delta(h) = t \in \tau(\Delta^{\mathsf{lin}}(F))$. Since t is an arbitrary tuple of $\mathbf{grd}(\tau(F))$, $\mathbf{grd}(\tau(F)) \subseteq \tau(\Delta^{\mathsf{lin}}(F))$. This concludes the proof. □

THEOREM 6.10. *Let Σ be a satisfiable set of full dependencies. For every relation I, there exists a* linCQ*-nucleus of $I^{\downarrow\uparrow\Sigma}$ that is linear and whose size is polynomially bounded in $|I|$.*

PROOF. By Corollary 5.4, we can assume the existence of a CQ-nucleus G of $I^{\downarrow\uparrow\Sigma}$, $\tau(I^{\downarrow\uparrow\Sigma}) = \mathbf{grd}(\tau(G))$ for each tableau query τ. Let Δ^{lin} be a linearization of G. For any linear tableau query τ, $\mathbf{grd}(\tau(G)) = \mathbf{grd}(\tau(\Delta^{\mathsf{lin}}(G)))$ by Lemma 6.9, hence $\mathbf{grd}(\tau(\Delta^{\mathsf{lin}}(G))) = \tau(I^{\downarrow\uparrow\Sigma})$. It follows that $\Delta^{\mathsf{lin}}(G)$ is a linCQ-nucleus of $I^{\downarrow\uparrow\Sigma}$.

Let H be a reduced tableau equivalent to $\Delta^{\mathsf{lin}}(G)$. By Corollary 3.5, H is a linCQ-nucleus of $I^{\downarrow\uparrow\Sigma}$. Every symbol in H is either a constant in $\mathbf{asd}(I) \cup \mathbf{asd}(\Sigma)$ or a variable, which may be denoted $\bullet$. Let l be the length of Σ. Let n be the arity of I. The number of constants appearing in I and Σ is at most $n|I| + l$. Hence, H can contain no more than $(n|I| + l + 1)^n$ tuples, or else H would not be reduced. This concludes the proof. □

The following corollary is now straightforward.

COROLLARY 6.11. *Let Σ a satisfiable set of full dependencies. If the complexity of the set*

$$\{(I, t) \mid I \in \mathfrak{R},\ t \text{ is a linear tuple and } I^{\downarrow\uparrow\Sigma} \stackrel{*}{\geq} \{t\}\}$$

is in **P**, *then computing a* linCQ*-nucleus of I and Σ is in polynomial time in $|I|$.*

6.3 Quantifier-free Tableau Queries

Quantifier-free tableau queries further restrict linear tableau queries. The following results correspond to those obtained for linear tableau queries.

Definition 6.12. A tableau query $\tau = (B, h)$ is said to be *quantifier-free* if every variable that occurs in B also occurs in h. The class of quantifier-free tableau queries is denoted qfCQ.

Obviously, qfCQ $\subset$ linCQ $\subset$ CQ.

LEMMA 6.13. *For every tableau F, for every quantifier-free tableau query τ, $\mathbf{grd}(\tau(F)) = \tau(\mathbf{grd}(F))$.*

PROOF. Let $\tau = (B, h)$ be a quantifier-free tableau query. It suffices to show $\mathbf{grd}(\tau(F)) \subseteq \tau(\mathbf{grd}(F))$ and $\tau(\mathbf{grd}(F)) \subseteq \mathbf{grd}(\tau(F))$.

Since $F \succeq \mathbf{grd}(F)$ is obvious, $\tau(F) \succeq \tau(\mathbf{grd}(F))$ by Lemma 3.4. Since $\tau(\mathbf{grd}(F))$ is ground, $\tau(\mathbf{grd}(F)) \subseteq \mathbf{grd}(\tau(F))$.

For the inclusion $\mathbf{grd}(\tau(F)) \subseteq \tau(\mathbf{grd}(F))$, assume an arbitrary ground tuple $t \in \mathbf{grd}(\tau(F))$. We can assume a substitution θ such that $\theta(B) \subseteq F$ and $\theta(h) = t$. If $\theta(B)$ is not ground, then since (B, h) is quantifier-free, $\theta(h)$ is not ground, a contradiction. We conclude by contradiction that $\theta(B)$ is ground, hence $\theta(B) \subseteq \mathbf{grd}(F)$. If follows $t \in \tau(\mathbf{grd}(F))$. □

THEOREM 6.14. *Let Σ be a satisfiable set of full dependencies. For every relation I, there exists a* qfCQ*-nucleus of $I^{\downarrow\uparrow\Sigma}$ that is ground and whose size is polynomially bounded in $|I|$.*

PROOF. Similar to the proof of Theorem 6.10. □

COROLLARY 6.15. *Let Σ a satisfiable set of full dependencies. If the complexity of the set*

$$\{(I, t) \mid I \in \mathfrak{R},\ t \text{ is a ground tuple and } I^{\downarrow\uparrow\Sigma} \overset{*}{\succeq} \{t\}\}$$

is in **P**, *then computing a* qfCQ*-nucleus of I and Σ is in polynomial time in $|I|$.*

7. KEY DEPENDENCIES

In this section and the following one, we show practical cases where the nucleus construction (and hence consistent query answering) is tractable. We also determine frontiers of tractability. We first look at relations I with a key constraint. For linear tableau queries, the nucleus construction takes $\mathcal{O}(m \log m)$ time, where $m = |I|$. Consistent query answering becomes intractable if we omit the linearity restriction.

The dependency ϵ_2 of Figure 1 encodes the key dependency *key*(Model, Version). It is well-known how to express a key dependency as a set of fegd's; the following definition is included for completeness.

Definition 7.1. Assume arity n. Let $K \subseteq \{1, 2, \ldots, n\}$. Let $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$ be distinct variables. Let $s = \langle x_1, \ldots, x_n \rangle$. Let t be a tuple such that for every $i \in \{1, 2, \ldots, n\}$, if $i \in K$, then $t(i) = x_i$, else $t(i) = y_i$. We write *key*(K) for the set Σ of fegd's containing $(\{s, t\}, x_j = y_j)$ for every $j \notin K$ $(1 \leq j \leq n)$. *key*(K) is called a *key dependency*, and every $i \in K$ is called a *key* attribute. Curly braces will be omitted: we write *key*$(i_1, \ldots, i_k)$ for *key*$(\{i_1, \ldots, i_k\})$.

Figure 7 illustrates the construction of a linCQ-nucleus G of $I^{\downarrow\uparrow key(1,2)}$: for every group $J \subseteq I$ of tuples that agree on all key attributes, G contains a single tuple t such that $t(i) = c$ if $s(i) = c$ for each $s \in J$; otherwise $t(i)$ is a new distinct variable not occurring elsewhere $(1 \leq i \leq n, c \in \mathbf{dom})$. It is clear that for any relation I, this construction takes only $\mathcal{O}(m \log m)$ time, where $m = |I|$, the time to sort I by key values. We show that this construction is correct.

I

1	2	3	4
a	b	d	f
a	b	e	f
a	c	d	f
a	c	d	h

G

1	2	3	4
a	b	x	f
a	c	d	y

Fig. 7. G is a linCQ-nucleus of $I^{\downarrow\uparrow key(1,2)}$.

LEMMA 7.2. *Assume arity n and let $K \subseteq \{1, 2, \ldots, n\}$. Let t be a linear tuple. Deciding whether $I^{\downarrow\uparrow key(K)} \stackrel{*}{\succeq} \{t\}$ is in polynomial time in the cardinality of the input relation I.*

PROOF. Let $\mathcal{P}$ be the partition of I defined by the equivalence relation $\equiv$ defined as: $s \equiv s'$ if and only if for each $i \in K$, $s(i) = s'(i)$ $(s, s' \in I)$. For each partition class $P \in \mathcal{P}$, for each $i \in \{1, 2, \ldots, n\}$, define $P^i := \{s(i) \mid s \in P\}$, the set of constants occurring in the ith column of P. For every tuple $r \in P^1 \times P^2 \times \cdots \times P^n$, let $\Delta^r : P \times \{1, 2, \ldots, n\} \to \mathbf{dom} \cup \mathbf{var}$ denote the transformation such that for each $s \in P$, for each $i \in \{1, 2, \ldots, n\}$, $\Delta^r(s, i) = s(i)$ if $s(i) = r(i)$; otherwise $\Delta^r(s, i)$ is a new distinct variable not occurring elsewhere. It is easy to see that $\Delta^r(P)$ is a fix of P and $key(K)$, and that a chase of $\Delta^r(P)$ by $key(K)$ ends with $\{r\}$, which is ground and hence an uprepair. Hence, for every $r \in P^1 \times \cdots \times P^n$, $\{r\}$ is an uprepair generated by P and $key(K)$. It can be easily verified that fixes containing variables in some of the key columns result in uprepairs properly containing $\{r\}$ for some $r \in P^1 \times \cdots \times P^n$. Then, $P^{\downarrow\uparrow key(K)} \succeq \{t\}$ if and only if for every $i \in \{1, 2, \ldots, n\}$, if $t(i)$ is a constant, then for all $s \in P$, $s(i) = t(i)$. Since tuples belonging to different partition classes of $\mathcal{P}$ disagree on some key attribute, $I^{\downarrow\uparrow key(K)} \succeq \{t\}$ if and only if for some $P \in \mathcal{P}$, $P^{\downarrow\uparrow key(K)} \succeq \{t\}$. It is easy to see that the time complexity for deciding $I^{\downarrow\uparrow key(K)} \stackrel{*}{\succeq} \{t\}$ is $\mathcal{O}(m \log m)$ where $m = |I|$. □

THEOREM 7.3. *Assume arity n and let $K \subseteq \{1, 2, \ldots, n\}$. Constructing a* linCQ*-nucleus of $I^{\downarrow\uparrow key(K)}$ is in polynomial time in the cardinality of the input relation I.*

PROOF. From Lemma 7.2 and Corollary 6.11. □

This result does not extend to not-necessarily-linear tableau queries, since consistent query answering is **NP**-complete if we omit the linearity requirement.

THEOREM 7.4. *For the Boolean tableau query*

τ

1	2	3
x	y	a
z	y	b
1		

,

CQA($key(1), \tau$) *is* **NP***-complete.*

PROOF. Membership of **NP** follows from Theorem 5.10. The proof, inspired by Theorem 3.3 in Chomicki and Marcinkowski [2005a], is a reduction from MONOTONE 3SAT. Let $\phi = \bigwedge_{i=1}^{m} C_i$ where each C_i is either a disjunction of three negated Boolean variables or a disjunction of three nonnegated Boolean

variables. Construct a relation I such that for each $i \in \{1, \ldots, n\}$, for each p that occurs in C_i, if C_i contains $\neg p$, then I contains $\langle i, p, b\rangle$, else I contains $\langle i, p, a\rangle$, where $a, b \in \mathbf{dom}$. That is, $C_i = p \vee q \vee r$ gives rise to the tuples in I shown below. For $C_i = \neg p \vee \neg q \vee \neg r$, simply replace a by b. It can be verified that all fixes of I and $key(1)$ must contain F_1, F_2, or F_3 shown below, up to a permutation of p, q, r; the chase results in G_1, G_2, and G_3, respectively. Note incidentally that the shown parts of F_1, F_2, and F_3 are not comparable by $\sqsupseteq$.

I

1	2	3
i	p	a
i	q	a
i	r	a

F_1

1	2	3
i	p	a
i	•	a
i	•	a

F_2

1	2	3
i	p	a
i	•	a
•	r	a

F_3

1	2	3
i	p	a
•	q	a
•	r	a

G_1

1	2	3
i	p	a

G_2

1	2	3
i	p	a
•	r	a

$G_3 = F_3$

Let B and h denote the body and the head of τ, respectively. We show that ϕ has a satisfying truth assignment if and only if $\tau(I^{\downarrow\uparrow key(1)}) = \{\}$. For the *if part*, assume $\tau(I^{\downarrow\uparrow key(1)}) = \{\}$. Since τ is Boolean, by Lemma 2.6 and Theorem 3.22, for some fix F of I and Σ, $G \not\geq B$ where G is the final tableau in a chase of F by $key(1)$. For every $i \in \{1, 2, \ldots, n\}$, G contains exactly one tuple of the form $\langle i, \alpha, \beta\rangle$, where α is a Boolean variable and $\beta \in \{a, b\}$. Let θ be a truth assignment such that $\theta(p) = \mathbf{true}$ if $\langle i, p, a\rangle \in G$, and $\theta(p) = \mathbf{false}$ if $\langle i, p, b\rangle \in G$. The same variable cannot be assigned both **true** en **false**, or else $G \geq B$, a contradiction. It can be easily verified that θ is a satisfying truth assignment of ϕ. The *only-if part* is now straightforward. □

Obviously, when tableau queries can appear disguised as ftgd's, the restriction to quantifier-free tableau queries does not lower the complexity of consistent query answering.

COROLLARY 7.5. *For the ftgd τ_1 and the Boolean* qfCQ *query τ_2,*

τ_1

1	2	3
x	y	a
z	y	b
1	1	1

τ_2

1	2	3
1	1	1
1		

,

CQA($key(1) \cup \{\tau_1\}, \tau_2$) *is* **NP***-complete.*

8. CONTRADICTION-GENERATING DEPENDENCIES

We study update-based repairing when all constraints are fegd's of the form $(B, 0 = 1)$. The nucleus construction (and hence consistent query answering) is tractable for quantifier-free tableau queries, but consistent query answering becomes intractable for linear tableau queries.

8.1 Comprehensive Example

Definition 8.1. A *contradiction-generating dependency* (cgd) is an fegd of the form $(B, 0 = 1)$, where 0 and 1 are distinct constants.

An example is the cgd ϵ_3 of our running example, which expresses that car models available in luxury versions do not exist in bottom priced variants. A fortiori, luxury models are never bottom priced (ϵ_7). Each cgd corresponds logically to a Horn clause without a positive literal: the cgd ϵ_3 expresses $\forall x \forall y \forall z$ $(\neg R(x, \text{luxury}, z) \vee \neg R(x, y, \text{bottom}))$.

ϵ_3

Model	Version	PriceRange
x	luxury	z
x	y	bottom
$0 = 1$		

ϵ_7

Model	Version	PriceRange
x	luxury	bottom
$0 = 1$		

Since ϵ_3 logically implies ϵ_7, $\{\epsilon_3, \epsilon_7\}$ and $\{\epsilon_3\}$ are equivalent sets of constraints. An entry in (the body of) a cgd is *irrelevant* if it contains a variable that occurs only once in the cgd; otherwise it is *relevant*. In ϵ_3, the irrelevant entries are the entries containing y and z; in ϵ_7, the entry containing x is irrelevant.

We illustrate how the following relation CARS $= \{s, t\}$ can be made subconsistent relative to $\{\epsilon_3, \epsilon_7\}$. Obviously, every tableau that is subconsistent with respect to to a set of cgd's is also consistent.

CARS

Model	Version	PriceRange	
4×4	luxury	medium	(s)
4×4	luxury	bottom	(t)

The substitution $\theta_1 = \{x/4 \times 4, y/\text{luxury}, z/\text{medium}\}$ is a one-one homomorphism from the body of ϵ_3 to CARS, hence CARS is inconsistent. In order to gain consistency, we need to put a variable at any entry of CARS that is related via θ_1 to a relevant entry of ϵ_3: at the first coordinate of s or t (both related to x), the second coordinate of s, or the third coordinate of t. This will be encoded as the *fixset* $D_1 = \{(s, 1), (t, 1), (s, 2), (t, 3)\}$ of CARS-entries. Note incidentally that the third coordinate of s (medium) is related via θ_1 to an irrelevant entry of ϵ_3 (the entry containing z) and that substituting a variable for "medium" is useless for gaining consistency. Likewise, the substitution $\theta_2 = \{x/4 \times 4\}$ is a one-one homomorphism from the body of ϵ_7 to CARS, resulting in the fixset $D_2 = \{(t, 2), (t, 3)\}$. Note here that the entry containing x is not relevant in ϵ_7. For the current set of cgd's, there are no further one-one homomorphisms.

Next, if E is a hitting set of $\{D_1, D_2\}$ (E intersects both D_1 and D_2), then CARS is rendered consistent by putting variables at all entries of E. There are four minimal (with respect to $\subseteq$) hitting sets: $E_1 = \{(s, 1), (t, 2)\}$, $E_2 = \{(s, 2), (t, 2)\}$, $E_3 = \{(t, 1), (t, 2)\}$, $E_4 = \{(t, 3)\}$. For example, we show the

consistent tableau $\Delta^{E_1}(\mathsf{CARS})$ obtained from CARS by putting variables at the entries of E_1 (see Definition 5.6):

$\Delta^{E_1}(\mathsf{CARS})$	Model	Version	PriceRange
	•	luxury	medium
	4×4	•	bottom

8.2 $\sqsupseteq$-closed Sets of Contradiction-Generating Dependencies

The starting point of the introductory example is a set $\Sigma = \{\epsilon_3, \epsilon_7\}$ of cgd's such that for any tableau F, if F is inconsistent, then the body of some cgd is one-one homomorphic to F. The limitation to homomorphisms that are one-one is not a limitation (Theorem 8.3) and eases the technical treatment to follow.

Definition 8.2. A set Σ of cgd's is $\sqsupseteq$-*closed* if and only if for every tableau F, if $F \succeq B$ for some $(B, 0 = 1)$ in Σ, then for some $(B', 0 = 1)$ in Σ, $F \sqsupseteq B'$.

To see that $\{\epsilon_3\}$ is not $\sqsupseteq$-closed, it suffices to note that the body of ϵ_3 is not one-one homomorphic to the (inconsistent) body of ϵ_7. On the other hand, $\{\epsilon_3, \epsilon_7\}$ is $\sqsupseteq$-closed.

Theorem 8.3 shows that every set of cgd's is equivalent to a $\sqsupseteq$-closed set of cgd's. Moreover, the proof provides an effective way to construct a $\sqsupseteq$-closed equivalent set of cgd's. It is not difficult to develop more efficient constructions using most general unifiers [Abiteboul et al. 1995].

THEOREM 8.3. *Every set Σ of cgd's is logically equivalent to a $\sqsupseteq$-closed set of cgd's.*

PROOF. Assume arity n. For every cgd $\epsilon = (B, 0 = 1)$ of Σ, construct Σ_ϵ as the set of cgd's containing $(B', 0 = 1)$ for every tableau B' satisfying (i) $\mathbf{asd}(B') \subseteq \mathbf{asd}(B)$, (ii) $|B'| \leq |B|$, and (iii) $\mu(B) = B'$ for some substitution μ. Clearly, Σ_ϵ is finite because of (i) and (ii). Note incidentally $\epsilon \in \Sigma_\epsilon$. Let $\Sigma' = \bigcup_{\epsilon \in \Sigma} \Sigma_\epsilon$. It is easy to verify that Σ and Σ' are equivalent.

To show that Σ' is $\sqsupseteq$-closed, assume a tableau F such that $F \succeq B'$ for some cgd $(B', 0 = 1)$ in Σ'. The construction ensures the existence of a cgd $(B, 0 = 1)$ in Σ such that $B' \succeq B$. By transitivity, $F \succeq B$. We can assume a substitution θ for the variables in B such that $\theta(B) \subseteq F$. Let C be the set of constants that occur in $\theta(B)$ but not in B. For each $c \in C$, let x_c denote a new distinct variable not occurring elsewhere. Let μ be the substitution satisfying for each variable x occurring in B, if $\theta(x) = c$ for some $c \in C$, then $\mu(x) = x_c$, else $\mu(x) = \theta(x)$.

Let $G = \mu(B)$. Obviously, $\theta(B) \sqsupseteq G$. We have $|G| \leq |B|$, G contains no constants not in B, and G contains not more variables than B. Then, our construction ensures that $G \simeq H$ for some cgd $(H, 0 = 1)$ in Σ_ϵ. From $\theta(B) \subseteq F$, it follows $F \sqsupseteq \theta(B)$. By transitivity, $F \sqsupseteq G$. Since $G \simeq H$, $F \sqsupseteq H$. This concludes the proof. □

8.3 Hitting Set

Hitting sets are well-known from complexity theory. Significantly, when we refer to minimal hitting sets in the technical treatment to follow, minimality refers to set inclusion, not cardinality.

Definition 8.4. Let U be a finite set and $\mathbf{S} \subseteq 2^U$. A set $E \subseteq U$ is a *hitting set* of $\mathbf{S}$ if for every $S \in \mathbf{S}$, $E \cap S \neq \{\}$. A hitting set E of $\mathbf{S}$ is said to be *minimal* if no proper subset of E is also a hitting set of $\mathbf{S}$.

LEMMA 8.5. *Let U be a finite set. Let $\mathbf{S} \subseteq 2^U$. Let E be a minimal (with respect to $\subseteq$) hitting set of $\mathbf{S}$. For every $b \in E$, there exists $S \in \mathbf{S}$ such that $E \cap S = \{b\}$.*

PROOF. Assume $b \in E$. Since E is a hitting set of $\mathbf{S}$, for every $S \in \mathbf{S}$, $E \cap S \neq \{\}$. Assume that for every $S \in \mathbf{S}$, $E \cap S \neq \{b\}$. Then, $E \setminus \{b\}$ is a hitting set of $\mathbf{S}$, hence E is not minimal, a contradiction. We conclude by contradiction that for some $S \in \mathbf{S}$, $E \cap S = \{b\}$. □

We provide a necessary and sufficient condition for the existence of a minimal hitting set that contains a given element b.

THEOREM 8.6. *Let U be a finite set. Let $\mathbf{S} \subseteq 2^U$ and $b \in U$. Then, $\mathbf{S}$ has a minimal (with respect to $\subseteq$) hitting set containing b if and only if there exists a set $S_b \in \mathbf{S}$ such that $b \in S_b$ and for every $S \in \mathbf{S}$, if $b \notin S$, then $S \not\subseteq S_b$.*

PROOF. *If part.* Assume $S_b \in \mathbf{S}$ such that $b \in S_b$ and for every $S \in \mathbf{S}$, $b \notin S$ implies $S \not\subseteq S_b$, i.e. $S \setminus S_b \neq \{\}$. Let E be a set such that $b \in E$ and for every $S \in \mathbf{S}$ such that $b \notin S$, E contains an element of $S \setminus S_b$. Clearly, E is a hitting set of $\mathbf{S}$, and $E \setminus \{b\}$ is not a hitting set of $\mathbf{S}$. It follows that E contains a minimal (with respect to $\subseteq$) hitting set E' of $\mathbf{S}$ such that E' contains b.

Only-if part. Assume E is a minimal (with respect to $\subseteq$) hitting set of $\mathbf{S}$ containing b. By Lemma 8.5, we can assume the existence of an element $S_b \in \mathbf{S}$ such that $S_b \cap E = \{b\}$. Assume $S \in \mathbf{S}$ such that $b \notin S$. Since E is a hitting set of $\mathbf{S}$, we can assume $c \in E$ such that $c \in S$. Assume $c \in S_b$. From $c \in S$ and $b \notin S$, it follows $c \neq b$. Then $S_b \cap E$ contains two distinct elements b and c, a contradiction. We conclude by contradiction $c \notin S_b$. Since $c \in S$, $S \not\subseteq S_b$. □

Consequently, it can be decided in polynomial time in $|\mathbf{S}|$ whether some minimal hitting set of $\mathbf{S}$ contains a given element b.

8.4 Fixes and Hitting Sets

An entry in a tableau F is not relevant if it contains a variable not occurring elsewhere. Relevant entries are used for determining fixsets, as was shown in the introductory example.

Definition 8.7. Let F be a tableau of arity n. An entry (t, i) of F is said to be *relevant* in F if $t(i) \in \mathbf{dom}$ or if for some entry (s, j) in F, $(t, i) \neq (s, j)$ and $t(i) = s(j)$. We write *relev*(F) for the set of relevant entries of F.

Let I be a relation and Σ a set of cgd's. If for some $(B, 0 = 1)$ of Σ, the substitution θ is a one-one homomorphism from B to I, then the set $\{(\theta(t), i) \mid (t, i) \in \mathit{relev}(B)\}$ of I-entries is called a *fixset* of I and Σ. We write *fixsets*(I, Σ) for the set of all fixsets of I and Σ.

The set *fixsets*(I, Σ) can be constructed in polynomial time in $|I|$: every cgd $(B, 0 = 1)$ results in no more than $|I|^m$ fixsets, where $m = |B|$. We show that hitting sets correspond to consistent tableaux.

LEMMA 8.8. *Assume arity n. Let I be a relation and Σ a $\sqsupseteq$-closed set of cgd's. Let E be a set of I-entries (i.e. $E \subseteq I \times \{1, 2, \ldots, n\}$). Then, E is a hitting set of $\mathit{fixsets}(I, \Sigma)$ if and only if $\Delta^E(I)$ is consistent.*

PROOF. *If part.* Assume E is not a hitting set of $\mathit{fixsets}(I, \Sigma)$. We can assume a cgd $(B, 0 = 1)$ in Σ and a substitution θ such that $\theta(B) \subseteq I$, θ identifies no two distinct tuples of B, and E contains no element of the fixset $\{(\theta(t), i) \mid (t, i) \in \mathit{relev}(B)\}$. Let ω be the substitution such that for each symbol p that occurs in B (let (t, i) be a B-entry such that $t(i) = p$), $\omega(p) = \Delta^E(\theta(t), i)$.

We show that ω is the identity on constants: if $t(i)$ is a constant (say a) for some entry (t, i) in B, then, since $(t, i) \in \mathit{relev}(B)$, $(\theta(t), i) \notin E$, hence $\omega(a) = \Delta^E(\theta(t), i) = a$. To show that ω is well-defined on variables, we prove that if $s(j) = t(j)$ is the same variable (say x) for distinct B-entries (s, j) and (t, i), then $\Delta^E(\theta(t), i) = \Delta^E(\theta(s), j)$. Since $(t, i) \in \mathit{relev}(B)$ and $(s, j) \in \mathit{relev}(B)$, it follows $(\theta(t), i) \notin E$ and $(\theta(s), j) \notin E$. Then, $\Delta^E(\theta(t), i) = \theta(t(i)) = \theta(x)$ and $\Delta^E(\theta(s), j) = \theta(s(j)) = \theta(x)$, hence $\Delta^E(\theta(t), i) = \Delta^E(\theta(s), j)$.

We have $\omega(B) = \Delta^E(\theta(B)) \subseteq \Delta^E(I)$, hence $\Delta^E(I)$ is inconsistent.

Only-if part. Assume $\Delta^E(I)$ inconsistent. Since Σ is $\sqsupseteq$-closed, there exists a cgd $(B, 0 = 1)$ in Σ and a substitution θ such that $\theta(B) \subseteq \Delta^E(I)$ and θ identifies no two distinct tuples of B. Let μ be the substitution such that for every $t \in \Delta^E(I)$, $\Delta^E(\mu(t)) = t$. Let $\omega = \mu \circ \theta$. Then, $\omega(B) \subseteq I$ and ω identifies no two distinct tuples of B. Let $K = \{(\omega(s), j) \mid (s, j) \in \mathit{relev}(B)\}$. Clearly, $K \in \mathit{fixsets}(I, \Sigma)$. To show that E is not a hitting set of $\mathit{fixsets}(I, \Sigma)$, it suffices to show that E contains no element of K. Assume, on the contrary, that E contains $(\omega(s), j) \in K$. Two cases can occur:

Case $s(j)$ is a constant. From $\Delta^E(\omega(s), j) = \Delta^E(\mu(\theta(s)), j) = \theta(s(j))$, it follows $\Delta^E(\omega(s), j)$ is a constant. From $(\omega(s), j) \in E$, it follows $\Delta^E(\omega(s), j)$ is a variable. Then $\mathbf{dom} \cap \mathbf{var} \neq \{\}$, a contradiction.

Case $s(j)$ is a variable. Since $(s, j) \in \mathit{relev}(B)$, there exists a B-entry (t, i) distinct from (s, j) such that $t(i) = s(j)$. From $(\omega(s), j) \in E$, it follows $\Delta^E(\omega(s), j)$ is a new distinct variable not occurring elsewhere. On the other hand, $\Delta^E(\omega(s), j) = \Delta^E(\mu(\theta(s)), j) = \theta(s(j))$ and $\Delta^E(\omega(t), i) = \Delta^E(\mu(\theta(t)), i) = \theta(t(i))$. Since $t(i) = s(j)$, $\Delta^E(\omega(s), j) = \Delta^E(\omega(t), i)$, hence $\Delta^E(\omega(s), j)$ occurs twice in $\Delta^E(I)$, a contradiction.

We conclude by contradiction that E is not a hitting set of $\mathit{fixsets}(I, \Sigma)$. □

We show that fixes correspond to minimal (with respect to $\subseteq$) hitting sets.

LEMMA 8.9. *Assume arity n. Let I be a relation and Σ a $\sqsupseteq$-closed set of cgd's. Let E be a set of I-entries. If $\Delta^E(I)$ is a fix of I and Σ, then E is a minimal hitting set of $\mathit{fixsets}(I, \Sigma)$.*

PROOF. Assume E is not a minimal hitting set of $\mathit{fixsets}(I, \Sigma)$. If E is not a hitting set, then $\Delta^E(I)$ is inconsistent by Lemma 8.8. If E is a hitting set, then we can assume $E' \subsetneq E$ such that E' is a minimal hitting set of $\mathit{fixsets}(I, \Sigma)$. Since $\Delta^{E'}(I)$ is consistent by Lemma 8.8 and since $I \supseteq \Delta^{E'}(I) \supset \Delta^E(I)$ is obvious, $\Delta^E(I)$ is not a fix of I and Σ. □

Remarkably, the inverse is not true. For a counterexample, consider the relation $I = \{s, t\}$ and the cgd's $\epsilon_8, \epsilon_9, \epsilon_{10}$ in Figure 8 ($a, b, c, d \in \mathbf{dom}$). The cgd ϵ_8 results

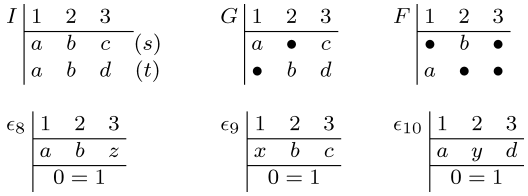


Fig. 8.

in the fixsets $D_1 = \{(s, 1), (s, 2)\}$ and $D_2 = \{(t, 1), (t, 2)\}$. The cgd ϵ_9 results in the fixset $D_3 = \{(s, 2), (s, 3)\}$, and ϵ_{10} in $D_4 = \{(t, 1), (t, 3)\}$. A minimal (with respect to $\subseteq$) hitting set of $\{D_1, D_2, D_3, D_4\}$ is $E = \{(s, 1), (s, 3), (t, 2), (t, 3)\}$. We have $F = \Delta^E(I)$ but F is not a fix, since $I \sqsupseteq G \sqsupset F$ and G is also consistent. Note incidentally that $G = \Delta^{E'}(I)$ with $E' = \{(s, 2), (t, 1)\}$, which is also a minimal hitting set of $\{D_1, D_2, D_3, D_4\}$.

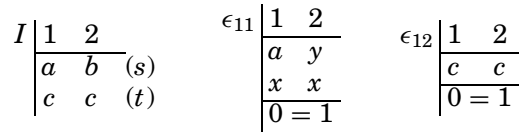
8.5 qfCQ-nucleus for Contradiction-Generating Dependencies

Finally, we show that for each relation I and satisfiable set Σ of cgd's, a qfCQ-nucleus of all uprepairs can be constructed in polynomial time. This is the strongest result possible as any quantification results in intractability.

LEMMA 8.10. *Assume arity n. Let I be a relation and Σ a coherent, $\sqsupseteq$-closed set of cgd's. Let $t \in I$. Then, $I^{\downarrow\uparrow\Sigma} \overset{*}{\succeq} \{t\}$ if and only if for every $i \in \{1, 2, \ldots, n\}$, there is no minimal hitting set of fixsets(I, Σ) containing (t, i).*

PROOF. *If part.* Assume $I^{\downarrow\uparrow\Sigma} \overset{*}{\not\succeq} \{t\}$. From Lemma 2.6 and Theorem 3.22, there is a fix F of I and Σ such that $t \notin F$. By Lemma 5.7, for some set E of I-entries, $\Delta^E(I) \simeq F$. By Lemma 8.9, E is a minimal hitting set of *fixsets*(I, Σ). Since $t \notin \Delta^E(I)$, E contains (t, i) for some $i \in \{1, 2, \ldots, n\}$. *Only-if part.* Assume the existence of a minimal hitting set E containing at least one entry of $\{(t, i) \mid 1 \leq i \leq n\}$. Since $I \sqsupseteq \Delta^E(I)$ is obvious and $\Delta^E(I)$ consistent by Lemma 8.8, there exists some fix F such that $I \sqsupseteq F \sqsupseteq \Delta^E(I)$. Let $E' = E \setminus \{(t, i) \mid 1 \leq i \leq n\}$. Since $E' \subsetneq E$, E' is not a hitting set of *fixsets*(I, Σ), hence $\Delta^{E'}(I)$ inconsistent by Lemma 8.8. Intuitively, $\Delta^{E'}(I)$ can be obtained from $\Delta^E(I)$ by "unfixing" t. The fix F cannot contain t, or else $F \sqsupseteq \Delta^{E'}(I)$, hence F inconsistent, a contradiction. We conclude that for some fix F of I and Σ, $t \notin F$. Consequently, $I^{\downarrow\uparrow\Sigma} \overset{*}{\not\succeq} \{t\}$. □

For example, for the coherent and $\sqsupseteq$-closed set $\Sigma = \{\epsilon_{11}, \epsilon_{12}\}$ and the relation $I = \{s, t\}$ shown next, we get *fixsets*$(I, \Sigma) = \{\{(s, 1), (t, 1), (t, 2)\}, \{(t, 1), (t, 2)\}\}$. The minimal hitting sets of *fixsets*(I, Σ) are $\{(t, 1)\}$ and $\{(t, 2)\}$. Since no minimal hitting set contains $(s, 1)$ or $(s, 2)$, it follows that each fix contains s.



COROLLARY 8.11. *Let Σ be a satisfiable set of cgd's. Let t a ground tuple. Deciding whether $I^{\downarrow\uparrow\Sigma} \overset{*}{\succeq} \{t\}$ is in polynomial time in the cardinality of the input relation I.*

PROOF. Clearly, if $t \notin I$ or Σ not coherent, then $I^{\downarrow\uparrow\Sigma} \not\stackrel{*}{\sqsupseteq} \{t\}$. Next assume $t \in I$ and Σ coherent. We first construct a $\sqsupseteq$-closed set Σ' equivalent to Σ, and then compute $\mathit{fixsets}(I, \Sigma')$, which takes only polynomial time in $|I|$. The result then follows from Lemma 8.10 and Theorem 8.6. □

THEOREM 8.12. *Let Σ be a satisfiable set of cgd's. Constructing a* qfCQ*-nucleus of $I^{\downarrow\uparrow\Sigma}$ is in polynomial time in the cardinality of the input relation I.*

PROOF. From Corollary 8.11 and Corollary 6.15. □

Remarkably, a shift from quantifier-free to linear tableau queries results in intractability. That is, although a linCQ-nucleus of polynomial size exists (Theorem 6.10), its construction takes exponential time, unless **P** = **NP**.

THEOREM 8.13. *Let $\Sigma = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_7\}$ and τ as follows. Then,* CQA(Σ, τ) *is* **NP***-complete.*

ϵ_1

1	2	3	4	5
c	a	x	v	w
x	y	u'	v'	w'
$0 = 1$				

ϵ_2

1	2	3	4	5
c	a	u	x	w
x	y	u'	v'	w'
$0 = 1$				

ϵ_3

1	2	3	4	5
c	a	u	v	x
x	y	u'	v'	w'
$0 = 1$				

ϵ_4

1	2	3	4	5
c	b	y	v	w
x	y	u'	v'	w'
$0 = 1$				

ϵ_5

1	2	3	4	5
c	b	u	y	w
x	y	u'	v'	w'
$0 = 1$				

ϵ_6

1	2	3	4	5
c	b	u	v	y
x	y	u'	v'	w'
$0 = 1$				

ϵ_7

1	2	3	4	5
x	x	u'	v'	w'
$0 = 1$				

τ

1	2	3	4	5
c	z	u	v	w
1				

PROOF. Membership of **NP** follows from Theorem 5.10. Reduction from MONOTONE 3SAT. Let a, b, c be distinct constants. Let $\phi = \bigwedge_{i=1}^{m} C_i$, where each C_i is a disjunction of three distinct Boolean variables. Let V be the set of Boolean variables occurring in ϕ. The formula ϕ is encoded in a relation I as follows. For every clause $C_i = p \vee q \vee r$, I contains a tuple $\langle c, a, p, q, r\rangle$. For a clause $C_i = \neg p \vee \neg q \vee \neg r$, replace a by b. Furthermore, for each Boolean variable $p \in V$, I contains a tuple $t_p := \langle p, p, p, p, p\rangle$. For example, $C_i = p \vee q \vee r$ gives rise to the tuples in I shown below.

I

1	2	3	4	5	
		$\vdots$			
c	a	p	q	r	
		$\vdots$			
p	p	p	p	p	(t_p)
q	q	q	q	q	(t_q)
r	r	r	r	r	(t_r)
		$\vdots$			

F_θ

1	2	3	4	5	
		$\vdots$			
•	a	p	q	r	
		$\vdots$			
p	•	p	p	p	(t'_p)
•	q	q	q	q	(t'_q)
r	•	r	r	r	(t'_r)
		$\vdots$			

The only tuples in Σ that can be mapped to t_p are of the form $\langle \cdot, \cdot, u', v', w' \rangle$. Since u', v', w' never occur more than once in the body of some cgd, every fix of I and Σ must contain a tuple of the form $\langle \cdot, \cdot, p, p, p \rangle$, which will be denoted t'_p. Furthermore, either $t'_p(1)$ or $t'_p(2)$ must be a variable because of ϵ_7.

Since Σ contains only cgd's and τ is a Boolean tableau query, it follows from Theorem 3.22 that $\tau(I^{\downarrow\uparrow\Sigma}) = \{\}$ if and only if $\tau(F) = \{\}$ for some fix F of I and Σ. Then, $\tau(I^{\downarrow\uparrow\Sigma}) = \{\}$ if and only if some fix contains no occurrence of the constant c in its first column. So it suffices to show that ϕ has a satisfying truth assignment if and only if some fix does not contain c in its first column.

Assume that θ is a satisfying truth assignment of ϕ. For every $p \in V$, let $t'_p = \langle p, \bullet, p, p, p \rangle$ if $\theta(p) = \textbf{true}$, and let $t'_p = \langle \bullet, p, p, p, p \rangle$ if $\theta(p) = \textbf{false}$. Furthermore, each occurrence of c is replaced by a new distinct variable not occurring elsewhere. Call the resulting tableau F_θ (see the example above where $\theta(p) = \theta(r) = \textbf{true}$ and $\theta(q) = \textbf{false}$). It can be easily verified that $I \sqsupseteq F_\theta$ and F_θ consistent. We now show that F_θ is maximal (with respect to $\sqsupseteq$). For every tuple of the form $\langle \bullet, a, p, q, r \rangle$ (encoding $p \vee q \vee r$), we have $t'_p(1) = p$ or $t'_q(1) = q$ or $t'_r(1) = r$; otherwise the encoded truth assignment θ does not satisfy $p \vee q \vee r$. Then, we cannot substitute c for $\bullet$ in $\langle \bullet, a, p, q, r \rangle$, or at least one of ϵ_1, ϵ_2, or ϵ_3 would be falsified. Likewise, for every tuple of the form $\langle \bullet, b, p, q, r \rangle$ (encoding $\neg p \vee \neg q \vee \neg r$), we have $t'_p(2) = p$ or $t'_q(2) = q$ or $t'_r(2) = r$. Again, we cannot substitute c for $\bullet$ in $\langle \bullet, b, p, q, r \rangle$, or one of ϵ_4, ϵ_5, or ϵ_6 would be falsified. It follows that F_θ is a fix of I and Σ containing no occurrence of c.

Conversely, assume there exists a fix F containing no occurrence of the constant c in its first column. Then, for each tuple $\langle c, a, p, q, r \rangle \in I$, there is a corresponding tuple $\langle x, a, p, q, r \rangle$ in F ($x \in \textbf{var}$). Since F is a fix and $I \sqsupseteq id_{x=c}(F) \sqsupset F$, the tableau $id_{x=c}(F)$ is not subconsistent; substituting c for x results in a violation of some cgd, which must be one of ϵ_1, ϵ_2, or ϵ_3. Then, F must contain one of $\langle p, \bullet, p, p, p \rangle$, $\langle q, \bullet, q, q, q \rangle$, or $\langle r, \bullet, r, r, r \rangle$. By the same reasoning, for each tuple of the form $\langle c, b, p, q, r \rangle \in I$ (encoding $\neg p \vee \neg q \vee \neg r$), F contains at least one of $\langle \bullet, p, p, p, p \rangle$, $\langle \bullet, q, q, q, q \rangle$, or $\langle \bullet, r, r, r, r \rangle$. Then, a satisfying truth assignment of ϕ is the truth assignment θ satisfying for every $p \in V$, $\theta(p) = \textbf{true}$ if F contains $\langle p, \bullet, p, p, p \rangle$, and $\theta(p) = \textbf{false}$ if F contains $\langle \bullet, p, p, p, p \rangle$. □

9. INEQUALITY AND UNION

We show that nuclei no longer exist if we extend tableau queries with inequalities or unions.

9.1 Inequalities

Assume arity 1. Let $I = \{\langle a \rangle\}$ and Σ the singleton containing $(\{\langle a \rangle\}, 0 = 1)$. The unique (up to $\sim$) fix is $\{\langle x \rangle\}$, and the set of uprepairs is given by $I^{\downarrow\uparrow\Sigma} = \{\{\langle p \rangle\} \mid p \in \textbf{dom}, p \neq a\}$: every uprepair changes a in another constant.

For every constant $c \in \textbf{dom}$, let $Q^{\neq c}$ be the query such that for every relation J, $Q^{\neq c}(J) = \{\langle 1 \rangle\}$ if J contains some tuple $\langle b \rangle$ with $b \neq c$; otherwise $Q^{\neq c}(J) = \{\}$. That is, $Q^{\neq c}$ corresponds to the first-order logic formula

$\exists x(R(x) \wedge x \neq c)$. Clearly,

$$Q^{\neq c}(I^{\downarrow\uparrow\Sigma}) = \begin{cases} \{\langle 1\rangle\} & \text{if } c = a \\ \{\} & \text{if } c \neq a \end{cases}.$$

Let Q be the class of queries defined as $\mathsf{Q} = \{Q^{\neq c} \mid c \in \mathbf{dom}\}$. There is no tableau F such that for every $Q^{\neq c} \in \mathsf{Q}$, $Q^{\neq c}(F) = Q^{\neq c}(I^{\downarrow\uparrow\Sigma})$, because whenever F is a tableau containing some constant, then $Q^{\neq c}(F) = \{\langle 1\rangle\}$ for some constant $c \neq a$, while a tableau F without constants cannot distinguish between a and constants $c \neq a$.

9.2 Union

Assume arity 2. Let $I = \{\langle a, a\rangle\}$ and Σ the singleton containing $(\{\langle z, z\rangle\}, 0 = 1)$. The two unique (up to $\sim$) fixes are $\{\langle a, y\rangle\}$ and $\{\langle x, a\rangle\}$. Let $\tau_1 = (\{\langle a, w\rangle\}, \langle 1\rangle)$ and $\tau_2 = (\{\langle u, a\rangle\}, \langle 1\rangle)$, two Boolean tableau queries. Let $\tau_{1\vee 2}$ be the query such that for each relation J, $\tau_{1\vee 2}(J) = \tau_1(J) \cup \tau_2(J)$. Then, $\tau_{1\vee 2}(I^{\downarrow\uparrow\Sigma}) = \{\langle 1\rangle\}$ and $\tau_1(I^{\downarrow\uparrow\Sigma}) = \tau_2(I^{\downarrow\uparrow\Sigma}) = \{\}$. Obviously, there is no single tableau F such that $\tau_{1\vee 2}(F) = \{\langle 1\rangle\}$ and $\tau_1(F) = \tau_2(F) = \{\}$.

10. COMPARISON WITH RELATED WORK

We first summarize our main results, and then we compare our results with existing work.

10.1 Summary of Main Results

This article extends and improves an earlier conference publication [Wijsen 2002]. The major contributions are: first, the new notion of update-based repairing, and second, the construction of single databases, called nuclei, that can replace all (possibly infinitely many) repairs for the purpose of consistent query answering. The major technical contributions are listed below. Recall that CQ denotes the class of conjunctive queries, linCQ is the class of conjunctive queries in which no quantified variable occurs more than once, and qfCQ is the class of quantifier-free conjunctive queries. Obviously, qfCQ $\subset$ linCQ $\subset$ CQ. All complexity results refer to data complexity.

—Fix checking is in **P** for full dependencies. Consistent query answering is in **NP** for full dependencies and conjunctive queries.

—For each relation I and set Σ of full dependencies, there exists a computable CQ-nucleus of all uprepairs (generated by I and Σ). There exists a set of full dependencies such that for every $n \geq 7$, there exists a relation with $n - 1$ tuples allowing no CQ-nucleus of size less than 2^n.

—For each set of full dependencies, the size of a linCQ-nucleus is polynomially bounded in the size of the input relation.

—For each key dependency $key(K)$, a linCQ-nucleus of all uprepairs can be computed in polynomial time in the size of the input relation, but consistent query answering becomes **NP**-complete for CQ queries.

—For each set of contradiction-generating dependencies, a qfCQ-nucleus of all uprepairs can be computed in polynomial time in the size of the input relation, but consistent query answering becomes **NP**-complete for linCQ queries.

	qfCQ	linCQ	CQ
key dependency	**P**	**P**	**NP**-complete
cgd's	**P**	**NP**-complete	**NP**-complete
full dependencies	**NP**-complete	**NP**-complete	**NP**-complete

Fig. 9. Complexity of CQA(Σ, τ) for different classes of constraints and queries. The **P** cases also have a nucleus constructible in polynomial time.

The complexity results for consistent query answering are summarized in Figure 9.

To simplify the notation, we have assumed a unirelational database containing a single relation of arity n. Nevertheless, all results extend to multirelational databases. It suffices to add relation symbols to distinguish tuples belonging to different tableaux. For example, let $\Sigma = \{\forall x \forall y\ (R(x, y) \rightarrow S(x)), \forall x \forall y\ (R(x, y) \wedge S(y) \rightarrow 0 = 1)\}$. That is, S contains every constant that occurs in the first column of R, and no constant of the second column of R. Let $I = \{R(a, b), S(b)\}$. The two fixes are $F_1 = \{R(a, b), S(x)\}$ and $F_2 = \{R(a, y), S(b)\}$. A chase of F_1 by Σ results in $G_1 = \{R(a, b), S(x), S(a)\} \sim \{R(a, b), S(a)\}$. A chase of F_2 by Σ results in $G_2 = \{R(a, y), S(a), S(b)\}$. A CQ-nucleus is $H = \{R(a, w), S(a)\}$. The consistent answer to the linear query $\langle x \rangle \leftarrow R(x, y) \wedge S(x)$ is $\{\langle a \rangle\}$.

10.2 Comparison with Related Work

10.2.1 *Defining Repairs.* Although theoretical approaches to reasoning with inconsistent information date back to the 80s, the distinction between "consistent" and "inconsistent" answers to queries on databases that violate integrity constraints, is often attributed to Bry [1997], who founded the idea on provability in minimal logic. Consistent query answering gained momentum with the advent of a model-theoretic construct of repair [Arenas et al. 1999]: a repair is a model of the integrity constraints that is "as close as possible" to the original database. Bertossi and Chomicki [2003] give an overview of database repairing and consistent query answering up to 2002. The clear distinction between possibly incorrect data and correct integrity constraints distinguishes database repairing from work on reasoning in inconsistent knowledge-bases [Arieli 2000; Grant and Subrahmanian 1995].

Nearly all model-based approaches define repairs in terms of the sets of inserted and deleted tuples. Insertions and deletions are mostly treated symmetrically [Arenas et al. 1999]. For example, a relation $I = \{\langle a \rangle\}$ has two repairs relative to the constraint $R(a) \rightarrow R(b)$: we can either delete $\langle a \rangle$, giving $U_1 = \{\}$, or insert $\langle b \rangle$, giving $U_2 = \{\langle a \rangle, \langle b \rangle\}$. An asymmetric treatment of insertions and deletions is the loosely-sound semantics introduced by Calì et al. [2003a], which minimizes (with respect to $\subseteq$) the set of deleted tuples, irrespective of the tuples to be inserted. Under these semantics, U_2 is the only repair. The wish to delete as few data as possible also governs our approach: I itself would be the only fix, and U_2 the only uprepair. In Greco et al. [2003a], the symmetric treatment of insertions and deletions can be overruled by user-defined *prioritized update rules*: the rule $\mathit{insert}(\langle b \rangle) \preceq \mathit{delete}(\langle a \rangle)$, for example, states that inserting $\langle b \rangle$ is

to be preferred over deleting $\langle a \rangle$. In Chomicki and Marcinkowski [2005a], only tuple deletions are allowed.

Most approaches minimize sets of deleted and/or inserted tuples relative to set inclusion. Some authors also consider minimization with respect to cardinality [Arieli et al. 2002, 2004; Lin and Mendelzon 1998]. Greco et al. [2003b] propose a formalism where preferences among repairs can be specified by a user-defined evaluation function.

Instead of defining a repair as a set of plain tuples, de Amo et al. [2002] distinguish between true and controversial tuples within a repair. The modal operator $\bullet$ in front of an atom indicates that the atom is controversial. For the foregoing example, the two repairs of $I = \{\langle a \rangle\}$ are $U_1' = \{\bullet\langle a \rangle\}$ and $U_2' = \{\langle a \rangle, \bullet\langle b \rangle\}$. The first repair switches $\langle a \rangle$ from true to controversial; the second repair switches $\langle b \rangle$ from false (or absent) to controversial. The use of more-than-two-valued logic for characterizing uncertainty about a tuple also appears in Arenas et al. [2000] and Arieli et al. [2002].

A distinguishing feature of our approach is that it does not treat tuples as atomic units of repairing. It is easy to think of applications where this way of repairing is more satisfactory than deletion/insertion-based repairing. Consider a patient record in a medical database, storing a large number of measurements. When we find out that some value is physiologically impossible (for example, hematocrit above 80%), probably due to an input error, then we generally do not want to delete the entire tuple nor do we wish to mark the entire tuple as controversial. Our approach allows "hiding" the faulty value, while keeping the patient record. Franconi et al. [2001] define a repair as a mapping from a possibly inconsistent relation to a new consistent relation, and thus also consider changes in attribute values as basic repair actions.

10.2.2 *Complexity Results.* So far, we found no instance of the consistent query answering problem that is known to be tractable under deletion/insertion-based repairing and intractable under update-based repairing, or *vice versa*:

Key dependency. For a single key dependency, our Theorem 7.4 imitates Theorem 3.3 in Chomicki and Marcinkowski [2005a], meaning that consistent query answering is intractable for tableau queries in both repairing paradigms. The problem becomes tractable for linear tableau queries (see our Theorem 7.3). Although we were not able to find any explicit references that a restriction to linear tableau queries also gives us tractability under deletion/insertion-based repairing (for quantifier-free tableau queries, this follows from Theorem 3.1 in Chomicki and Marcinkowski [2005a]), it is not difficult to see that the construction of Theorem 7.3 carries over to deletion/insertion-based repairing. So consistent query answering is tractable for linear tableau queries in both paradigms.

In the case of a single key dependency, update-based and deletion/insertion-based repairing yield the same consistent answers to linear tableau queries, but can differ on nonlinear queries. For example, consider the relation PERSON of Figure 10 where Name is the primary key, and the nonlinear query τ asking for persons living in their place of birth. Under deletion/insertion-based repairing,

PERSON	Name	BirthPlace	Residence
	Ed	Mons	Mons
	Ed	Bergen	Bergen

τ	Name	BirthPlace	Residence
	x	y	y
	x		

F	Name	BirthPlace	Residence
	Ed	•	Mons
	Ed	Bergen	•

J	Name	BirthPlace	Residence
	Ed	Bergen	Mons

Fig. 10.

"Ed" is in the consistent answer. However, the consistent query answer is empty under update-based repairing. To see why, notice that F is a fix generating the uprepair J.

Contradiction-generating dependencies. For cgd's, it is not difficult to adapt our Theorem 8.13 to deletion/insertion-based repairing, meaning that consistent query answering is intractable for linear tableau queries in both paradigms. On the other hand, the problem is tractable for quantifier-free tableau queries in both paradigms (see our Theorem 8.12 and Theorem 3.1 in Chomicki and Marcinkowski [2005a]).

Chomicki and Marcinkowski [2005a, 2005b] extensively discuss our previous work, and mention intractability of repair checking in our approach (see Theorem 5.3 in Chomicki and Marcinkowski [2005a]). The same result has appeared in Wijsen [2004]. It is important to mention here that this complexity result holds if fixes are allowed to contain multiple occurrences of the same variable. In Section 5, we disallowed multiple occurrences of the same variable in a fix. This restriction not only leads to more natural uprepairs in general, but also gives us tractability. So our Theorem 5.9 is not in contradiction with Theorem 5.3 reported in Chomicki and Marcinkowski [2005a].

We have noticed that it is generally a waste of time to examine more than one representative of an $\sim$-equivalence class. For space efficiency reasons, we may be interested in using a reduced representative—a representative of the smallest possible size. In general, deciding whether a tableau can be reduced such that its cardinality drops below a given size k, is an **NP**-complete problem. On the other hand, it is straightforward to minimize (with respect to size) a linear tableau in polynomial time. Fagin et al. [2003] identify more general classes of tableaux that allow a polynomial time minimization.

10.2.3 *Computing Repairs and Consistent Query Answers.* Several authors have investigated the use of logic solvers for database repairing. In Arenas et al. [2003a]; Arieli et al. [2004] and Greco et al. [2003a], the database and integrity constraints are rewritten into a logical theory that associates to each atom p a new "switch" or "update" atom (denoted p' in Arenas et al. [2003a], s_p in Arieli et al. [2004], p_u in Greco et al. [2003a]). Every model M of the new theory leads to a database repair by inserting or deleting atoms p depending on the truth value of p's "switch" atom. In Arieli et al. [2002], the role of the switch atom is played by a truth value $\top$, which is distinct from true and false: if the truth value of p is $\top$ in the three-valued model M, then p must be deleted from the database if p is in the database; otherwise p must be inserted. These

approaches thus reduce the computation of repairs to the computation of models in some logical framework for which resolution methods exist. They are not restricted to full dependencies and can be used to compute consistent answers to any first-order query.

In Arieli et al. [2004], quantifiers in integrity constraints are restricted to range over the active domain of the database (i.e. the set of constants appearing in the database), which allows quantifier elimination. In our approach, variables in fixes can be thought of as being existentially quantified, but as we do not assume active domain semantics, the number of uprepairs is infinite in general.

Theorem 4.14 in Greco et al. [2003a] implies that for full inclusion dependencies, it can be decided in polynomial time which ground tuples are in every repair under deletion/insertion-based repairing. In our framework, if Σ is a set of full tuple-generating dependencies, then a relation is its own fix and the corresponding uprepair is obtained in polynomial time by the chase.

A major motivation for our restriction to full dependencies is the termination of the chase. Nevertheless, not-necessarily-full tuple-generating dependencies are of practical importance for capturing inclusion dependencies and referential integrity in general. The chase also terminates for the class of acyclic inclusion dependencies defined in Sciore [1983]. Consistent query answering under such acyclic inclusion dependencies is studied by Bravo and Bertossi [2004]. Significantly, Fagin et al. [2002] recently showed termination of the chase for the larger class of weakly-acyclic tuple-generating dependencies, which strictly includes both full tuple-generating dependencies and acyclic inclusion dependencies.

An elegant approach to computing consistent query answers is query rewriting: a given query is rewritten such that the new query is guaranteed to return the consistent answers on any, possibly inconsistent, database. Obviously, if the rewritten query is wished to be first-order expressible [Arenas et al. 1999; Celle and Bertossi 2000], then unless $\mathbf{P} = \mathbf{NP}$, this approach is limited to sets Σ of constraints and queries τ for which consistent query answering (the complexity of $\mathsf{CQA}(\Sigma, \tau)$) is in $\mathbf{P}$.

For example, assume an SQL table containing columns K and A, where K is the primary key. Then, the SQL query:

```
SELECT DISTINCT A FROM I
```

may be rewritten as:

```
 SELECT DISTINCT I1.A FROM I AS I1 WHERE NOT EXISTS
(SELECT * FROM I AS I2 WHERE I2.K=I1.K AND I2.A<>I1.A)
```

It can be easily verified that the rewritten query on the original database gives us consistent query answers under both update-based and deletion/insertion-based repairing [Chomicki et al. 2004a]. The original query can be expressed as a linear tableau query, and the time required to answer the rewritten query is no less than the time needed for constructing a linCQ-nucleus (i.e. $\mathcal{O}(m \log m)$ where $m = |I|$, see Theorem 7.3). So in this example, nuclei may be

a useful alternative to query rewriting. The results on the nucleus construction presented in Section 4 do not rely on the actual notion of repair, meaning that nuclei can be constructed for deletion/insertion-based repairs. From the discussion earlier in this section, it follows that in the case of a single key dependency, update-based and deletion/insertion-based repairing share linCQ-nuclei, but differ on CQ-nuclei. The nucleus idea was largely unexplored until now. In Bertossi and Schwind [2002, 2004], we just found one phrase where the authors wonder whether they "can obtain an implicit and compact representation of the database repairs."

For deletion/insertion-based repairing in a multirelational setting, Fuxman and Miller [2005] give an algorithm for first-order rewriting of restricted simple conjunctive queries when there is at most one key dependency per relation, which implies tractability of consistent query answering under these conditions. A conjunctive query is simple if no relation symbol occurs more than once in it. Our query classes do not limit the number of occurrences of relation symbols, but the number of occurrences of quantified variables (0 for qfCQ, 0 or 1 for linCQ, unlimited for CQ). In particular, a linCQ query need not be simple, and a simple query need not be linear.

Chomicki et al. [2004a, 2004b] have realized a practical implementation for consistent query answering in the case of denial constraints, a generalization of cgd's. The same constructs are used in Arenas et al. [2003b] for consistent answering to aggregate queries in combination with functional dependencies. We explain the technique for a set Σ of cgd's. Define the *conflict hypergraph* of a relation I and Σ as the set $\mathbf{S} \subseteq 2^I$ such that $\mathbf{S}$ contains $\theta(B)$ whenever $(B, 0 = 1)$ is a cgd in Σ and $\theta(B) \subseteq I$ for some substitution θ. A *maximal independent set* of $\mathbf{S}$ is a maximal (with respect to $\subseteq$) subset of I that contains no element of $\mathbf{S}$ as a subset. The implementation relies on the easily verifiable property that every maximal independent set is a deletion-based repair, and *vice versa*. It is not hard to see that J is a maximal independent set of $\mathbf{S}$ if and only if $I \setminus J$ is a minimal hitting set of $\mathbf{S}$. So our characterization of fixes in terms of hitting sets is reminiscent of this work. A crucial difference, however, is that in our approach, fixes correspond to hitting sets, but the inverse is not true (see Lemma 8.9 and the discussion after it).

10.2.4 *Data Exchange.* Lemma 3.10, which appeared in Wijsen [2002] and imitates Lemma 3 in Beeri and Vardi [1984], implies that the chase of a subconsistent tableau F by a satisfiable set of full dependencies produces a consistent tableau that is homomorphic to each consistent relation I satisfying $I \succeq F$. The same property appeared at the same time in Fagin et al. [2002, Theorem 1] and is used to compute universal solutions in a data exchange setting. In that paper, the authors mention that they "have not been able to find any explicit references to the fact that the chase can produce instances that have homomorphisms to all instances satisfying the dependencies under consideration."

So, universal solutions in Fagin et al. [2002] share the same property as the tableaux obtained from chasing fixes in our framework. In both works, this property underlies the usage of chase results for query answering (see

Proposition 2 in Fagin et al. [2002] and our Lemma 3.21). A distinguished step in our framework is that in the case of multiple fixes, a nucleus is obtained as the maximal (with respect to $\succeq$) tableau homomorphic to all the tableaux obtained from chasing fixes. Note that the nucleus construction itself does not use the chase.

All approaches cited so far (including ours) assume a single database context. In data integration systems [Lembo et al. 2002; Bravo and Bertossi 2003; Calì et al. 2003b, 2004], queries are asked against a global database that results from integrating local source databases, and inconsistency arises relative to integrity constraints expressed over the global schema.

REFERENCES

ABITEBOUL, S., HULL, R., AND VIANU, V. 1995. *Foundations of Databases*. Addison-Wesley.

ARENAS, M., BERTOSSI, L. E., AND CHOMICKI, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of the 18th ACM Symposium on Principles of Database Systems*. ACM Press, 68–79.

ARENAS, M., BERTOSSI, L. E., AND CHOMICKI, J. 2003a. Answer sets for consistent query answering in inconsistent databases. *Theory and Practice of Logic Programming 3*, 3-4, 393–424.

ARENAS, M., BERTOSSI, L. E., CHOMICKI, J., HE, X., RAGHAVAN, V., AND SPINRAD, J. 2003b. Scalar aggregation in inconsistent databases. *Theor. Comput. Sci. 296*, 3, 405–434.

ARENAS, M., BERTOSSI, L. E., AND KIFER, M. 2000. Applications of Annotated Predicate Calculus to Querying Inconsistent Databases. In *Proceedings of the 1st International Conference on Computational Logic (CL 2000)*. LNAI, vol. 1861. Springer, 926–941.

ARIELI, O. 2000. An algorithmic approach to recover inconsistent knowledge-bases. In *Proceedings of the European Workshop on Logics in Artificial Intelligence (JELIA 2000)*. LNCS, vol. 1919. Springer, 148–162.

ARIELI, O., DENECKER, M., NUFFELEN, B. V., AND BRUYNOOGHE, M. 2002. Repairing inconsistent databases: A model-theoretic approach and abductive reasoning. In *Paraconsistent Computational Logic*. Datalogiske Skrifter, vol. 95. Roskilde University, Roskilde, Denmark, 51–65.

ARIELI, O., DENECKER, M., NUFFELEN, B. V., AND BRUYNOOGHE, M. 2004. Database repair by signed formulae. In *Proceedings of the 3rd International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2004)*. LNCS, vol. 2942. Springer, 14–30.

BEERI, C. AND VARDI, M. Y. 1984. A proof procedure for data dependencies. *J. ACM 31*, 4 (Oct.), 718–741.

BERTOSSI, L. E. AND CHOMICKI, J. 2003. Query answering in inconsistent databases. In *Logics for Emerging Applications of Databases*, J. Chomicki, R. van der Meyden, and G. Saake, Eds. Springer, Chapter 2, 43–83.

BERTOSSI, L. E. AND SCHWIND, C. 2002. Analytic tableaux and database repairs: Foundations. In *Proceedings of the 2nd International Symposium on Foundations on Information and Knowledge Systems (FoIKS 2002)*. LNCS, vol. 2284. Springer, 32–48.

BERTOSSI, L. E. AND SCHWIND, C. 2004. Database repairs and analytic tableaux. *Annals of Mathematics and Artificial Intelligence 40*, 1-2, 5–35.

BRAVO, L. AND BERTOSSI, L. E. 2003. Logic programs for consistently querying data integration systems. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 10–15.

BRAVO, L. AND BERTOSSI, L. E. 2004. Consistent query answering under inclusion dependencies. In *CASCON*. IBM, 202–216.

BRY, F. 1997. Query answering in information systems with integrity constraints. In *First IFIP WG 11.5 Working Conference on Integrity and Internal Control in Information Systems: Increasing the Confidence in Information Systems, Zurich, Switzerland, December 4-5, 1997*. Chapman Hall, 113–130.

CALÌ, A., CALVANESE, D., GIACOMO, G. D., AND LENZERINI, M. 2004. Data integration under integrity constraints. *Inf. Syst. 29*, 2, 147–163.

Calì, A., Lembo, D., and Rosati, R. 2003a. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proceedings of the 22nd ACM Symposium on Principles of Database Systems*. ACM, 260–271.

Calì, A., Lembo, D., and Rosati, R. 2003b. Query rewriting and answering under constraints in data integration systems. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 16–21.

Celle, A. and Bertossi, L. E. 2000. Querying inconsistent databases: Algorithms and implementation. In *Proceedings of the 1st International Conference on Computational Logic (CL 2000)*. LNAI, vol. 1861. Springer, 942–956.

Chomicki, J. and Marcinkowski, J. 2005a. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput. 197*, 1-2, 90–121.

Chomicki, J. and Marcinkowski, J. 2005b. On the computational complexity of minimal-change integrity maintenance in relational databases. In *Inconsistency Tolerance*. LNCS, vol. 3300. Springer, 119–150.

Chomicki, J., Marcinkowski, J., and Staworko, S. 2004a. Computing consistent query answers using conflict hypergraphs. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management (CIKM '04)*. ACM, 417–426.

Chomicki, J., Marcinkowski, J., and Staworko, S. 2004b. Hippo: A system for computing consistent answers to a class of SQL queries. In *Proceedings of the 9th International Conference on Extending Database Technology (EDBT 2004)*. LNCS, vol. 2992. Springer, 841–844.

de Amo, S., Carnielli, W. A., and Marcos, J. 2002. A logical framework for integrating inconsistent information in multiple databases. In *Proceedings of the 2nd International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2002)*. LNCS, vol. 2284. Springer, 67–84.

Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. 2002. Data exchange: Semantics and query answering. In *Proceedings of the 9th International Conference on Database Theory (ICDT 2003)*. LNCS, vol. 2572. Springer, 207–224.

Fagin, R., Kolaitis, P. G., and Popa, L. 2003. Data exchange: Getting to the core. In *Proceedings of the 22nd ACM Symposium on Principles of Database Systems*. ACM, 90–101. A journal version will appear in ACM Trans. Database Syst.

Franconi, E., Palma, A. L., Leone, N., Perri, S., and Scarcello, F. 2001. Census data repair: A challenging application of disjunctive logic programming. In *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2001)*. LNCS, vol. 2250. Springer, 561–578.

Fuxman, A. D. and Miller, R. J. 2005. First-order rewriting for inconsistent databases. In *Proceedings of the 10th International Conference on Database Theory (ICDT 2005)*. LNCS, vol. 3363. Springer, 337–351.

Grant, J. and Subrahmanian, V. S. 1995. Reasoning in inconsistent knowledge bases. *IEEE Trans. Knowl. Data Eng. 7*, 1, 177–189.

Greco, G., Greco, S., and Zumpano, E. 2003a. A logical framework for querying and repairing inconsistent databases. *IEEE Trans. Knowl. Data Eng. 25*, 6, 1389–1408.

Greco, S., Sirangelo, C., Trubitsyna, I., and Zumpano, E. 2003b. Preferred repairs for inconsistent databases. In *7th International Database Engineering and Applications Symposium (IDEAS 2003)*. IEEE Computer Society, 202–211.

Hell, P. and Zhu, X. 1995. The existence of homomorphisms to oriented cycles. *SIAM J. Discrete Math. 8*, 2, 208–222.

Lembo, D., Lenzerini, M., and Rosati, R. 2002. Source inconsistency and incompleteness in data integration. In *Proceedings of the 9th International Workshop on Knowledge Representation Meets Databases (KRDB 2002)*. Number 54 in CEUR Workshop Proceedings. Technical University of Aachen (RWTH).

Lin, J. and Mendelzon, A. O. 1998. Merging databases under constraints. *Int. J. Cooperative Inf. Syst. 7*, 1, 55–76.

Nair, M. 1982a. A new method in elementary prime number theory. *J. London Math. Soc. 25*, 385–391.

Nair, M. 1982b. On Chebyshev-type inequalities for primes. *Amer. Math. Monthly 89*, 126–129.

PLOTKIN, G. D. 1969. A note on inductive generalization. In *Machine Intelligence 5*, B. Meltzer and D. Michie, Eds. Edinburgh University Press, Edinburgh, 153–163.

SCIORE, E. 1983. Inclusion dependencies and the universal instance. In *Proceedings of the 2nd ACM Symposium on Principles of Database Systems*. ACM, 48–57.

WIJSEN, J. 2002. Condensed representation of database repairs for consistent query answering. In *Proceedings of the 9th International Conference on Database Theory (ICDT 2003)*. LNCS, vol. 2572. Springer, 378–393.

WIJSEN, J. 2004. Making more out of an inconsistent database. In *Proceedings of the 8th East European Conference on Advances in Databases and Information Systems (ADBIS 2004)*. LNCS, vol. 3255. Springer, 291–305.

Received April 2004; revised January 2005; accepted April 2005