Information Systems 34 (2009) 578–601

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

On the consistent rewriting of conjunctive queries under primary key constraints

Jef Wijsen

Université de Mons, Place du Parc 20, B-7000 Mons, Belgium

ARTICLE INFO

Keywords:
Certain query answering
Consistent query answering
Database repairing

ABSTRACT

This article deals with the computation of consistent answers to queries on relational databases that violate primary key constraints. A repair of such inconsistent database is obtained by selecting a maximal number of tuples from each relation without ever selecting two distinct tuples that agree on the primary key. We are interested in the following problem: Given a Boolean conjunctive query q, compute a Boolean first-order (FO) query ψ such that for every database **db**, ψ evaluates to true on **db** if and only if q evaluates to true on every repair of **db**. Such ψ is called a consistent FO rewriting of q.

We use novel techniques to characterize classes of queries that have a consistent FO rewriting. In this way, we are able to extend previously known classes and discover new ones. Finally, we use an Ehrenfeucht–Fraïssé game to show the non-existence of a consistent FO rewriting for $\exists x \exists y(R(\underline{x}, y) \wedge R(\underline{y}, c))$, where c is a constant and the first coordinate of R is the primary key.

1. Introduction

Consistent query answering (CQA) was introduced by Arenas et al. [1] and has gained considerable interest in recent years; see for example the invited talk by Chomicki [2]. The aim of CQA is to get consistent information out of inconsistent databases. In technical terms, the repairs of an inconsistent database **db** are defined as the consistent databases that can be obtained from **db** by some minimal change. If, as in this article, the constraints are primary keys, then it is natural to take as repairs the maximal consistent subsets of **db**. Given a Boolean query q, the problem then is to decide whether q evaluates to true on every repair of **db**.

For example, the relation EMP in Fig. 1, which violates the primary key Name, has two repairs, each containing one tuple. The query $\exists y(\text{EMP}(\underline{\text{Blake}}, y, 10\text{K}))$ evaluates to true on both repairs, so "*Blake earns* 10*K*" is accepted as a consistent piece of information. On the other hand, $\exists z(\text{EMP}(\underline{\text{Blake}}, \text{Paris}, z))$ is not true in every repair, so we cannot be sure that Blake lives in Paris.

We deal with conjunctive queries in this article. For a fixed Boolean conjunctive query q, CQA(q) is the following problem: On input of a not-necessarily-consistent database **db**, decide whether q evaluates to true on every repair of **db**. It is by now well known (see for example [2]) that CQA(q_1) is **coNP**-complete for the following Boolean query q_1:

$$q_1 : \exists x \exists y \exists z(R(\underline{x}, z) \wedge S(\underline{y}, z)),$$

where primary key positions are underlined. On the other hand, CQA(q_2) is in **P** for the following query q_2 [3]:

$$q_2 : \exists x \exists y \exists z(R(\underline{x}, z) \wedge S(\underline{z}, y)).$$

The different computational behavior arises because the "join" variable z (i.e. the variable common to both atoms) constitutes a primary key in the second query, but not in the first one.

Fuxman and Miller [3] showed that for every query q in some syntactically restricted class, called C_{forest}, there exists a computable Boolean first-order (FO) query ψ such

E-mail address: wijsen@umh.ac.be

0306-4379/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2009.03.011

EMP	Name	City	Sal
	Blake	Paris	10K
	Blake	London	10K

Repair1	Name	City	Sal
	Blake	Paris	10K

Repair2	Name	City	Sal
	Blake	London	10K

Fig. 1. Relation with two repairs.

that for every database **db**, q evaluates to true on every repair of **db** if and only if ψ evaluates to true on **db**. We call such ψ a *consistent FO rewriting* of q. Clearly, if q has a consistent FO rewriting ψ, then CQA(q) is in **P** (because ψ can be evaluated in polynomial time on any database). For the query q_2, a consistent FO rewriting is

$$\psi_2 : \quad \exists x \exists z' (R(\underline{x}, z') \wedge \forall z (R(\underline{x}, z) \rightarrow \exists y (S(\underline{z}, y))))).$$

Intuitively, ψ_2 checks whether for all R-tuples with primary key value x, there exists a joining tuple in S.

Query rewriting is a clean and elegant approach to consistent query answering. This article presents a number of new results in this field; its main contributions can be summarized as follows:

1. We define the class of *key-rooted*[1] Boolean conjunctive queries and give a rewrite function that computes a consistent FO rewriting for *every* query in this class. The function consists of two rewrite rules. The class of key-rooted queries seems to be large: we are unaware of Boolean conjunctive queries that are FO rewritable and not key-rooted (but we have no formal proof that no such query can exist).
2. As the notion of key-rooted queries is a semantical one, the task then is to define syntactic restrictions on queries that guarantee "key-rootedness" (and hence guarantee applicability of our rewrite function). The advantage of our approach is that the notion of key-rootedness hides the syntactical intricacies that complicate FO rewriting.
   Instead of Fuxman Miller (FM) join graphs, we use the join trees defined by Beeri, Fagin, Maier and Yannakakis [5], called BFMY join trees hereafter. This technique allows us to characterize new, previously unknown classes of queries with a consistent FO rewriting (some of which have cyclic FM join graphs, but acyclic BFMY join trees).
3. We pay special attention to queries with multiple occurrences of the same relation name, a class for which consistent FO rewriting was largely unexplored until now. For the query

   $$q = \exists x \exists y \exists z (R(\underline{x}, z) \wedge R(\underline{y}, z) \wedge x \neq y),$$

   it is known that CQA(q) is in **P** but q has no consistent FO rewriting [6]. We show that the same holds for the query $q = \exists x \exists y (R(\underline{x}, y) \wedge R(\underline{y}, c))$, where c is a constant. This result is surprising, since the join variable y appears as primary key. It indicates that consistent FO rewriting easily fails for conjunctive queries with self-joins (i.e. in which the same relation name occurs more than once).

This article is organized as follows. The next section introduces the notations and terminology used throughout the article. In particular, the term "rule" will be used as a shorthand for "Boolean conjunctive query." Section 3 discusses related work. Section 4 defines the model-theoretic class of key-rooted rules. Section 5 gives a rewrite function that computes a consistent FO rewriting for any key-rooted rule. Section 6 deals with different kinds of join trees and exhibits some useful properties. Section 7 characterizes classes of key-rooted rules in terms of BFMY join trees. Section 8 shows that for the query $q = \exists x \exists y (R(\underline{x}, y) \wedge R(\underline{y}, c))$, CQA($q$) is in **P** but q has no consistent FO rewriting. Section 9 shows how to deal with conjunctive queries with free variables. Section 10 concludes the article. Some lengthy proofs and helping lemmas have been moved to the Appendix.

2. Notations and terminology

A *symbol* is either a constant or a variable. Let X be a set of variables. A *valuation* over X is a mapping θ from X to constants; the mapping θ is extended to all symbols as follows: if s is a variable that does not occur in X or if s is a constant, then $\theta(s) = s$.

If $\vec{x}$ is a sequence of symbols, then vars($\vec{x}$) is the set of variables that occur in $\vec{x}$. A *valuation of* $\vec{x}$ is a valuation over vars($\vec{x}$).

Key-equal atoms: A *database schema* is a finite set of *relation names*. Every relation name R has a unique *signature*, which is a pair $[n, k]$ with $n \geqslant k \geqslant 1$: n is the *arity* of the relation name and the coordinates $1, 2, \ldots, k$ make up the *primary key*. If R is a relation name with signature $[n, k]$, then $R(s_1, \ldots, s_n)$ is an *R-atom* (or simply atom), where each s_i is a constant or a variable ($1 \leqslant i \leqslant n$). Such an atom is commonly written as $R(\underline{\vec{x}}, \vec{y})$ where $\vec{x} = s_1, \ldots, s_k$ and $\vec{y} = s_{k+1}, \ldots, s_n$. An atom is *ground* if it contains no variables. All constructs that follow are defined relative to a fixed database schema.

A *database* is a finite set I of ground atoms using only the relation names of the schema. Two ground atoms $R_1(\underline{\vec{a}_1}, \vec{b}_1), R_2(\underline{\vec{a}_2}, \vec{b}_2) \in I$ are *key-equal* if $R_1 = R_2$ and $\vec{a}_1 = \vec{a}_2$. We write $[\![R_1(\underline{\vec{a}_1}, \vec{b}_1)]\!]_I$ for the set containing each atom of I that is key-equal to $R_1(\underline{\vec{a}_1}, \vec{b}_1)$. This notation extends naturally to subsets $J \subseteq I$, as follows: $[\![J]\!]_I = \bigcup \{ [\![A]\!]_I \mid A \in J \}$.

Repair: A database I is *consistent* if it does not contain two distinct atoms that are key-equal. Thus, I is consistent if for every atom $A \in I$, $[\![A]\!]_I = \{A\}$. A *repair* of a database I is a maximal (under set inclusion) consistent subset J of I.

Non-ordered and ordered rules: As in [7, p. 41], the term *rule* will be used as a shorthand for *rule-based conjunctive query*. Moreover, all rules are understood to be Boolean. Thus we have the following definitions.

[1] In [4], these queries were called rooted. We prefer the term key-rooted in this article, to avoid confusion with the construct of rooted tree that will also be used.

A (*non-ordered*) *rule* is a finite set

$$q = \{R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\}$$

of atoms. This rule is satisfied by a database I, denoted $I \models q$, if there exists a valuation θ of $\vec{x}_1\vec{y}_1 \cdots \vec{x}_m\vec{y}_m$ such that for each $i \in \{1, \ldots, m\}$, $R_i(\theta(\underline{\vec{x}_i}), \theta(\vec{y}_i)) \in I$.

We say that a rule q has a *self-join* if two distinct atoms of q share the same relation name.

In several places of the technical development, the order in which atoms of a rule are listed is significant. In particular, the syntactic rewrite function of Section 5 processes the atoms of a rule from left to right. Therefore, an *ordered rule* of *length* m is a sequence

$$q_o = \langle R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\rangle$$

of (not necessarily distinct) atoms. The ith atom of q_o, $1 \leqslant i \leqslant m$, is denoted $q_o[i]$, that is, $q_o[i] = R_i(\underline{\vec{x}_i}, \vec{y}_i)$. The first atom $R_1(\underline{\vec{x}_1}, \vec{y}_1)$ is called the *prefix* of the rule, and $\langle R_2(\underline{\vec{x}_2}, \vec{y}_2), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\rangle$ the *tail*. The length m of q_o is denoted by $|q_o|$. Satisfaction of ordered rules is defined in the same way as for non-ordered rules. In general, any construct that is defined for non-ordered rules naturally carries over to ordered rules by ignoring the order and by eliminating duplicates (if any).

Notice that rules contain no built-in predicates.

Consistently true: A rule q is *consistently true* in I, denoted $I \models_{\mathsf{sure}} q$, if for every repair J of I, $J \models q$. The problem $\mathsf{CQA}_S(q)$, where S is a database schema and q is a rule, is the complexity of (testing membership of) the set:

$$\mathsf{CQA}_S(q) = \{I \mid I \text{ is a database over } S \text{ and } I \models_{\mathsf{sure}} q\}.$$

Throughout this article, the schema S will be implicitly understood and therefore omitted.

Consistent FO rewriting: We say that a Boolean FO query ψ is a *consistent FO rewriting* of a rule q if for every database I, $I \models_{\mathsf{sure}} q$ if and only if $I \models \psi$. Thus, q has a consistent FO rewriting if and only if $\mathsf{CQA}(q)$ is first-order definable.

3. Related work

The repairs defined above are maximal consistent subsets of the original database. In the case of primary keys, it makes no difference whether maximality is expressed relative to set inclusion (as in [1]) or cardinality (as in [8]). Inserting new tuples is useless for restoring primary key violations. Tuple modifications, as proposed in [9], are not considered in this article.

The idea of consistent query rewriting first appeared in [1]. Fuxman and Miller [3] have made a number of breakthroughs in the consistent FO rewriting of rules under primary key constraints, which motivated the ConQuer system [10]. Their results have been generalized and extended to exclusion dependencies by Grieco et al. [11] and to unions of conjunctive queries by Lembo et al. [12].

Up to Section 9, we limit our attention to Boolean queries. Thus, all variables of a rule are understood to be implicitly existentially quantified. The definition of

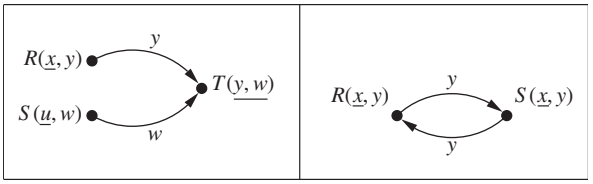

Fig. 2. FM join graphs of $\{R(\underline{x}, y), S(\underline{u}, w), T(\underline{y}, \underline{w})\}$ and $\{R(\underline{x}, y), S(\underline{x}, y)\}$.

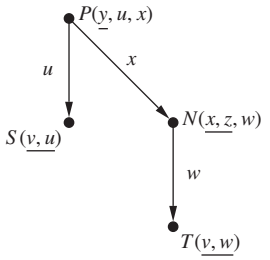

Fig. 3. FM join graph of $\{P(\underline{y}, u, x), S(\underline{v}, \underline{u}), N(\underline{x}, \underline{z}, w), T(\underline{v}, \underline{w})\}$.

Fuxman Miller join graph first appeared in [13] and was slightly adapted in [3].

FM join graph: The *FM join graph* of a (Boolean) rule $q = \{R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\}$ is a directed graph whose vertices are the atoms of q; there is a directed edge from $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ to $R_j(\underline{\vec{x}_j}, \vec{y}_j)$ if $i \neq j$ and $\mathsf{vars}(\vec{y}_i) \cap \mathsf{vars}(\vec{x}_j\vec{y}_j) \neq \{\}$.

It is common to label an edge from $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ to $R_j(\underline{\vec{x}_j}, \vec{y}_j)$ in the FM join graph of q with the (nonempty) set of variables that occur in both $\vec{y}_i$ and $\vec{x}_j\vec{y}_j$.

Fig. 2 shows two FM join graphs; neither is a directed tree (the left graph has a vertex with two incoming edges). Fig. 3 shows an FM join graph that is a directed tree.

Fuxman and Miller [3] give an algorithm that computes a consistent FO rewriting for any rule $q = \{R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\}$ with the following properties:

1. $1 \leqslant i < j \leqslant m$ implies $R_i \neq R_j$. Thus, no relation name occurs more than once in q;
2. the FM join graph of q is a directed forest; and
3. if there is a directed edge from atom $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ to $R_j(\underline{\vec{x}_j}, \vec{y}_j)$, then $\mathsf{vars}(\vec{x}_j) \subseteq \mathsf{vars}(\vec{y}_i)$. Using the terminology of Fuxman and Miller [3], an edge from $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ to $R_j(\underline{\vec{x}_j}, \vec{y}_j)$ implies a *full nonkey-to-key join* from $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ to $R_j(\underline{\vec{x}_j}, \vec{y}_j)$.

This class of rules is called C_{forest}. In this article, we denote by C_{tree} the subclass of C_{forest} that contains $q \in C_{forest}$ whenever the FM join graph of q is a (connected) directed tree.[2] Lemma 2 in [3] implies that whenever two distinct C_{tree} components q_1 and q_2 of a C_{forest} query share a variable x, then x can only occur in the primary keys of the root atoms of q_1 and q_2.

[2] Caveat: This class is not the same as the class C_{tree} in Fuxman and Miller's original conference article [13]. The definition of C_{tree} in [13] does not require that the Fuxman Miller join graph be connected.

The class C^{+}_{tree}, defined by Grieco et al. [11], omits the third condition in the definition of C_{forest}, that is, nonkey-to-key joins need not be full in C^{+}_{tree}. Fuxman and Miller give a query q without self-join such that the FM join graph of q is a directed forest and CQA(q) is **coNP**-hard [3]. Proposition 1 slightly modifies that query such that its FM join graph becomes a tree, while maintaining intractability. It follows that under the assumption $\mathbf{P}\neq\mathbf{NP}$, not all queries in C^{+}_{tree} have a consistent FO rewriting. Notice that in the rule of Proposition 1, the atom $P(\underline{y},u,x)$ contains some, but not all, variables that occur in the primary key of $S(\underline{v,u})$. Hence, the rule contains a nonkey-to-key join that is not full.

Proposition 1. *If* $q=\{P(\underline{y},u,x),\ S(\underline{v,u}),\ N(\underline{x,z},w),\ T(\underline{v,w})\}$, *then* CQA($q$) *is* **coNP**-*hard*.

Complexity results on consistent query answering for larger classes of constraints appear in [14,15].

4. Key-rooted rules

We define the model-theoretic notion of key-rooted rule. Key-rootedness is based on the following model-theoretic property which can be easily verified. Let I be a database. An ordered rule q with prefix $R_1(\underline{\vec{x}_1},\vec{y}_1)$ and tail q' is true in every repair of I if the following condition is satisfied:

> there exists a valuation θ of $\vec{x}_1$ (let $\theta(\vec{x}_1)=\vec{a}$) such that I contains an R_1-atom with primary key value $\vec{a}$ and for every such atom $R_1(\underline{\vec{a}},\vec{b})\in I$, there exists a valuation $\theta_{\vec{b}}$ of $\vec{x}_1\vec{y}_1$ such that $\theta_{\vec{b}}(\vec{x}_1\vec{y}_1)=\vec{a}\vec{b}$ and $\theta_{\vec{b}}(q')$ is true in every repair of I that contains $R_1(\underline{\vec{a}},\vec{b})$.

The above condition deals with all ways to repair multiple R_1-atoms with the same primary key value $\vec{a}$. There are two points to observe:

1. If such valuation θ exists, then $\theta(q)$ is also true in every repair of I.
2. Consistent truth of q is reduced to consistent truth of shorter rules $\theta_{\vec{b}}(q')$ for every $R(\underline{\vec{a}},\vec{b})\in I$.

The above condition is sufficient for $I\models_{\mathsf{sure}} q$. Key-rootedness will be defined in such a way that every key-rooted ordered rule q satisfies that condition whenever $I\models_{\mathsf{sure}} q$. Observation (1) leads to the notion of "reifiability" (Definition 1); observation (2) to a recursive definition of key-rooted ordered rules (Definition 2). Lemma 1 provides a sufficient condition for reifiability that will be used in some proofs later on. Proposition 2 then indicates that the class of key-rooted rules is of practical interest: it encompasses the class C_{tree}, which contains many common, practical queries [13]. Moreover, as we will see later on, it covers relevant queries not in C_{forest}, such as the "intersection" query $\exists\vec{x}\exists\vec{y}(R(\underline{\vec{x}},\vec{y})\wedge S(\underline{\vec{x}},\vec{y}))$, where R and S have the same signature.

Definition 1. Let q be a rule containing $R(\underline{\vec{x}},\vec{y})$. We call $R(\underline{\vec{x}},\vec{y})$ *reifiable in* q if for every database I, if $I\models_{\mathsf{sure}} q$, then there exists a valuation θ of $\vec{x}$ such that $I\models_{\mathsf{sure}}\theta(q)$.

For a consistent database J, we define:

$$\mathsf{Reifies}(q,\vec{x},J)=\{\theta\mid\theta\text{ valuation of }\vec{x},J\models\theta(q)\}.$$

Example 1. We show that $S(\underline{y})$ is not reifiable in $q=\{S(\underline{y}),R(\underline{x},y)\}$. Let $I=\{S(\underline{b}),\ S(\underline{c}),\ R(\underline{a},b),\ R(\underline{a},c)\}$. The two repairs of I are J_1 and J_2:

$$J_1=\{S(\underline{b}),S(\underline{c}),R(\underline{a},b)\}$$
$$J_2=\{S(\underline{b}),S(\underline{c}),R(\underline{a},c)\}$$

Since $J_1\models q$ and $J_2\models q$, we have $I\models_{\mathsf{sure}} q$. However, there is no constant e such that $I\models_{\mathsf{sure}}\{S(\underline{e}),R(\underline{x},e)\}$.

On the other hand, from the results in this article, it will follow that $R(\underline{x},y)$ is reifiable in $q=\{S(\underline{y}),R(\underline{x},y)\}$.

Example 2. Neither $R(\underline{x},y)$ nor $R(\underline{y},c)$ is reifiable in $q=\{R(\underline{x},y),R(\underline{y},c)\}$. Consider the database $I=\{R(\underline{a},b),\ R(\underline{b},c),R(\underline{b},d),R(\underline{d},c)\}$. The two repairs of I are J_1 and J_2:

$$J_1=\{R(\underline{a},b),R(\underline{b},c),R(\underline{d},c)\}$$
$$J_2=\{R(\underline{a},b),R(\underline{b},d),R(\underline{d},c)\}$$

Clearly, $J_1\models q$ and $J_2\models q$. However, there exists no constant e such that J_1 and J_2 both satisfy $\{R(\underline{e},y),R(\underline{y},c)\}$. Likewise, there exists no constant f such that J_1 and J_2 both satisfy $\{R(\underline{x},f),R(\underline{f},c)\}$.

Example 3. Let $q=\{R(\underline{x},y),R(\underline{y},z)\}$. Let $J=\{R(\underline{a},b),\ R(\underline{b},c),\ R(\underline{c},d),\ R(\underline{e},e)\}$. Then, $\mathsf{Reifies}(q,x,J)$ contains the following valuations of x: $\{x\mapsto a\}$, $\{x\mapsto b\}$, and $\{x\mapsto e\}$. $\mathsf{Reifies}(q,x,J)$ does not contain $\{x\mapsto c\}$, because $J\not\models\{R(\underline{c},y),R(\underline{y},z)\}$.

Let q be a rule containing atom $R(\underline{\vec{x}},\vec{y})$. Let I be a database. For every repair J of I, there can be zero, one or more valuations θ of $\vec{x}$ such that $J\models\theta(q)$. The following question arises: Is there a repair J such that for every valuation ω of $\vec{x}$, $\omega(q)$ is satisfied by J only if $\omega(q)$ is satisfied by *every* repair of I? Lemma 1 states that if the answer to this question is "yes" for every database I, then $R(\underline{\vec{x}},\vec{y})$ is reifiable in q.

Lemma 1. *Let* q *be a rule containing* $R(\underline{\vec{x}},\vec{y})$. *The atom* $R(\underline{\vec{x}},\vec{y})$ *is reifiable in* q *if for each database* I, *for all repairs* J_1,J_2 *of* I, *there exists a repair* J *of* I *such that* $\mathsf{Reifies}(q,\vec{x},J)\subseteq\mathsf{Reifies}(q,\vec{x},J_1)\cap\mathsf{Reifies}(q,\vec{x},J_2)$.

Proof. Let I be a database. Assume that for all repairs J_1,J_2 of I, there exists a repair J of I such that $\mathsf{Reifies}(q,\vec{x},J)\subseteq\mathsf{Reifies}(q,\vec{x},J_1)\cap\mathsf{Reifies}(q,\vec{x},J_2)$. We can assume a repair J satisfying $\mathsf{Reifies}(q,\vec{x},J)=\bigcap\{\mathsf{Reifies}(q,\vec{x},J')\mid J'\text{ repair of }I\}$; the latter intersection is finite, since the number of repairs is finite. We distinguish two cases:

- $\mathsf{Reifies}(q,\vec{x},J)\neq\{\}$. Then, we can assume a valuation θ of $\vec{x}$ such that for each repair J' of I, $J'\models\theta(q)$. It follows $I\models_{\mathsf{sure}}\theta(q)$.
- $\mathsf{Reifies}(q,\vec{x},J)=\{\}$. Then, for every valuation θ of $\vec{x}$, $J\not\models\theta(q)$. It follows $J\not\models q$. Since J is a repair of I, $I\not\models_{\mathsf{sure}} q$.

Since I is arbitrary, it follows that $R(\underline{\vec{x}},\vec{y})$ is reifiable in q. □

We now define the conditions for an ordered rule to be key-rooted: the rule's prefix must be reifiable, and its tail

must be key-rooted under *every* valuation of its prefix. To get the recursion off the ground, the empty rule is key-rooted. Notice that this definition is *not* relative to a given database.

Definition 2. We define *key-rooted* ordered rules:

1. The empty rule is key-rooted.
2. The ordered rule $q = \langle R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\rangle$ with $m \geqslant 1$ is key-rooted if
   (a) $R_1(\underline{\vec{x}_1}, \vec{y}_1)$ is reifiable in q; and
   (b) for each valuation θ of $\vec{x}_1\vec{y}_1$, the ordered rule $\theta(q')$ is key-rooted where q' is the tail of q.

A non-ordered rule is called *key-rooted* if it is key-rooted under some linear ordering of its atoms.

The following proposition will serve in certain examples. It is subsumed by more general theorems to follow.

Proposition 2.

1. *If* $|q| = 1$, *then* q *is key-rooted.*
2. *The ordered rule* $\langle R_1(\underline{\vec{a}}, \vec{y}_1), R_2(\underline{\vec{x}_2}, \vec{y}_2)\rangle$, *where* $\vec{a}$ *contains no variables, is key-rooted (possibly* $R_1 = R_2$*).*
3. *Let* q *be a (non-ordered) rule in* C_{tree}, *and let* τ *be its FM join tree. Let*

   $q_o = \langle R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\rangle$

   be the ordered rule obtained from q *by listing its atoms in increasing depth. Thus, if* τ *contains a directed edge from* $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ *to* $R_j(\underline{\vec{x}_j}, \vec{y}_j)$, *then* $i<j$. *Then,* q_o *is key-rooted.*

Proof. The first item is subsumed by Theorem 4. The second item is subsumed by Theorem 2. The third item follows from the proof of Corollary 5. □

Lemma 2 shows that for every key-rooted ordered rule q, $\mathsf{CQA}(q)$ is in **P**. The proof characterizes $\mathsf{CQA}(q)$ as the set of databases satisfying a property, called Property FO, which can be checked in polynomial time. As was to be expected, Property FO expresses the condition that motivated the definition of key-rootedness (see first paragraph of Section 4). Significantly, we will show in Section 5 that Property FO is first-order expressible, which thus gives us a consistent FO rewriting for any key-rooted rule.

Lemma 2. *If* q *is a key-rooted ordered rule, then* $\mathsf{CQA}(q)$ *is in* **P**.

Proof. Let $q = \langle R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\rangle$ be a key-rooted ordered rule. If $m = 0$, then the desired result is obvious. Next assume $m \geqslant 1$.

Let I be a database such that $I \models_{\mathsf{sure}} q$. Since $R_1(\underline{\vec{x}_1}, \vec{y}_1)$ is reifiable in q, we can assume the existence of a valuation θ of $\vec{x}_1$ such that $I \models_{\mathsf{sure}} \theta(q)$. Assume w.l.o.g. that $\theta(\vec{x}_1) = \vec{a}$. Then, there exists an atom $R_1(\underline{\vec{a}}, \vec{b}') \in I$ such that every repair J of I contains exactly one atom of $[\![R_1(\underline{\vec{a}}, \vec{b}')]\!]_I$.

Let $R_1(\underline{\vec{a}}, \vec{b}) \in I$ be key-equal to $R_1(\underline{\vec{a}}, \vec{b}')$. Let J be a repair of I such that $R_1(\underline{\vec{a}}, \vec{b}) \in J$. Since $J \models \theta(q)$, it follows that there exists a valuation $\theta_{\vec{b}}$ of $\vec{x}_1\vec{y}_1$ such that $\theta_{\vec{b}}(\vec{x}_1) = \theta(\vec{x}_1) = \vec{a}$, $\theta_{\vec{b}}(\vec{y}_1) = \vec{b}$, and $J \models \theta_{\vec{b}}(q)$. Clearly, if J is a repair of I such that $R_1(\underline{\vec{a}}, \vec{b}) \in J$, then J is a repair of $(I \setminus [\![R_1(\underline{\vec{a}}, \vec{b}')]\!]_I) \cup \{R_1(\underline{\vec{a}}, \vec{b})\}$; and the inverse is also true. It follows $(I \setminus [\![R_1(\underline{\vec{a}}, \vec{b}')]\!]_I) \cup \{R_1(\underline{\vec{a}}, \vec{b})\} \models_{\mathsf{sure}} \theta_{\vec{b}}(q)$. Hence, if $I \models_{\mathsf{sure}} q$, then the following condition holds:

Property FO. For some atom $R_1(\underline{\vec{a}}, \vec{b}') \in I$, for every key-equal atom $R_1(\underline{\vec{a}}, \vec{b}) \in I$, there exists a valuation $\theta_{\vec{b}}$ of $\vec{x}_1\vec{y}_1$ such that $\theta_{\vec{b}}(\vec{x}_1\vec{y}_1) = \vec{a}\vec{b}$ and

$$(I \setminus [\![R_1(\underline{\vec{a}}, \vec{b}')]\!]_I) \cup \{R_1(\underline{\vec{a}}, \vec{b})\} \models_{\mathsf{sure}} \theta_{\vec{b}}(q'), \tag{1}$$

where q' is the tail of q.

It is easy to see that Property FO is also sufficient for $I \models_{\mathsf{sure}} q$. The quintessence now is that $\theta_{\vec{b}}(q')$ is key-rooted by Definition 2. That is, we have reduced the test $I \models_{\mathsf{sure}} q$ to tests of the form $I_0 \models_{\mathsf{sure}} q_0$ where $I_0 \subseteq I$ and $|q_0| = |q| - 1$. For every query of smaller length, its query prefix can be mapped to at most $|I|$ different atoms. For a database I that is ordered by primary key values, the overall complexity for testing $I \in \mathsf{CQA}(q)$ is $O(|I|^m)$ where $m = |q|$. □

5. Consistent first-order rewriting of key-rooted ordered rules

We show that if q is a key-rooted rule, then we can construct a FO formula ψ_q that checks membership of $\mathsf{CQA}(q)$; that is, for every database I, $I \models_{\mathsf{sure}} q$ if and only if $I \models \psi_q$. The formula ψ_q is essentially nothing else than a first-order encoding of Property FO in the proof of Lemma 2.

To start with a simple example, consider the singleton rule $q_0 = R_1(\underline{a}, b)$, which is obviously key-rooted because it contains no variables. To ease the technical treatment, we encode this rule as $R_1(\underline{x}, y) \wedge \varphi$ where $\varphi = (x = a) \wedge (y = b)$. The following formula starts encoding Property FO:

$$\exists x \exists y'(R_1(\underline{x}, y') \wedge \forall y(R_1(\underline{x}, y) \rightarrow \mathsf{Rewrite}(\varphi))).$$

Intuitively, "for some atom $R_1(\underline{\vec{a}}, \vec{b}') \in I$" is encoded by $\exists x \exists y'(R_1(\underline{x}, y') \wedge \cdots)$, and "for every key-equal atom $R_1(\underline{\vec{a}}, \vec{b}) \in I$" is encoded by $\forall y(R_1(\underline{x}, y) \rightarrow \cdots)$. Since the formula φ is a conjunction of equalities, its truth is database independent; it will be defined that $\mathsf{Rewrite}(\varphi) = \varphi$. Thus, we obtain the following consistent FO rewriting for q_0:

$$\exists x \exists y'(R_1(\underline{x}, y') \wedge \forall y(R_1(\underline{x}, y) \rightarrow (x = a) \wedge (y = b))),$$

which is equivalent to

$$\exists y'(R_1(\underline{a}, y') \wedge \forall y(R_1(\underline{a}, y) \rightarrow (y = b))).$$

As a follow-up example, consider the ordered rule $q_1 = \langle R_1(\underline{a}, y), R_2(\underline{x}, y)\rangle$, which is key-rooted by Proposition 2. Note incidentally that this rule has a cyclic FM join graph and hence does not belong to C_{tree}. We first write

this rule as

$R_1(\underline{x_1}, y_1) \wedge R_2(\underline{x_2}, y_2) \wedge \varphi$,

where

$\varphi = (x_1 = a) \wedge (y_1 = y_2)$.

We proceed as in the first example:

$$\exists x_1 \exists y_1'(R_1(\underline{x_1}, y_1') \wedge \forall y_1(R_1(\underline{x_1}, y_1) \rightarrow \mathsf{Rewrite}(R_2(\underline{x_2}, y_2) \wedge \varphi))) \quad (2)$$

A subtlety to note is that in Eq. (1) of Property FO, $\theta_{\vec{b}}(q')$ must be true, not in every repair of I, but in every repair of I *that contains* $R_1(\vec{a}, \vec{b})$. So we have to distinguish two cases:

- If $R_2 \neq R_1$, then $\mathsf{Rewrite}(R_2(\underline{x_2}, y_2) \wedge \varphi)$ can be computed as before, because $R_2(\underline{x_2}, y_2) \wedge \varphi$ is true in every repair of I *that contains* $R_1(\underline{x_1}, y_1)$ if and only if $R_2(\underline{x_2}, y_2) \wedge \varphi$ is true in every repair of I:

$$\begin{aligned}&\mathsf{Rewrite}(R_2(\underline{x_2}, y_2) \wedge \varphi)\\ &\quad = \exists x_2 \exists y_2'(R_2(\underline{x_2}, y_2') \wedge\\ &\qquad \forall y_2(R_2(\underline{x_2}, y_2) \rightarrow (x_1 = a) \wedge (y_1 = y_2))).\end{aligned}$$

- On the other hand, if $R_2 = R_1$, then it becomes significant that the formula $\mathsf{Rewrite}(R_2(\underline{x_2}, y_2) \wedge \varphi)$ must be true in I if and only if $R_2(\underline{x_2}, y_2) \wedge \varphi$ is true in each repair of I *that contains* $R_1(\underline{x_1}, y_1)$. This yields two cases: if $x_2 \neq x_1$, then we proceed as before; if $x_2 = x_1$, then $R_2(\underline{x_2}, y_2)$ must be identified with $R_1(\underline{x_1}, y_1)$. Thus, for $R_2 = R_1$, we obtain

$$\begin{aligned}&\mathsf{Rewrite}(R_2(\underline{x_2}, y_2) \wedge \varphi)\\ &\quad = \exists x_2 \exists y_2'((x_2 \neq x_1) \wedge R_2(\underline{x_2}, y_2') \wedge\\ &\qquad \forall y_2(R_2(\underline{x_2}, y_2) \rightarrow (x_1 = a) \wedge (y_1 = y_2)))\\ &\qquad\qquad \vee\\ &\qquad\qquad \exists x_2 \exists y_2 \begin{pmatrix} & (x_2 = x_1)\\ \wedge & (y_2 = y_1)\\ \wedge & (x_1 = a)\\ \wedge & (y_1 = y_2)\end{pmatrix}.\end{aligned}$$

  In this particular example, the second disjunct is equivalent to simply $(x_1 = a)$ and is implied by the first disjunct. Hence, if $R_2 = R_1$, then the formula $\mathsf{Rewrite}(R_2(\underline{x_2}, y_2) \wedge \varphi)$ is equivalent to $(x_1 = a)$. When we substitute this result in formula (2), we find a formula equivalent to

  $\exists x_1 \exists y_1'(R_1(\underline{x_1}, y_1') \wedge (x_1 = a))$.

  It can be verified that the latter formula correctly checks membership of $\mathsf{CQA}(q_1)$ by noticing that $q_1 = \langle R_1(\underline{a}, y), R_2(\underline{x}, y)\rangle$ is equivalent to $R_1(\underline{a}, y)$ if $R_1 = R_2$.

This concludes the introductory example.

Definition 3 defines the "equational form" for ordered rules which is assumed by our rewrite function. Definition 4 then introduces our rewrite function, which takes the form $\mathsf{Rew}_{q_1}(q_2 \wedge \varphi)$, where q_1 "remembers" the part of the query that has already been rewritten, so that atoms of q_2 can possibly be identified with atoms of q_1 (as in the above example). Note that the use of the separator $\wedge$ instead of a comma (,) is just for readability. Theorem 1 then states that this rewrite function computes a consistent FO rewriting for every key-rooted ordered rule.

Definition 3. Let $q = \langle R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\rangle$ be an ordered rule. Let $q' = \langle R_1(\underline{\vec{u}_1}, \vec{w}_1) \ldots, R_m(\underline{\vec{u}_m}, \vec{w}_m)\rangle$be the ordered rule obtained from q by putting a new fresh variable at each position. Thus, q' is constant-free and contains no two occurrences of the same variable. Let V be the set of variables that occur in q'. Let μ be the (unique) substitution over V such that $\mu(q') = q$. Let φ be a conjunction of equations such that:

1. whenever $v \in V$ and $\mu(v) = c$, where c is a constant, then φ contains $v = c$; and
2. whenever $v_1, v_2 \in V$ and $\mu(v_1) = \mu(v_2) = z$, where z is a variable, then φ contains $v_1 = v_2$.

Then, the formula q_{ef} defined as (the existential closure of)

$q_{ef} = R_1(\underline{\vec{u}_1}, \vec{w}_1) \wedge \cdots \wedge R_m(\underline{\vec{u}_m}, \vec{w}_m) \wedge \varphi'$,

where φ' is equal or equivalent to φ, is called an *equational form* for q.

A conjunction φ of equalities is *satisfiable* if there exists a valuation θ over the variables in φ such that for every equality $r = s$ in φ, $\theta(r) = \theta(s)$.

We write $q_1 \equiv q_2$, where q_1 and q_2 are Boolean queries, if for every database I, $I \models q_1$ if and only if $I \models q_2$.

It is easy to see that if $q_{ef} = \bigwedge_{j=1}^{m} R_j(\underline{\vec{u}_j}, \vec{w}_j) \wedge \varphi'$ is an equational form for q, then $q \equiv q_{ef}$. Moreover, this equivalence remains valid under valuations in the sense explained in the next paragraph. Consequently, the notions of reifiable atom and key-rootedness directly extend to equational forms.

Consider the ith symbol position in q and q_{ef} (counting from the left and ignoring φ; thus, the first position in q_{ef} is occupied by the leftmost variable in $\vec{u}_1$, and the last position is occupied by the rightmost variable in $\vec{w}_m$). Assume that q_{ef} contains variable u at position i. Two cases can occur:

1. q contains a variable x at position i. Let ω and θ be the valuations $\omega = \{x \mapsto a\}$ and $\theta = \{u \mapsto a\}$, where a is any constant. Then, $\omega(q) \equiv \theta(q_{ef})$.
2. q contains some constant b at position i. In this case, φ' implies $u = b$. For $\theta_b = \{u \mapsto b\}$, we have $\theta_b(q_{ef}) \equiv q$. Moreover, for every constant $c \neq b$, if $\theta_c = \{u \mapsto c\}$, then $\theta_c(\varphi')$ is unsatisfiable.

Example 4. Let c be a constant. Let

$q = \langle R(\underline{x}, y, y), S(\underline{y}, z, c)\rangle$,

$$q_{ef} = R(\underline{u}, v, w) \wedge S(\underline{r}, s, t) \wedge \begin{pmatrix} & (v = w)\\ \wedge & (w = r)\\ \wedge & (t = c)\end{pmatrix}.$$

Then, q_{ef} is an equational form for q. The substitution $\mu = \{(u,x),(v,y),(w,y),(r,y),(s,z),(t,c)\}$ maps the atoms of q_{ef} to the corresponding atoms of q.

Consider symbol position 3. The third symbol position in q is occupied by y. In q_{ef}, the variable w occurs at position 3. Let $\omega = \{y \mapsto a\}$ and $\theta = \{w \mapsto a\}$. The Boolean queries $\omega(q)$ and $\theta(q_{ef})$ are obviously equivalent:

$$\omega(q) = \langle R(\underline{x}, a, a), S(\underline{a}, z, c)\rangle,$$

$$\theta(q_{ef}) = R(\underline{u}, v, a) \wedge S(\underline{r}, s, t) \wedge \begin{pmatrix} & (v = a) \\ \wedge & (a = r) \\ \wedge & (t = c) \end{pmatrix}.$$

Definition 4. Let q_1, q_2 be (possibly empty) conjunctions of constant-free atoms. Let φ be a conjunction of equations.

$\mathsf{Rew}_{q_1}(q_2 \wedge \varphi)$ is inductively defined as follows:

Basis: $q_2 = \{\}$.

$$\mathsf{Rew}_{q_1}(\varphi) = \varphi$$

Step: $q_2 = R(\vec{x}, \vec{y}) \wedge q_3$.

$$\mathsf{Rew}_{q_1}(R(\vec{x}, \vec{y}) \wedge q_3 \wedge \varphi)$$
$$= \left(\bigvee_{R(\vec{v},\vec{w}) \in q_1} \exists\vec{x}\exists\vec{y} \begin{pmatrix} & (\vec{x} = \vec{v}) \\ \wedge & (\vec{y} = \vec{w}) \\ \wedge & \mathsf{Rew}_{q_1}(q_3 \wedge \varphi) \end{pmatrix} \right)$$
$$\vee$$
$$\exists\vec{x}\exists\vec{y}(R(\vec{x}, \vec{y}) \wedge (\textstyle\bigwedge_{R(\vec{v},\vec{w}) \in q_1} (\vec{x} \neq \vec{v})) \wedge$$
$$\forall\vec{y}(R(\vec{x}, \vec{y}) \rightarrow \mathsf{Rew}_{q_1 \cup \{R(\underline{\vec{x}},\vec{y})\}}(q_3 \wedge \varphi))).$$

It is understood that $\vec{x} = \vec{v}$ is a shorthand for $x_1 = v_1 \wedge \cdots \wedge x_k = v_k$, where $\vec{x} = \langle x_1, \ldots, x_k\rangle$ and $\vec{v} = \langle v_1, \ldots, v_k\rangle$. Likewise for $\vec{y} = \vec{w}$. The disequality $\vec{x} \neq \vec{v}$ is shorthand for $\neg(\vec{x} = \vec{v})$. Furthermore, $\exists\vec{x}$ is a shorthand for $\exists x_1 \cdots \exists x_k$. The empty disjunction is false and the empty conjunction is true.

For example, the complete rewriting of the rule $\langle R(\underline{a}, y), R(\underline{x}, y)\rangle$ now goes as follows. First, we write this rule in equational form, giving $R(\underline{x_1}, y_1) \wedge R(\underline{x_2}, y_2) \wedge (x_1 = a) \wedge (y_1 = y_2)$. Next,

$$\mathsf{Rew}_{\{\}}(R(\underline{x_1}, y_1) \wedge R(\underline{x_2}, y_2) \wedge (x_1 = a) \wedge (y_1 = y_2))$$
$$= \exists x_1 \exists y_1 (R(\underline{x_1}, y_1) \wedge \forall y_1 (R(\underline{x_1}, y_1)$$
$$\rightarrow \mathsf{Rew}_{\{R(\underline{x_1},y_1)\}}(R(\underline{x_2}, y_2) \wedge (x_1 = a) \wedge (y_1 = y_2)))),$$

where we have omitted empty disjunctions and

$$\mathsf{Rew}_{\{R(\underline{x_1},y_1)\}}(R(\underline{x_2}, y_2) \wedge (x_1 = a) \wedge (y_1 = y_2))$$
$$= \exists x_2 \exists y_2 \begin{pmatrix} & (x_2 = x_1) \\ \wedge & (y_2 = y_1) \\ \wedge & (x_1 = a) \\ \wedge & (y_1 = y_2) \end{pmatrix}$$
$$\vee$$
$$\exists x_2 \exists y_2 (R(\underline{x_2}, y_2) \wedge (x_2 \neq x_1) \wedge$$
$$\forall y_2 (R(\underline{x_2}, y_2) \rightarrow (x_1 = a) \wedge (y_1 = y_2))).$$

Theorem 1. *Let $q \wedge \varphi$ be an equational form for some key-rooted ordered rule. For every database I, $I \models_{\mathsf{sure}} q \wedge \varphi$ if and only if $I \models \mathsf{Rew}_{\{\}}(q \wedge \varphi)$.*

To obtain consistent FO rewritings of shorter length, in Definition 4, the first disjunct (between big brackets) can be equivalently rewritten as

$$\exists\vec{x}\exists\vec{y}\left(\mathsf{Rew}_{q_1}(q_3 \wedge \varphi) \wedge \left(\bigvee_{R(\vec{v},\vec{w}) \in q_1} (\vec{x} = \vec{v}) \wedge (\vec{y} = \vec{w})\right)\right).$$

Despite this shortening, since $\mathsf{Rew}_{q_1}(R(\vec{x}, \vec{y}) \wedge q_3 \wedge \varphi)$ calls both $\mathsf{Rew}_{q_1}(q_3 \wedge \varphi)$ and $\mathsf{Rew}_{q_1 \cup \{R(\underline{\vec{x}},\vec{y})\}}(q_3 \wedge \varphi)$, the length of $\mathsf{Rew}_{\{\}}(q \wedge \varphi)$ is $O(2^m)$ where $m = |q|$. However, if no relation name occurs more than once (i.e. for rules without self-join), the disjunction $\bigvee_{R(\vec{v},\vec{w}) \in q_1}(\cdots)$ is empty, resulting in a rewriting of length linear in $|q|$. In the literature, we found no algorithms for the consistent FO rewriting of rules with self-joins.

6. Join trees

Now that we are able to compute a consistent FO rewriting for every key-rooted rule, we can shift our attention to characterizing syntactic classes of key-rooted rules. This is essential, because our definition of key-rooted rules is semantic and provides no syntactic test to verify whether a rule is key-rooted. We will not use FM join graphs employed by others for characterizing classes of rules with a consistent FO rewriting. Instead, we use the join trees defined by Beeri et al. [5].

6.1. BFMY join trees

We recall the notion of join tree introduced by Beeri, Fagin, Maier, and Yannakakis [5]; the authors' initials will be used to make a distinction with the FM join trees introduced by Fuxman and Miller [3]. Fig. 4 shows two BFMY join trees. Compare with the FM join graphs of the same queries in Fig. 2.

Definition 5. A *BFMY join tree* for a rule q is an undirected (connected) tree whose vertices are the atoms of q such that:

Connectedness Condition: whenever the same variable x occurs in two distinct atoms $R_i(\vec{x_i}, \vec{y_i})$ and $R_j(\vec{x_j}, \vec{y_j})$, then x occurs in each atom on the unique path linking $R_i(\vec{x_i}, \vec{y_i})$ and $R_j(\vec{x_j}, \vec{y_j})$.

A rule is called *acyclic* if it has a BFMY join tree.

It is customary to label the edges of a join tree as follows: if e is an edge between $R_i(\vec{x_i}, \vec{y_i})$ and $R_j(\vec{x_j}, \vec{y_j})$,

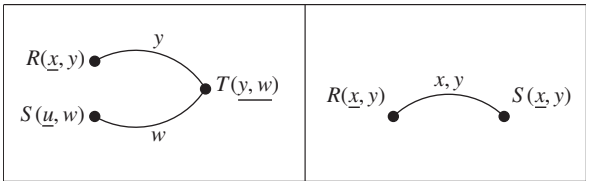

Fig. 4. BFMY join trees of $\{R(\underline{x}, y), S(\underline{u}, w), T(\underline{y}, w)\}$ and $\{R(\underline{x}, y), S(\underline{x}, y)\}$.

then e is labeled by the set of variables that occur in both $R_i(\vec{x}_i, \vec{y}_i)$ and $R_j(\vec{x}_j, \vec{y}_j)$. An edge label may be the empty set.

Unlike FM join graphs, BFMY join trees are undirected graphs. Nevertheless, if τ is a BFMY join tree, then a *directed rooted BFMY join tree* is obtained from τ by singling out one vertex of τ as the root.

The term *Connectedness Condition* appears in [16] and refers to the following property: if τ is a BFMY join tree and x a variable, then the set of vertices in which x occurs induces a (connected) subtree of τ.

The rule $\{P(\underline{y}, u, x), S(\underline{v}, \underline{u}), N(\underline{x}, \underline{z}, w), T(\underline{v}, \underline{w})\}$, whose FM join tree is shown in Fig. 3, is cyclic. The graph in Fig. 5 explains why there exists no BFMY join tree for that rule: the graph is a cycle and removing any one edge results in a tree that violates the *Connectedness Condition*.

Note that each rule q has a unique FM join graph, but can have zero, one or more BFMY join trees.

6.2. Number join trees

When a rule q is ordered, then its atoms can be indicated by their position in q. Roughly, we use the term "number join tree" for a BFMY join tree in which each atom is replaced by its position in q. Additionally, we will require that every path in a number join tree that starts from vertex 1 is increasing. Number join trees will be handy in the technical treatment because, as explained in Section 6.3, they remain unchanged under variable assignments.

Definition 6. Let q be an ordered rule of length m. A *number join tree* for q is an undirected tree τ whose vertices are the integers $1, \ldots, m$ such that:

1. *Connectedness Condition:* whenever the same variable x occurs in two atoms $q[i]$ and $q[j]$, then x occurs in $q[k]$ for every k on the unique path linking i and j $(1 \leqslant i, j \leqslant m)$.
2. *Increasingness Condition:* if $1 = i_0, i_1, i_2, \ldots, i_k = j$ is a path in τ from 1 to j, then $i_0 < i_1 < i_2 < \cdots < i_k$. Thus, vertices are strictly increasing along each path that starts from 1.

Optionally, edge labels can be added: if e is an edge linking i and j, then the label of e is the set of variables that occur in both $q[i]$ and $q[j]$.

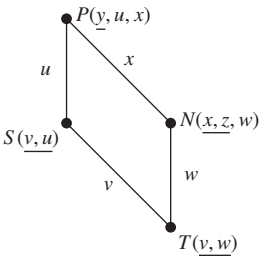

Fig. 5. The rule $\{P(\underline{y}, u, x), S(\underline{v}, \underline{u}), N(\underline{x}, \underline{z}, w), T(\underline{v}, \underline{w})\}$ is cyclic.

If the vertex 1 is chosen as the root, we obtain a directed rooted number join tree with root 1, whose vertices are numbered in increasing depth.

It is straightforward to transform a BFMY join tree into a number join tree: single out a vertex as the root and number it 1, then number all other vertices in increasing depth.

Lemma 3. *Let q be an acyclic rule and $m = |q|$. Let $\tau = (q, E)$, where E is the edge set, be a BFMY join tree for q. Let $R(\vec{x}, \vec{y})$ be any atom of q, and let $\tau^{R(\vec{x},\vec{y})}$ denote the directed rooted BFMY join tree obtained from τ by selecting $R(\vec{x}, \vec{y})$ as its root. Let $f : q \to \{1, \ldots, m\}$ be a bijection that numbers the atoms of q in increasing depth of $\tau^{R(\vec{x},\vec{y})}$. That is,*

1. *$f(R(\vec{x}, \vec{y})) = 1$; and*
2. *for all atoms $A_1, A_2 \in q$, if A_1 is the parent of A_2 in $\tau^{R(\vec{x},\vec{y})}$, then $f(A_1) < f(A_2)$.*

Then, the graph $(\{1, \ldots, m\}, f(E))$ is a number join tree for the ordered rule $\langle f^{-1}(1), \ldots, f^{-1}(m)\rangle$.

Proof. Let $q_o = \langle f^{-1}(1), \ldots, f^{-1}(m)\rangle$. Obviously, $(\{1, \ldots, m\}, f(E))$ is a tree.

Assume that the same variable x occurs in $q_o[i] = f^{-1}(i)$ and $q_o[j] = f^{-1}(j)$, and that k is on the path linking i and j. Then in τ, $f^{-1}(k)$ is on the path linking $f^{-1}(i)$ and $f^{-1}(j)$. By the *Connectedness Condition* for τ, x occurs in $f^{-1}(k) = q_o[k]$.

Assume that $1 = i_0, i_1, \ldots, i_k = j$ is a path from 1 to j in $(\{1, \ldots, m\}, f(E))$. Then, $f^{-1}(1) = f^{-1}(i_0), f^{-1}(i_1), \ldots, f^{-1}(i_k) = f^{-1}(j)$ is a path from $R(\vec{x}, \vec{y}) = f^{-1}(1)$ to $f^{-1}(j)$ in $\tau^{R(\vec{x},\vec{y})}$. By condition (2) in the statement of the lemma, $1 = i_0 < i_1 < \cdots < i_k = j$. □

Example 5. See Fig. 6. Let

$$q = \{R(\underline{x}), S(\underline{x}, u), T(\underline{x}, \underline{u}, y, z), R(\underline{y}), R(\underline{z})\}.$$

The tree with edge set

$$E = \{\{R(\underline{x}), S(\underline{x}, u)\}, \{S(\underline{x}, u), T(\underline{x}, \underline{u}, y, z)\}, \{T(\underline{x}, \underline{u}, y, z), R(\underline{y})\}, \{T(\underline{x}, \underline{u}, y, z), R(\underline{z})\}\}$$

is a BFMY join tree for q. If $R(\underline{x})$ is selected as the root, then the vertices of q are already listed in increasing depth. The vertex numbering f is defined by $f(R(\underline{x})) = 1$, $f(S(\underline{x}, u)) = 2$,

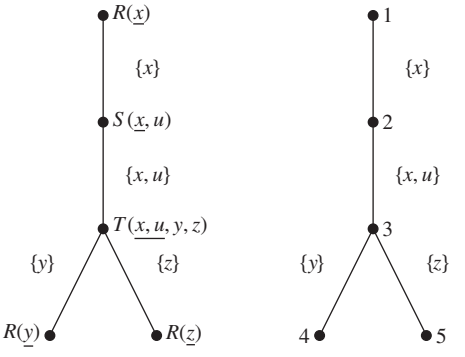

Fig. 6. BFMY join tree (left) and number join tree (right) for the ordered rule $\langle R(\underline{x}), S(\underline{x}, u), T(\underline{x}, \underline{u}, y, z), R(\underline{y}), R(\underline{z})\rangle$.

$f(T(\underline{x},\underline{u},y,z)) = 3$, $f(R(\underline{y})) = 4$, $f(R(\underline{z})) = 5$. We have

$f(E) = \{\{1,2\},\{2,3\},\{3,4\},\{3,5\}\}$,

which defines a number join tree for the ordered rule

$q_o = \langle R(\underline{x}), S(\underline{x},u), T(\underline{x},\underline{u},y,z), R(\underline{y}), R(\underline{z})\rangle$.

Conversely, let τ be a number join tree for an ordered rule q in which no atom occurs more than once; it is straightforward to transform τ into a BFMY join tree for q.

6.3. Operations on number join trees

Lemma 4 expresses that for any valuation θ, every number join tree for an ordered rule q is also a number join tree for $\theta(q)$. Notice incidentally that edge labels are optional and need to be recomputed when moving from q to $\theta(q)$. This straightforward property of number join trees is not shared by BFMY join trees, as illustrated by Example 6. Because of this, number join trees are somewhat handier than BFMY join trees in certain technical developments.

Lemma 4. *Let q be an ordered rule. Let V be the set of variables that occur in q, and $X \subseteq V$. Let θ be a valuation over X. Every number join tree for q is a number join tree for $\theta(q)$.*

Proof. Straightforward. □

Example 6. This is a continuation of Example 5. For the valuation $\theta = \{x \mapsto a, z \mapsto a\}$, we obtain

$\theta(q) = \{R(\underline{a}), S(\underline{a},u), T(\underline{a},\underline{u},y,a), R(\underline{y})\}$

and

$\theta(E) = \{\{R(\underline{a}), S(\underline{a},u)\}, \{S(\underline{a},u), T(\underline{a},\underline{u},y,a)\},$
$\{T(\underline{a},\underline{u},y,a), R(\underline{y})\}, \{T(\underline{a},\underline{u},y,a), R(\underline{a})\}\}$.

The edges of $\theta(E)$ are shown in Fig. 7 (left). Since θ maps both $R(\underline{x})$ and $R(\underline{z})$ to $R(\underline{a})$, $\theta(E)$ contains a cycle and hence is not a tree.

On the other hand, the tree with vertices $\{1,2,3,4,5\}$ and edges

$\{\{1,2\},\{2,3\},\{3,4\},\{3,5\}\}$

is a number join tree for the ordered rule

$\theta(q_o) = \langle R(\underline{a}), S(\underline{a},u), T(\underline{a},\underline{u},y,a), R(\underline{y}), R(\underline{a})\rangle$

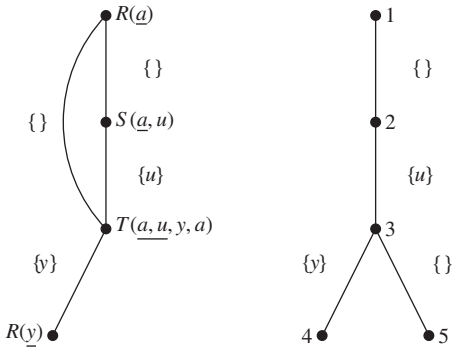

Fig. 7. Results of applying the valuation $\{x \mapsto a, z \mapsto a\}$ on the join trees of Fig. 6.

in which $R(\underline{a})$ occurs twice; it is shown in Fig. 7 (right). Notice that edge labels have been recomputed.

The following definition introduces an operator for constructing rules corresponding to subtrees in a directed rooted number join tree.

Definition 7. Let q be an ordered rule of length m that has a number join tree τ. Let τ^1 denote the directed rooted number join tree obtained from τ by selecting 1 as its root. Let $i \in \{1,2,\ldots,m\}$ and let d be a nonnegative integer. Let $i = i_0 < i_1 < i_2 < \cdots < i_k$ be the ascending sequence that contains i and all descendants of i in τ^1 that are at (graph) distance $\leqslant d$ from i. Then, $subd_q^\tau(i,d)$ is defined as the following ordered rule:

$subd_q^\tau(i,d) := \langle q[i_0], q[i_1], \ldots, q[i_k]\rangle$.

Finally, we define

$sub_q^\tau(i) := subd_q^\tau(i,\infty)$,

where ∞ denotes a large integer greater than the depth of τ.

Example 7. Let q be an ordered rule of length 12 that has a number join tree with root 1 as shown in Fig. 8. The vertex 3 and its descendants in increasing value are $3<5<7<8<9<10<11<12$. Hence, $sub_q^\tau(3) = \langle q[3], q[5], q[7], q[8], q[9], q[10], q[11], q[12]\rangle$.

The vertex 3 and its descendants at a distance $\leqslant 2$ from 3 are $3<5<7<8<11<12$. Hence, $subd_q^\tau(3,2) = \langle q[3], q[5], q[7], q[8], q[11], q[12]\rangle$.

Lemma 5. *Let q be an ordered rule of length m that has a number join tree τ. For each $i \in \{1,\ldots,m\}$, $sub_q^\tau(i)$ has a number join tree.*

Proof. Let $\tau = (\{1,\ldots,m\}, E)$. Let $i = i_1 < i_2 < \cdots < i_k$ be the ascending sequence that contains i and all descendants of i in τ^1. Let τ' be the vertex-induced subgraph induced by $\{i_1, i_2, \ldots, i_k\}$. Let f be the renumbering defined by $f(i_1) = 1$, $f(i_2) = 2, \ldots, f(i_k) = k$. A number join tree for $sub_q^\tau(i)$ is obtained by renumbering the vertices of τ' according to f. □

6.4. Merging key-rooted ordered rules

We show that if two key-rooted ordered rules q_1 and q_2 have no variable in common, then any merge of q_1 and q_2 results in a key-rooted rule.

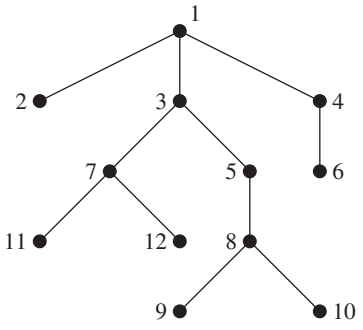

Fig. 8. Number join tree.

Definition 8. Let q_1 and q_2 be ordered rules. We say that an ordered rule

$$q_{merge} = \langle R_1(\vec{x}_1, \vec{y}_1), \ldots, R_m(\vec{x}_m, \vec{y}_m)\rangle$$

is a *merge* of the ordered rules q_1 and q_2 if there exist two sequences $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant m$ and $1 \leqslant j_1 < j_2 < \cdots j_l \leqslant m$ such that:

1. $\{i_1, \ldots, i_k\} \cap \{j_1, \ldots, j_l\} = \{\}$;
2. $\{i_1, \ldots, i_k\} \cup \{j_1, \ldots, j_l\} = \{1, 2, \ldots, m\}$;
3. $q_1 = \langle R_{i_1}(\vec{x}_{i_1}, \vec{y}_{i_1}), \ldots, R_{i_k}(\vec{x}_{i_k}, \vec{y}_{i_k})\rangle$; and
4. $q_2 = \langle R_{j_1}(\vec{x}_{j_1}, \vec{y}_{j_1}), \ldots, R_{j_l}(\vec{x}_{j_l}, \vec{y}_{j_l})\rangle$.

Thus, q_{merge} lists the atoms of q_1 in the order in which they occur in q_1, and q_{merge} lists the atoms of q_2 in the order in which they occur in q_2.

Lemma 6. *Let q_1 and q_2 be ordered rules such that no variable x occurs in both q_1 and q_2. Let q_{merge} be a merge of q_1 and q_2. If q_1 and q_2 are key-rooted, then q_{merge} is key-rooted.*

Proof. Proof by induction on $|q_{merge}|$, the length of q_{merge}. The result is trivial for the induction basis $|q_{merge}| = 0$, because the empty rule is key-rooted. For the induction step, assume $|q_{merge}| > 0$. Assume the numbering of Definition 8. Assume w.l.o.g. that $i_1 = 1$. Thus, the first atom of q_{merge} is the first atom of q_1. We use the following notations:

- $q'_1 = \langle R_{i_2}(\vec{x}_{i_2}, \vec{y}_{i_2}), \ldots, R_{i_k}(\vec{x}_{i_k}, \vec{y}_{i_k})\rangle$, the tail of q_1; and
- $q'_{merge} = \langle R_2(\vec{x}_2, \vec{y}_2), \ldots, R_m(\vec{x}_m, \vec{y}_m)\rangle$, the tail of q_{merge}.

Let I be a database such that $I \models_{\mathsf{sure}} q_{merge}$. Clearly, $I \models_{\mathsf{sure}} q_1$ and $I \models_{\mathsf{sure}} q_2$. Since q_1 is key-rooted, it follows:

1. there exists a valuation θ of $\vec{x}_1$ such that $I \models_{\mathsf{sure}} \theta(q_1)$; and
2. for each valuation μ of $\vec{x}_1\vec{y}_1$, $\mu(q'_1)$ is key-rooted.

Let J be a repair of I. Let X_1 and X_2 be the sets of variables that occur in q_1 and q_2, respectively. By the first item above, there exists a valuation ω_1 over X_1 such that $\omega_1(\vec{x}_1) = \theta(\vec{x}_1)$ and $J \models \omega_1(q_1)$. Since $I \models_{\mathsf{sure}} q_2$, there exists a valuation ω_2 over X_2 such that $J \models \omega_2(q_2)$. Let ω be the valuation over $X_1 \cup X_2$ such that $\omega(x) = \omega_1(x)$ if $x \in X_1$ and $\omega(x) = \omega_2(x)$ if $x \in X_2$. The valuation ω is well defined because $X_1 \cap X_2 = \{\}$. Clearly, $J \models \omega(q_{merge})$. Hence, $J \models \theta(q_{merge})$. Since J is an arbitrary repair of I, $I \models_{\mathsf{sure}} \theta(q_{merge})$.

Let μ be a valuation of $\vec{x}_1\vec{y}_1$. Clearly, $\mu(q'_{merge})$ is a merge of $\mu(q'_1)$ and $\mu(q_2)$. Since no variable x occurs in both $\vec{x}_1\vec{y}_1$ and q_2, $\mu(q_2) = q_2$. Since $\mu(q'_1)$ is key-rooted by the second item above, and q_2 is key-rooted by our initial assumption, $\mu(q'_{merge})$ is a merge of two key-rooted ordered rules. Since the length of $\mu(q'_{merge})$ is $|q_{merge}| - 1$, by the induction hypothesis, $\mu(q'_{merge})$ is key-rooted.

We have shown:

1. $I \models_{\mathsf{sure}} \theta(q_{merge})$ for some valuation θ of $\vec{x}_1$; and
2. for each valuation μ of $\vec{x}_1\vec{y}_1$, $\mu(q'_{merge})$ is key-rooted, where q'_{merge} is the tail of q_{merge}.

Consequently, q_{merge} is key-rooted. □

The following corollary is very useful in establishing key-rootedness of ordered, acyclic rules.

Corollary 1. *Let q be an ordered rule of length m that has a number join tree τ with root 1. Let $q[1] = R(\underline{\vec{x}}, \vec{y})$ and let θ be a valuation of $\vec{x}\vec{y}$. If $\theta(sub_q^\tau(i))$ is key-rooted for each child i of 1, then $\theta(q')$ is key-rooted, where q' is the tail of q.*

Proof. Assume w.l.o.g. that $2, 3, \ldots, l$ are the children of 1 in the number join tree τ with root 1. Clearly, the tail q' of q is a merge of $sub_q^\tau(2), \ldots, sub_q^\tau(l)$. Consequently, $\theta(q')$ is a merge of $\theta(sub_q^\tau(2)), \ldots, \theta(sub_q^\tau(l))$. By the *Connectedness Condition*, the ordered rules $\theta(sub_q^\tau(2)), \ldots, \theta(sub_q^\tau(l))$ have pairwise disjoint sets of variables. By repeated application of Lemma 6, $\theta(q')$ is key-rooted. □

7. New classes of rules with a consistent first-order rewriting

This section consists of three subsections, each of which contains a theorem introducing a new class of key-rooted rules. Sections 7.1 and 7.2 cover rules in which the same relation name can occur multiple times. Section 7.3 then elaborates on the class C_{tree} defined in Section 3. Significantly, the key-rooted rules in Sections 7.1 and 7.3 can have cyclic FM join graphs (but always have acyclic BFMY join trees).

7.1. No variables in the primary key of the root

Theorem 2 uses the construct of number join tree to characterize a class of key-rooted rules—and for which Theorem 1 thus provides a consistent FO rewriting. The class contains the rule shown in Fig. 9. The rule has four occurrences of the same relation name and the FM join graph (not shown) would contain a directed edge from any atom to any other atom (and hence would not be a tree).

Theorem 2. *An ordered rule*

$$q = \langle R_1(\vec{x}_1, \vec{y}_1), \ldots, R_m(\vec{x}_m, \vec{y}_m)\rangle$$

is key-rooted if it has a number join tree τ with root 1 such that:

1. $\mathsf{vars}(\vec{x}_1) = \{\}$. *Thus, the primary key of the first atom contains only constants.*
2. *For $i, j \in \{1, \ldots, m\}$, if i is the parent of j, then j is a leaf vertex of τ or $\mathsf{vars}(\vec{x}_j) \subseteq \mathsf{vars}(\vec{x}_i\vec{y}_i)$.*

Theorem 2 immediately leads to the following result for (non-ordered) rules.

Corollary 2. *A rule $q = \{R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\}$ has a consistent FO rewriting if it has a directed rooted BFMY join tree τ such that*:

1. *if $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ is the root of τ, then $\mathsf{vars}(\vec{x}_i) = \{\}$; and*
2. *if $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ is the parent of $R_j(\underline{\vec{x}_j}, \vec{y}_j)$, then $R_j(\underline{\vec{x}_j}, \vec{y}_j)$ is a leaf vertex of τ or $\mathsf{vars}(\vec{x}_j) \subseteq \mathsf{vars}(\vec{x}_i\vec{y}_i)$.*

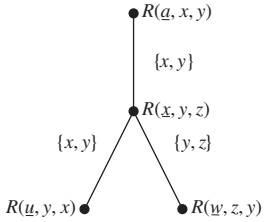

Fig. 9. BFMY join tree for the rule $\{R(\underline{a},x,y), R(\underline{x},y,z), R(\underline{u},y,x), R(\underline{w},z,y)\}$.

Proof. Clearly, by the construction in Lemma 3, there exists an ordered rule q_o such that q_o contains exactly the atoms of q and q_o has a number join tree that satisfies all conditions of Theorem 2. By Theorem 1, q_o has a consistent FO rewriting. □

The ordered rules captured by Theorem 2 reflect the definition of key-rooted rules (Definition 2). If q is an ordered rule in this class, then the primary key of q's prefix is variable-free, hence the prefix is reifiable. Next, the second condition in the theorem's statement guarantees that either q's tail is a singleton, or whenever θ is a valuation of q's prefix, then θ applied to q's tail is a shorter rule in the same class.

7.2. Single relation name

Theorem 3 characterizes another class of key-rooted rules in terms of number join trees. A rule of this class is shown in Fig. 10. Unlike the queries considered in [3], all atoms of rules in this class share the same relation name.

Theorem 3. *An ordered rule*

$$q = \langle R(\vec{x}_1,\vec{y}_1),\ldots,R(\vec{x}_m,\vec{y}_m)\rangle,$$

with a single relation name R, is key-rooted if it has a number join tree τ with root 1 such that:

1. *Every atom is constant-free and contains no two occurrences of the same variable.*
2. *For all internal vertices i and j, the ordered rules $subd_q^\tau(i,1)$ and $subd_q^\tau(j,1)$ are the same up to a renaming of variables.*
3. *All leaf vertices are at the same depth.*
4. *For $i,j \in \{1,\ldots,m\}$, if i is the parent of j, then $\mathsf{vars}(\vec{x}_i\vec{y}_i) \cap \mathsf{vars}(\vec{x}_j\vec{y}_j) = \mathsf{vars}(\vec{x}_j)$.*

Theorem 3 immediately leads to the following result for (non-ordered) rules.

Corollary 3. *A rule $q = \{R(\vec{x}_1,\vec{y}_1),\ldots,R(\vec{x}_m,\vec{y}_m)\}$, with a single relation name R, has a consistent FO rewriting if it has a directed rooted BFMY join tree τ such that*:

1. *Every atom is constant-free and contains no two occurrences of the same variable.*
2. *For all internal vertices $R(\underline{\vec{x}_i},\vec{y}_i)$ and $R(\underline{\vec{x}_j},\vec{y}_j)$, the subtree induced by $R(\underline{\vec{x}_i},\vec{y}_i)$ and all its children is the same, up to a renaming of variables, as the subtree induced by $R(\underline{\vec{x}_j},\vec{y}_j)$ and its children.*

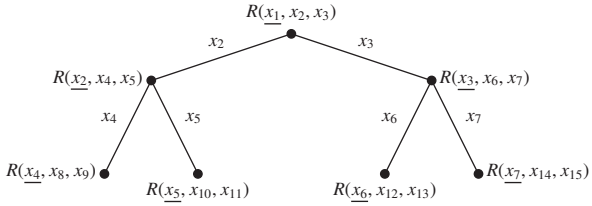

Fig. 10. BFMY join tree for a rule covered by Theorem 3.

3. *All leaf vertices are at the same depth.*
4. *If atom $R(\underline{\vec{x}_i},\vec{y}_i)$ is the parent of $R(\underline{\vec{x}_j},\vec{y}_j)$, then $\mathsf{vars}(\vec{x}_i\vec{y}_i) \cap \mathsf{vars}(\vec{x}_j\vec{y}_j) = \mathsf{vars}(\vec{x}_j)$.*

The following examples show that Theorem 3 is no longer valid if one of conditions (1)–(4) is dropped.

Example 8. The rule $q_1 = \langle R(\underline{x},y), R(\underline{y},c)\rangle$, where c is a constant, satisfies all but condition (1) in Theorem 3. In Section 8, it is shown that this rule has no consistent FO rewriting, and hence cannot be key-rooted.

The rule $q_4 = \langle R(\underline{x},y), R(\underline{y},x)\rangle$ satisfies all but condition (4). The atom $R(\underline{x},y)$ is not reifiable in q_4, hence q_4 is not key-rooted. Witness thereof is the database

$$I = \{R(\underline{a},b), R(\underline{b},a), R(\underline{b},c), R(\underline{c},b), R(\underline{c},d), R(\underline{d},c)\},$$

which has four repairs:

$$J_1 = \{R(\underline{a},b), R(\underline{b},a), R(\underline{c},b), R(\underline{d},c)\},$$
$$J_2 = \{R(\underline{a},b), R(\underline{b},c), R(\underline{c},b), R(\underline{d},c)\},$$
$$J_3 = \{R(\underline{a},b), R(\underline{b},a), R(\underline{c},d), R(\underline{d},c)\},$$
$$J_4 = \{R(\underline{a},b), R(\underline{b},c), R(\underline{c},d), R(\underline{d},c)\}.$$

Since each repair satisfies q_4, we have $I \models_{\mathsf{sure}} q_4$. The atom $R(\underline{x},y)$ is not reifiable in q_4, because for every valuation θ of x, there is at least one repair that falsifies $\theta(q_4)$.

Example 9. The ordered rule

$$q_2 = \langle R(\underline{x_1},x_2,x_3), R(\underline{x_2},x_4,x_5), R(\underline{x_5},x_6,x_7)\rangle$$

with a BFMY join tree as shown in Fig. 11, satisfies all but condition (2) in Theorem 3, since the two subrules

$$\langle R(\underline{x_1},x_2,x_3), R(\underline{x_2},x_4,x_5)\rangle, \quad \langle R(\underline{x_2},x_4,x_5), R(\underline{x_5},x_6,x_7)\rangle$$

are not equal up to a variable renaming. For the database $I = \{R(\underline{a},b,b),\ R(\underline{b},a,1),\ R(\underline{b},1,a)\}$, we have $I \models_{\mathsf{sure}} q_2$ but there is no valuation θ of x_1 such that $I \models_{\mathsf{sure}} \theta(q_2)$. Thus, $R(\underline{x_1},x_2,x_3)$ is not reifiable in q_2.

Example 10. The ordered rule

$$\begin{aligned} q_3 = \langle & R(\underline{x_1},x_2,x_3), R(\underline{x_2},x_4,x_5), R(\underline{x_3},x_6,x_7),\\ & R(\underline{x_4},x_8,x_9), R(\underline{x_5},x_{10},x_{11}), R(\underline{x_6},x_{12},x_{13}),\\ & R(\underline{x_7},x_{14},x_{15}), R(\underline{x_8},x_{16},x_{17}), R(\underline{x_9},x_{18},x_{19}),\\ & R(\underline{x_{14}},x_{20},x_{21}), R(\underline{x_{15}},x_{22},x_{23})\rangle \end{aligned}$$

of which a BFMY join tree is shown in Fig. 12, satisfies all but condition (3) in Theorem 3; for the database $I = \{R(\underline{a},c,3),\ R(\underline{b},3,c),\ R(\underline{c},1,2),\ R(\underline{c},2,1),\ R(\underline{3},2,2),\ R(\underline{2},1,1),\ R(\underline{1},0,0)\}$, we have $I \models_{\mathsf{sure}} q_3$ but there is no valuation θ of x_1 such that $I \models_{\mathsf{sure}} \theta(q_3)$. Thus, $R(\underline{x_1},x_2,x_3)$ is not reifiable in q_3.

7.3. No duplicate relation names

Theorem 4 and its Corollary 4 extend the class C_{tree} defined in Section 3. They deal with rules without self-joins (condition (1) in the theorem's statement). Condition (2b) states that the variables shared by a child and its parent are (possibly strictly) contained in the child's primary key variables. Thus, unlike C_{tree}, nonkey-to-key joins need not be full. Condition (2a) provides an alternative: the primary key of the child contains all variables of the parent's primary key.

Theorem 4. *An ordered rule*

$q = \langle R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\rangle$

is key-rooted if it has a number join tree τ with root 1 *such that*:

1. *If $i \neq j$, then $R_i \neq R_j$. Thus, no relation name occurs more than once in q.*
2. *If i is the parent of j $(1 \leqslant i, j \leqslant m)$, then at least one of the following two conditions is true*:
   (a) $\mathsf{vars}(\vec{x}_j) \supseteq \mathsf{vars}(\vec{x}_i)$; *or*
   (b) $\mathsf{vars}(\vec{x}_j) \supseteq \mathsf{vars}(\vec{x}_i\vec{y}_i) \cap \mathsf{vars}(\vec{x}_j\vec{y}_j)$. *That is,* $\mathsf{vars}(\vec{x}_j)$ *is a superset of the label on the edge between i and j.*

Theorem 4 immediately leads to the following result for (non-ordered) rules.

Corollary 4. *A rule $q = \{R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\}$ has a consistent FO rewriting if it has a directed rooted BFMY join tree τ such that*:

1. *no relation name occurs more than once in q; and*
2. *if $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ is the parent of $R_j(\underline{\vec{x}_j}, \vec{y}_j)$ $(1 \leqslant i, j \leqslant m)$, then at least one of the following conditions is true*:
   (a) $\mathsf{vars}(\vec{x}_j) \supseteq \mathsf{vars}(\vec{x}_i)$; *or*
   (b) $\mathsf{vars}(\vec{x}_j) \supseteq \mathsf{vars}(\vec{x}_i\vec{y}_i) \cap \mathsf{vars}(\vec{x}_j\vec{y}_j)$.

For example, the "intersection" rule $\{R(\underline{x}, y), S(\underline{x}, y)\}$ (right graph of Fig. 4) is covered by the corollary and hence has a consistent FO rewriting:

$\exists x \exists y (R(\underline{x}, y) \wedge S(\underline{x}, y) \wedge \forall z ((R(\underline{x}, z) \vee S(\underline{x}, z)) \rightarrow z = y))$.

The rule $\{R(\underline{x}, y), S(\underline{x}, y)\}$ is not in C_{tree} because its FM join graph is cyclic (right graph in Fig. 2). The acyclic rule

$\{R(\underline{x}, y), S(\underline{u}, w), T(\underline{y}, w)\}$

has a unique BFMY join tree (left graph of Fig. 4) but is not covered by Corollary 4, no matter which atom is selected as the root.

By proving that every rule in C_{tree} satisfies the conditions of Corollary 4, we obtain the following result.

Corollary 5. *Every rule in C_{tree} has a consistent FO rewriting.*

Proof. Let $q = \{R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_m(\underline{\vec{x}_m}, \vec{y}_m)\}$ be a rule in C_{tree} with FM join tree τ.

Assume that two distinct atoms $R_i(\underline{\vec{x}_i}, \vec{y}_i), R_j(\underline{\vec{x}_j}, \vec{y}_j) \in q$ have a variable u in common. Recall that our definition of C_{tree} implies that the FM join graph of q is connected. Then, two cases can occur:

1. One of $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ or $R_j(\underline{\vec{x}_j}, \vec{y}_j)$ is an ancestor of the other. Assume w.l.o.g. that $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ is an ancestor of $R_j(\underline{\vec{x}_j}, \vec{y}_j)$.Then $u \in \mathsf{vars}(\vec{x}_j) \backslash \mathsf{vars}(\vec{y}_j)$, or else the FM join graph would be cyclic, a contradiction. Let $R_l(\underline{\vec{x}_l}, \vec{y}_l)$ be the parent of $R_j(\underline{\vec{x}_j}, \vec{y}_j)$. Since all nonkey-to-key joins are full, $u \in \mathsf{vars}(\vec{y}_l)$. Then $l = i$, or else the FM join graph would be cyclic, a contradiction. It follows that $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ is the parent of $R_j(\underline{\vec{x}_j}, \vec{y}_j)$.
2. Neither of $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ or $R_j(\underline{\vec{x}_j}, \vec{y}_j)$ is an ancestor of the other. Then $u \in \mathsf{vars}(\vec{x}_i) \backslash \mathsf{vars}(\vec{y}_i)$, or else there would be a directed edge from $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ to $R_j(\underline{\vec{x}_j}, \vec{y}_j)$, a contradiction. Let $R_l(\underline{\vec{x}_l}, \vec{y}_l)$ be the parent of $R_i(\underline{\vec{x}_i}, \vec{y}_i)$. Since

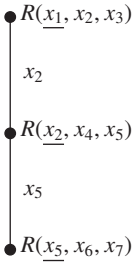

Fig. 11. BFMY join tree for a rule not covered by Theorem 3.

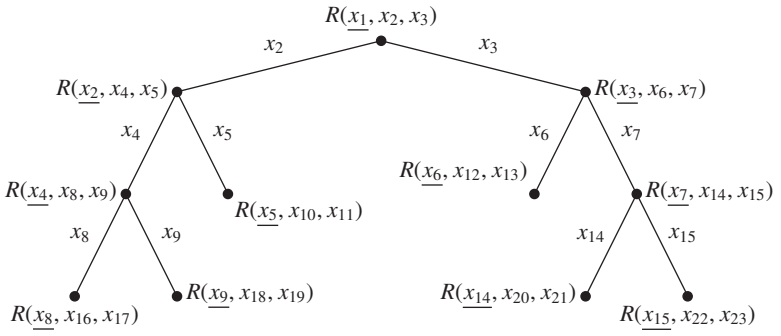

Fig. 12. BFMY join tree for a rule not covered by Theorem 3.

$u \in \mathsf{vars}(\vec{y}_l)$, there is a directed edge from $R_l(\underline{\vec{x}_l}, \vec{y}_l)$ to $R_j(\underline{\vec{x}_j}, \vec{y}_j)$. That is, $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ and $R_j(\underline{\vec{x}_j}, \vec{y}_j)$ are siblings and their common parent contains u.

Hence, if two distinct atoms share a variable u, then either one of the atoms is the parent of the other, or both atoms are siblings and their common parent atom contains u. It follows that the FM join tree of q is a directed rooted BFMY join tree satisfying the conditions of Corollary 4. □

8. $\exists x \exists y(R(\underline{x}, y) \wedge R(\underline{y}, c))$ has no consistent first-order rewriting

We found in the literature no rewriting algorithms that produce consistent FO rewritings for rules that contain self-joins. Theorems 2 and 3 seem to be the first positive results in this direction. We now argue that there is little hope to significantly extend these results.

Clearly, under the assumption $\mathbf{P} \neq \mathbf{NP}$, a rule q can have no consistent FO rewriting if $\mathsf{CQA}(q)$ is **coNP**-complete. We will now show that the simple rule $q = \{R(\underline{x}, y), R(\underline{y}, c)\}$, where c is a constant, has no consistent FO rewriting, even though $\mathsf{CQA}(q)$ is in **P**. This may come as a surprise, because the nonkey-to-key join is full, the rule has a BFMY join tree, and its FM join graph is a tree. Intuitively, if R encodes the edge set of a graph, the query asks whether there is a path of length 2 that ends in a distinguished vertex c. The rule is not covered by Corollary 2 because no primary key is ground; it is not covered by Corollary 3 because it contains a constant. So it turns out that the double occurrence of the same relation name in a rule q easily leads to the non-existence of a consistent FO rewriting (even if $\mathsf{CQA}(q)$ is in **P**).

The intuition behind (the proof of) Theorem 5 can be understood as follows. Let $q = \{R(\underline{x}, y), R(\underline{y}, c)\}$. Consider the class of databases encoding directed acyclic graphs with exactly two sinks c and ε. Let I be any database in this class. Notice that in every repair, every vertex can have at most one outgoing edge. We argue hereafter that $I \not\models_{\mathsf{sure}} q$ if and only if for every vertex x that is not a sink,

- $R(\underline{x}, c) \in I$, or
- I contains a directed path from x to ε.

For the if-direction, construct a repair J of I such that for every vertex x, if I contains a directed path from x to ε, then so does J; otherwise J contains $R(\underline{x}, c)$. This is illustrated by the database $\mathfrak{B}$ in Fig. 13 (right) and its repair of Fig. 14. Then J contains no path of length 2 ending in c. For the opposite direction (only-if), assume a vertex x, $x \neq c$ and $x \neq \varepsilon$, such that $R(\underline{x}, c) \notin I$ and there is no directed path from x to ε. Thus, each maximal path that starts from x, ends in c. Then it is easy to see that every repair J of I contains a directed path with length $\geqslant 2$ from x to c, hence $J \models q$. This is illustrated by the database $\mathfrak{A}$ in Fig. 13 (left), which contains no directed edge from δ to c, and no directed path from δ to ε.

The quintessence now is that reachability from x to ε is not first-order expressible. In particular, we show that for any given FO sentence ψ, the databases $\mathfrak{A}$ and $\mathfrak{B}$ can be chosen sufficiently large such that they cannot be distinguished by ψ. On the other hand, the existence of particular paths can be verified in polynomial time.

Theorem 5. *Let* $q = \{R(\underline{x}, y), R(\underline{y}, c)\}$.

1. $\mathsf{CQA}(q)$ *is in* **P**.
2. *There exists no Boolean FO query* ψ *such that for every database* I, $I \models_{\mathsf{sure}} q$ *if and only if* $I \models \psi$. *Thus,* q *has no consistent FO rewriting.*

9. Rules with free variables

The rewrite functions of Definition 4 and Theorem 1 are given for Boolean queries. We now show how to handle conjunctive queries with free variables.

Let $q(x_1, \ldots, x_n)$ denote an ordered rule with n free variables $x_1, \ldots, x_n$. Let $\vec{x} = \langle x_1, \ldots, x_n \rangle$. For every sequence $\vec{a} = \langle a_1, \ldots, a_n \rangle$ of constants:

- $q_{\vec{x} \mapsto \vec{a}}$ denotes the ordered rule obtained from q by replacing each free occurrence of x_i with a_i, for all $1 \leqslant i \leqslant n$; and
- $q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{a}}$ denotes an equational form for $q_{\vec{x} \mapsto \vec{a}}$ as defined in Definition 3.

Since every constant symbol is interpreted by itself, we have the following equivalences for any database I:

$$I \models_{\mathsf{sure}} q(\vec{a}) \iff I \models_{\mathsf{sure}} q_{\vec{x} \mapsto \vec{a}} \iff I \models_{\mathsf{sure}} q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{a}}.$$

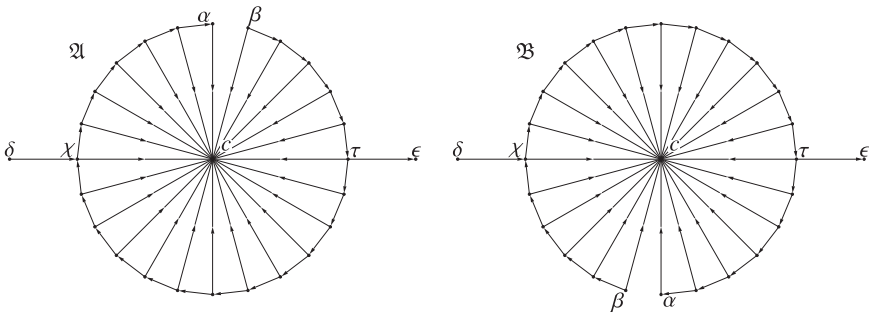

Fig. 13. $\exists x \exists y(R(\underline{x}, y) \wedge R(\underline{y}, c))$ is consistently true in $\mathfrak{A}$, but not in $\mathfrak{B}$.

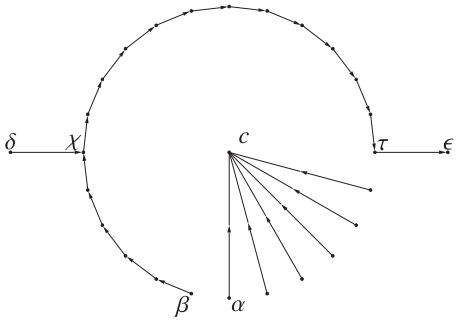

Fig. 14. Repair of $\mathfrak{B}$ falsifying $\exists x \exists y (R(\underline{x}, y) \wedge R(\underline{y}, c))$.

Example 11. Let

$$q(x_1, x_2) = \exists y (R(\underline{x_1}, y) \wedge S(\underline{y}, x_2)),$$

with two free variables x_1 and x_2. Let a_1 and a_2 be two (not necessarily distinct) constants. Then,

$$q_{\vec{x} \mapsto \vec{a}} = \langle R(\underline{a_1}, y), S(\underline{y}, a_2) \rangle,$$

$$q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{a}} = R(\underline{u_1}, w_1) \wedge S(\underline{w_2}, u_2) \wedge w_1 = w_2 \wedge u_1 = a_1 \wedge u_2 = a_2.$$

By Theorem 1, for each $\vec{a}$ such that the rule $q_{\vec{x} \mapsto \vec{a}}$ is key-rooted, for every database I,

$$I \models_{\mathsf{sure}} q(\vec{a}) \iff I \models \mathsf{Rew}_{\{\}}(q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{a}}). \tag{3}$$

Let $\vec{c} = \langle c_1, \ldots, c_n \rangle$ be a sequence of n new distinct constants. Clearly, for every $\vec{a}$, it can be assumed without loss of generality that $q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{a}}$ is obtained from $q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{c}}$ by replacing each occurrence of c_i with a_i, for all $1 \leqslant i \leqslant n$. Furthermore, it is easy to verify that the rewrite function of Definition 4 is such that for every $\vec{a}$, $\mathsf{Rew}_{\{\}}(q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{a}})$ can be obtained from $\mathsf{Rew}_{\{\}}(q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{c}})$ by replacing each c_i with a_i, for all $1 \leqslant i \leqslant n$.

The variables $x_1, \ldots, x_n$ do not occur in $\mathsf{Rew}_{\{\}}(q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{c}})$. Let Q be the query obtained from $\mathsf{Rew}_{\{\}}(q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{c}})$ by replacing each occurrence of c_i with x_i, for all $1 \leqslant i \leqslant n$. Then, Q is a query with free variables $x_1, \ldots, x_n$, denoted $Q(x_1, \ldots, x_n)$, such that for all $\vec{a}$, for every database I,

$$I \models Q(\vec{a}) \iff I \models \mathsf{Rew}_{\{\}}(q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{a}}). \tag{4}$$

From (3) and (4), it follows that for each $\vec{a}$ such that $q_{\vec{x} \mapsto \vec{a}}$ is key-rooted, for every database I,

$$I \models_{\mathsf{sure}} q(\vec{a}) \iff I \models Q(\vec{a}).$$

Thus, $Q(x_1, \ldots, x_n)$ is a consistent FO rewriting of $q(x_1, \ldots, x_n)$.

Example 12. Continuation of Example 11. We have

$$q(x_1, x_2) = \exists y (R(\underline{x_1}, y) \wedge S(\underline{y}, x_2)).$$

Let c_1 and c_2 be two distinct constants. Then,

$$q_{\vec{x} \mapsto \vec{c}} = \langle R(\underline{c_1}, y), S(\underline{y}, c_2) \rangle$$

and

$$q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{c}} = R(\underline{u_1}, w_1) \wedge S(\underline{w_2}, u_2) \wedge \begin{pmatrix} & (w_1 = w_2) \\ \wedge & (u_1 = c_1) \\ \wedge & (u_2 = c_2) \end{pmatrix}.$$

For $R \neq S$, we get the following rewriting:

$$\begin{aligned} &\mathsf{Rew}_{\{\}}(q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{c}}) \\ &\quad = \exists u_1 \exists w_1 (R(\underline{u_1}, w_1) \wedge \\ &\qquad \forall w_1 (R(\underline{u_1}, w_1) \rightarrow \exists w_2 \exists u_2 (S(\underline{w_2}, u_2) \wedge \\ &\qquad \forall u_2 (S(\underline{w_2}, u_2) \rightarrow \varphi)))), \end{aligned}$$

where

$$\varphi = (w_1 = w_2) \wedge (u_1 = c_1) \wedge (u_2 = c_2).$$

To help readability, we can simplify $\mathsf{Rew}_{\{\}}(q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{c}})$ as follows:

$$\begin{aligned} &\mathsf{Rew}_{\{\}}(q^{\mathsf{ef}}_{\vec{x} \mapsto \vec{c}}) \equiv \\ &\quad \exists w_1 (R(\underline{c_1}, w_1) \wedge \\ &\quad \forall w_1 (R(\underline{c_1}, w_1) \rightarrow \quad \exists u_2 (S(\underline{w_1}, u_2) \wedge \\ &\qquad\qquad\qquad\qquad \forall u_2 (S(\underline{w_1}, u_2) \rightarrow u_2 = c_2)))) \end{aligned}.$$

Finally, Q is obtained by replacing c_1 and c_2 with x_1 and x_2, respectively:

$$\begin{aligned} Q(x_1, x_2) = \exists w_1 (R(\underline{x_1}, w_1) \wedge \\ \forall w_1 (R(\underline{x_1}, w_1) \rightarrow \exists u_2 (S(\underline{w_1}, u_2) \wedge \\ \forall u_2 (S(\underline{w_1}, u_2) \rightarrow u_2 = x_2)))). \end{aligned}$$

Since it can be easily verified, using Theorem 2, that $q_{\vec{x} \mapsto \vec{a}}$ is key-rooted for all $\vec{a}$, it follows that for all a_1, a_2, $I \models_{\mathsf{sure}} q(a_1, a_2) \iff I \models Q(a_1, a_2)$.

10. Concluding remarks

In the past literature, consistent FO rewriting under primary keys has mainly be specified by procedural program code. Our consistent FO rewrite function of Definition 4 is considerably more succinct than existing rewrite algorithms, and yet turns out to be widely applicable. We proved that this rewrite function yields a consistent FO rewriting for any key-rooted rule. This result allowed us to shift our attention from the syntactical intricacies of FO rewriting toward characterizing classes of key-rooted rules. This characterization was successful using BFMY join trees (instead of FM join trees used so far in the literature). Finally, we showed that the rule $\{R(\underline{x}, y), R(\underline{y}, c)\}$ has no consistent FO rewriting.

In this article, the target language of the rewrite function is first-order. The motivation for this is that first-order queries execute in polynomial time data complexity and can be easily encoded in SQL. Nevertheless, the proof of Theorem 5 suggests adding some form of recursion to the target language.

The model-theoretic notion of key-rooted rule was engineered so as to capture the typical $\exists\forall$ quantifier alternation in consistent FO rewritings: the formula $\exists \vec{x} \exists \vec{z} \forall \vec{y} (R(\vec{x}, \vec{z}) \wedge (R(\vec{x}, \vec{y}) \rightarrow \psi))$ expresses that ψ must hold no matter how we repair R-atoms that agree on the

primary key $\vec{x}$. Three intriguing questions about key-rootedness are open for further research:

- Does there exist a rule q such that q has a consistent FO rewriting but q is not key-rooted, no matter how its atoms are ordered?
- Is it decidable whether a given rule q is key-rooted? Partial solutions to this question appear in [17].
- We used BFMY join trees to syntactically characterize key-rooted rules. Can generalizations of BFMY join trees [18] be used to characterize wider classes of key-rooted rules?

Acknowledgments

The comments and suggestions of the anonymous reviewers were highly appreciated.

Appendix A. Proof of Proposition 1

Proof of Proposition 1. Reduction form MONOTONE 3SAT. Let $\phi = \bigwedge_{i=1}^{n} \phi_i$, where each ϕ_i is either a disjunction of three positive literals or a disjunction of three negative literals. Let a be a constant that does not occur in ϕ. We can assume a fixed linear order $\preceq$ on the propositional variables occurring in ϕ. The formula ϕ induces a database, denoted $db(\phi)$, as follows.

- For each propositional variable p occurring in ϕ, $db(\phi)$ contains $S(\underline{p,p})$ and $T(\underline{p,p})$.
- For each $i \in \{1,2,\ldots,n\}$,
  - if ϕ_i contains only positive literals, then $db(\phi)$ contains $P(\underline{i,p,a})$ for each propositional variable p occurring in ϕ_i; and
  - if ϕ_i contains only negative literals, then $db(\phi)$ contains $N(\underline{a,i,p})$ for each propositional variable p occurring in ϕ_i.

We show that ϕ is satisfiable if and only if $db(\phi) \not\models_{\mathsf{sure}} q$.

For the *only-if* part, assume ϕ satisfiable. We can assume a truth assignment B satisfying ϕ. Construct a database J containing all S-atoms and T-atoms of $db(\phi)$ and such that for every $i \in \{1,2,\ldots,n\}$,

- if ϕ_i contains only positive literals, then J contains $P(\underline{i,p,a})$ where p is the smallest (under $\preceq$) variable of ϕ_i satisfying $B(p) = \mathbf{true}$; and
- if ϕ_i contains only negative literals, then J contains $N(\underline{a,i,p})$ where p is the smallest variable of ϕ_i satisfying $B(p) = \mathbf{false}$.

Clearly, J is a repair of $db(\phi)$. We next show $J \not\models q$. Assume, on the contrary, $J \models q$. Then, there exists a valuation θ such that $\{P(\underline{\theta(y)}, \theta(u), \theta(x)),\ S(\underline{\theta(v), \theta(u)}),\ N(\underline{\theta(x), \theta(z)}, \theta(w)),\ T(\underline{\theta(v), \theta(w)})\} \subseteq J$. Clearly, $\theta(x) = a$. Assume $\theta(u) = p$. For each S-fact $S(\underline{s,t}) \in J$, we have $s = t$; hence $\theta(v) = p$. For each T-fact $T(\underline{s,t}) \in J$, we have $s = t$; hence $\theta(w) = p$. Since J contains $P(\underline{\theta(y)}, p, a)$, $B(p) = \mathbf{true}$, and since J contains $N(\underline{a, \theta(z)}, p)$, $B(p) = \mathbf{false}$, a contradiction. We conclude by contradiction that $J \not\models q$. It follows $db(\phi) \not\models_{\mathsf{sure}} q$.

For the *if* part, assume $db(\phi) \not\models_{\mathsf{sure}} q$. We can assume a repair J of $db(\phi)$ such that $J \not\models q$. Clearly, J must contain all S-facts and T-facts from $db(\phi)$. Construct a truth assignment B as follows:

- if J contains $P(\underline{i,p,a})$ for some $i \in \{1,2,\ldots,n\}$, then $B(p) = \mathbf{true}$; and
- if J contains $N(\underline{a,j,p})$ for some $j \in \{1,2,\ldots,n\}$, then $B(p) = \mathbf{false}$.

Note that J cannot contain both $P(\underline{i,p,a})$ and $N(\underline{a,j,p})$, or else $J \models q$, a contradiction. Clearly, B can be extended to a truth assignment satisfying ϕ. □

Appendix B. Proof of Theorem 1

Definition 9. Let ψ be a FO formula with free variables $x_1,\ldots,x_n$. Let θ be a valuation over $\{x_1,\ldots,x_n\}$. Then, $\theta(\psi)$ denotes the FO formula obtained from ψ by replacing each free occurrence of x_i with $\theta(x_i)$, for all $1 \leqslant i \leqslant n$.

Lemma 7. *Let V be a set of variables. Let q_1, q_2 be conjunctions of constant-free atoms containing no variable of V. Let φ be a conjunction of equalities. For every valuation ω over V,* $\mathsf{Rew}_{q_1}(q_2 \wedge \omega(\varphi)) = \omega(\mathsf{Rew}_{q_1}(q_2 \wedge \varphi))$.

Proof. The proof runs by induction on the length of q_2. The basis of the induction, $q_2 = \{\}$, is trivial. For the induction step, assume $q_2 = R(\vec{\underline{x}}, \vec{y}) \wedge q_3$. We have

$$\mathsf{Rew}_{q_1}(R(\vec{\underline{x}}, \vec{y}) \wedge q_3 \wedge \omega(\varphi)) = \left(\bigvee_{R(\vec{\underline{v}}, \vec{w}) \in q_1} \exists \vec{x} \exists \vec{y} \left(\begin{array}{ll} & (\vec{x} = \vec{v}) \\ \wedge & (\vec{y} = \vec{w}) \\ \wedge & \mathsf{Rew}_{q_1}(q_3 \wedge \omega(\varphi)) \end{array} \right) \right)$$

$\vee$

$$\exists \vec{x} \exists \vec{y} (R(\vec{\underline{x}}, \vec{y}) \wedge (\textstyle\bigwedge_{R(\vec{\underline{v}}, \vec{w}) \in q_1} (\vec{x} \neq \vec{v})) \wedge \forall \vec{y} (R(\vec{\underline{x}}, \vec{y}) \rightarrow \mathsf{Rew}_{q_1 \cup \{R(\vec{\underline{x}}, \vec{y})\}}(q_3 \wedge \omega(\varphi))))$$

By the induction hypothesis,

$$\mathsf{Rew}_{q_1}(R(\vec{\underline{x}}, \vec{y}) \wedge q_3 \wedge \omega(\varphi)) = \left(\bigvee_{R(\vec{\underline{v}}, \vec{w}) \in q_1} \exists \vec{x} \exists \vec{y} \left(\begin{array}{ll} & (\vec{x} = \vec{v}) \\ \wedge & (\vec{y} = \vec{w}) \\ \wedge & \omega(\mathsf{Rew}_{q_1}(q_3 \wedge \varphi)) \end{array} \right) \right)$$

$\vee$

$$\exists \vec{x} \exists \vec{y} (R(\vec{\underline{x}}, \vec{y}) \wedge (\textstyle\bigwedge_{R(\vec{\underline{v}}, \vec{w}) \in q_1} (\vec{x} \neq \vec{v})) \wedge \forall \vec{y} (R(\vec{\underline{x}}, \vec{y}) \rightarrow \omega(\mathsf{Rew}_{q_1 \cup \{R(\vec{\underline{x}}, \vec{y})\}}(q_3 \wedge \varphi))))$$

Since ω is the identity on every variable that occurs in q_1, it follows $\mathsf{Rew}_{q_1}(R(\vec{\underline{x}}, \vec{y}) \wedge q_3 \wedge \omega(\varphi)) = \omega(\mathsf{Rew}_{q_1}(R(\vec{\underline{x}}, \vec{y}) \wedge q_3 \wedge \varphi))$. □

Proof of Theorem 1. Let $q_1 \wedge q_2 \wedge \varphi$ be a key-rooted ordered rule in equational form. Thus, $q_1 \wedge q_2$ is

constant-free and contains no two occurrences of the same variable. In particular, q_1 and q_2 have no variables in common. The expression φ is a satisfiable set of equations involving variables of $q_1 \wedge q_2$ and constants.

Let V_1 be the set of variables that occur in q_1. Let I be a database. Let θ be a valuation over V_1 such that:

Consistency: $\theta(q_1)$ is a consistent subset of I; and

Key-rootedness: $\theta(q_2 \wedge \varphi) = q_2 \wedge \theta(\varphi)$ is key-rooted.

We define the following shorthand:

$$\left[\!\!\left[\begin{matrix} I \\ \theta \\ q_1 \end{matrix} \right]\!\!\right] = (I \backslash [\![\theta(q_1)]\!]_I) \cup \theta(q_1).$$

It is easy to see that if some variable x has a free occurrence in $\mathsf{Rew}_{q_1}(q_2 \wedge \varphi)$, then $x \in V_1$. Hence, $\theta(\mathsf{Rew}_{q_1}(q_2 \wedge \varphi))$ is a closed formula. We prove that

$$\left[\!\!\left[\begin{matrix} I \\ \theta \\ q_1 \end{matrix} \right]\!\!\right] \models_{\mathsf{sure}} q_2 \wedge \theta(\varphi) \iff I \models \theta(\mathsf{Rew}_{q_1}(q_2 \wedge \varphi)).$$

The desired result follows by choosing $q_1 = \{\}$.

The proof runs by induction on q_2's length. The proof is trivial if q_2 is empty. For q_2 nonempty (let $q_2 = R(\underline{\vec{x}}, \vec{y}) \wedge q_3$) the result follows by equivalence of the statements (1)–(7):

1. $\left[\!\!\left[\begin{matrix} I \\ \theta \\ q_1 \end{matrix} \right]\!\!\right] \models_{\mathsf{sure}} R(\underline{\vec{x}}, \vec{y}) \wedge q_3 \wedge \theta(\varphi).$

2. Since $R(\underline{\vec{x}}, \vec{y}) \wedge q_3 \wedge \theta(\varphi)$ is key-rooted, it follows that for some valuation ξ of $\vec{x}$,

   $$\left[\!\!\left[\begin{matrix} I \\ \theta \\ q_1 \end{matrix} \right]\!\!\right] \models_{\mathsf{sure}} R(\xi(\underline{\vec{x}}), \vec{y}) \wedge q_3 \wedge \xi \circ \theta(\varphi).$$

3. For some valuation ξ of $\vec{x}$, for some valuation υ of $\vec{y}$, either:
   - for some atom $R(\underline{\vec{v}}, \vec{w}) \in q_1$, we have $R(\theta(\vec{v}), \theta(\vec{w})) = R(\xi(\vec{x}), \upsilon(\vec{y}))$ and

     $$\left[\!\!\left[\begin{matrix} I \\ \theta \\ q_1 \end{matrix} \right]\!\!\right] \models_{\mathsf{sure}} q_3 \wedge \theta \circ \upsilon \circ \xi(\varphi);$$ or

   - $R(\xi(\underline{\vec{x}}), \upsilon(\vec{y})) \in I \backslash [\![\theta(q_1)]\!]_I$ and for each valuation υ' of $\vec{y}$, if $R(\xi(\underline{\vec{x}}), \upsilon'(\vec{y})) \in I$, then

     $$\left[\!\!\left[\begin{matrix} I \\ \upsilon' \circ \xi \circ \theta \\ q_1 \cup \{R(\underline{\vec{x}}, \vec{y})\} \end{matrix} \right]\!\!\right] \models_{\mathsf{sure}} q_3 \wedge \upsilon' \circ \xi \circ \theta(\varphi).$$

   Concerning the first bulleted item, since $\theta(q_1)$ is consistent, every repair of the database $\left[\!\!\left[\begin{matrix} I \\ \theta \\ q_1 \end{matrix} \right]\!\!\right]$ contains $R(\theta(\vec{v}), \theta(\vec{w}))$. Concerning both items, since $R(\underline{\vec{x}}, \vec{y}) \wedge q_3 \wedge \theta(\varphi)$ is key-rooted, so is $q_3 \wedge \omega \circ \theta(\varphi)$ for any valuation ω of $\vec{x}\vec{y}$.

   The transition between (3) and (4) applies the induction hypothesis twice on rules of shorter length:
   - first,

     $$\left[\!\!\left[\begin{matrix} I \\ \theta \\ q_1 \end{matrix} \right]\!\!\right] \models_{\mathsf{sure}} q_3 \wedge \theta \circ \upsilon \circ \xi(\varphi)$$

     $$\Updownarrow$$

     $$I \models \theta(\mathsf{Rew}_{q_1}(q_3 \wedge \upsilon \circ \xi(\varphi))).$$

     The preconditions for the induction hypothesis are fulfilled, because:

     Consistency: $\theta(q_1)$ is a consistent subset of I.

     Key-rootedness: Since $q_2 \wedge \theta(\varphi) = R(\underline{\vec{x}}, \underline{\vec{y}}) \wedge q_3 \wedge \theta(\varphi)$ is key-rooted and $\upsilon \circ \xi$ is a valuation of $\vec{x}\vec{y}$, $q_3 \wedge \theta \circ \upsilon \circ \xi(\varphi)$ is key-rooted.

     Note that by Lemma 7,

     $$\theta(\mathsf{Rew}_{q_1}(q_3 \wedge \upsilon \circ \xi(\varphi))) = \theta \circ \upsilon \circ \xi(\mathsf{Rew}_{q_1}(q_3 \wedge \varphi)).$$

   - second,

     $$\left[\!\!\left[\begin{matrix} I \\ \upsilon' \circ \xi \circ \theta \\ q_1 \cup \{R(\underline{\vec{x}}, \vec{y})\} \end{matrix} \right]\!\!\right] \models_{\mathsf{sure}} q_3 \wedge \upsilon' \circ \xi \circ \theta(\varphi)$$

     $$\Updownarrow$$

     $$I \models \upsilon' \circ \xi \circ \theta(\mathsf{Rew}_{q_1 \cup \{R(\underline{\vec{x}}, \vec{y})\}}(q_3 \wedge \varphi)).$$

     The preconditions for the induction hypothesis are fulfilled, because:

     Consistency: $\upsilon' \circ \xi \circ \theta(q_1 \cup \{R(\underline{\vec{x}}, \vec{y})\}) = \theta(q_1) \cup \{R(\xi(\underline{\vec{x}}), \upsilon(\vec{y}))\}$ is a consistent subset of I since $R(\xi(\underline{\vec{x}}), \upsilon(\vec{y})) \in I \backslash [\![\theta(q_1)]\!]_I$.

     Key-rootedness: $q_3 \wedge \upsilon' \circ \xi \circ \theta(\varphi)$ is key-rooted.

4. Therefore, by the induction hypothesis and by Lemma 7, for some valuation ξ of $\vec{x}$, for some valuation υ of $\vec{y}$, either:
   - for some $R(\underline{\vec{v}}, \vec{w}) \in q_1$, I satisfies

     $$\begin{pmatrix} & (\xi(\vec{x}) = \theta(\vec{v})) \\ \wedge & (\upsilon(\vec{y}) = \theta(\vec{w})) \\ \wedge & \theta \circ \upsilon \circ \xi(\mathsf{Rew}_{q_1}(q_3 \wedge \varphi)) \end{pmatrix};$$ or

   - $R(\xi(\underline{\vec{x}}), \upsilon(\vec{y})) \in I \backslash [\![\theta(q_1)]\!]_I$ and for each valuation υ' of $\vec{y}$, if $R(\xi(\underline{\vec{x}}), \upsilon'(\vec{y})) \in I$, then

     $$I \models \upsilon' \circ \xi \circ \theta(\mathsf{Rew}_{q_1 \cup \{R(\underline{\vec{x}}, \vec{y})\}}(q_3 \wedge \varphi)).$$

5. For some valuation ξ of $\vec{x}$, for some valuation υ of $\vec{y}$, either:
   - I satisfies

     $$\bigvee_{R(\underline{\vec{v}}, \vec{w}) \in q_1} \begin{pmatrix} & (\xi(\vec{x}) = \theta(\vec{v})) \\ \wedge & (\upsilon(\vec{y}) = \theta(\vec{w})) \\ \wedge & \theta \circ \upsilon \circ \xi(\mathsf{Rew}_{q_1}(q_3 \wedge \varphi)) \end{pmatrix};$$ or

   - $R(\xi(\underline{\vec{x}}), \upsilon(\vec{y})) \in I$ and $(\bigwedge_{R(\underline{\vec{v}}, \vec{w}) \in q_1} \xi(\vec{x}) \neq \theta(\vec{v}))$ and for every valuation υ' of $\vec{y}$ such that $R(\xi(\underline{\vec{x}}), \upsilon'(\vec{y})) \in I$,

     $$I \models \upsilon' \circ \xi \circ \theta(\mathsf{Rew}_{q_1 \cup \{R(\underline{\vec{x}}, \vec{y})\}}(q_3 \wedge \varphi)).$$

6. After some rearranging, either

$$I \models \theta\left(\bigvee_{R(\underline{\vec{v}},\vec{w})\in q_1} \exists\vec{x}\exists\vec{y}\left(\begin{array}{ll} & (\vec{x}=\vec{v})\\ \wedge & (\vec{y}=\vec{w})\\ \wedge & \mathsf{Rew}_{q_1}(q_3\wedge\varphi))\end{array}\right)\right)$$

or

$$I \models \theta[\exists\vec{x}\exists\vec{y}(R(\underline{\vec{x}},\vec{y})\wedge(\textstyle\bigwedge_{R(\underline{\vec{v}},\vec{w})\in q_1}\vec{x}\neq\vec{v})\wedge \forall\vec{y}(R(\underline{\vec{x}},\vec{y})\rightarrow \mathsf{Rew}_{q_1\cup\{R(\underline{\vec{x}},\vec{y})\}}(q_3\wedge\varphi)))].$$

7. $I \models \theta(\mathsf{Rew}_{q_1}(R(\underline{\vec{x}},\vec{y})\wedge q_3\wedge\varphi))$. □

Appendix C. Proof of Theorem 2

The proof runs by induction on the length of q. The case $m\leqslant 1$ is trivial. Next assume $m>1$. Since $\vec{x}_1$ contains no variables, $R_1(\underline{\vec{x}_1},\vec{y}_1)$ is obviously reifiable.

Let $q'=\langle R(\underline{\vec{x}_2},\vec{y}_2),\ldots,R(\underline{\vec{x}_m},\vec{y}_m)\rangle$, the tail of q. We still need to show that $\theta(q')$ is key-rooted for every valuation θ of $\vec{x}_1\vec{y}_1$. Let i be a child of 1 in the number join tree τ with root 1. Two cases can occur:

1. i is a leaf vertex in τ. Thus, $\theta(sub_q^\tau(i))$ is a singleton. We show hereafter that every singleton rule is key-rooted.
2. i is an internal vertex in τ. The construction in the proof of Lemma 5 allows to build a number join tree (call it τ_i) for the ordered rule $sub_q^\tau(i)$. By Lemma 4, τ_i is a number join tree for $\theta(sub_q^\tau(i))$. It is now easy to check that the ordered rule $\theta(sub_q^\tau(i))$, of smaller length than q, satisfies the conditions of the theorem's statement.Hence, $\theta(sub_q^\tau(i))$ is key-rooted by the induction hypothesis.

Since i is arbitrary, $\theta(sub_q^\tau(i))$ is key-rooted for every child i of 1. By Corollary 1, $\theta(q')$ is key-rooted.

To show that every singleton rule $q_s=R(\underline{\vec{x}},\vec{y})$ is key-rooted, it suffices to show that $R(\underline{\vec{x}},\vec{y})$ is reifiable in q_s. Let I be a database. Two cases can occur:

- For every atom $R(\underline{\vec{a}},\vec{b})\in I$, there exists a key-equal atom $R(\underline{\vec{a}},\vec{c})\in I$ such that $R(\underline{\vec{a}},\vec{c})\not\models q_s$. Then, obviously, there exists a repair J of I such that $J\not\models q_s$, hence $I\not\models_{\mathsf{sure}} q_s$.
- I contains an atom $R(\underline{\vec{a}},\vec{b})$ such that for every key-equal atom $R(\underline{\vec{a}},\vec{c})\in I$, $R(\underline{\vec{a}},\vec{c})\models q_s$. Let θ be the valuation of $\vec{x}$ such that $\theta(\vec{x})=\vec{a}$. Since every repair J of I contains an atom of $[\![R(\underline{\vec{a}},\vec{b})]\!]_I$, it follows $I\models_{\mathsf{sure}}\theta(q_s)$.

Since I is arbitrary, it follows that $R(\underline{\vec{x}},\vec{y})$ is reifiable in q_s.

Appendix D. Proof of Theorems 3 and 4

Definition 10. Let $\vec{x}$ be a sequence of symbols and $\vec{a}$ a sequence of constants such that $|\vec{a}|=|\vec{x}|$. The valuation θ of $\vec{x}$ such that $\theta(\vec{x})=\vec{a}$, if it exists, will be denoted by $\mathsf{id}[\vec{x}\mapsto\vec{a}]$.

Let I be a database and $A\in I$. We define the following shorthand:

$$\left[\!\!\left[\begin{array}{l} I\\ A\end{array}\right]\!\!\right]=(I\backslash[\![A]\!]_I)\cup\{A\}.$$

Lemma 8. *Let q be a rule and I a database. Let V be the set of variables in q, and $X\subseteq V$. Let $R(\underline{\vec{a}},\vec{b})\in I$ such that for no valuation θ over V, $R(\underline{\vec{a}},\vec{b})\in\theta(q)$. Let $I_0=I\backslash[\![R(\underline{\vec{a}},\vec{b})]\!]_I$. Every repair J_0 of I_0 can be extended to a repair J of I such that for every valuation μ over X, $J\models\mu(q)$ implies $J_0\models\mu(q)$.*

Proof. Let J_0 be a repair of I_0. Let $J=J_0\cup\{R(\underline{\vec{a}},\vec{b})\}$. Clearly, J is a repair of I. Let μ be a valuation over X such that $J\models\mu(q)$. Hence, there exists a valuation ω over V such that $\omega(x)=\mu(x)$ for each $x\in X$ and $\omega(q)\subseteq J$. Since $R(\underline{\vec{a}},\vec{b})\notin\omega(q)$, it follows $\omega(q)\subseteq J_0$. Hence, $J_0\models\mu(q)$. □

Proof of Theorem 3. Let h be the height of the number join tree for q. Let J_1,J_2 be two repairs of the same database I. We will construct a sequence of sets of atoms

$$\{\}=N_0\subseteq N_1\subseteq\cdots\subseteq N_h\subseteq J_1\cup J_2$$

such that for each $d\in\{0,\ldots,h\}$:

(P1) N_d is consistent.
(P2) For all key-equal atoms $R(\underline{\vec{a}},\vec{b}),R(\underline{\vec{a}},\vec{c})$, one from J_1 and the other from J_2 (possibly $\vec{b}=\vec{c}$),
  (a) if N_d contains neither $R(\underline{\vec{a}},\vec{b})$ nor $R(\underline{\vec{a}},\vec{c})$, then $J_1\cup J_2\models_{\mathsf{sure}}\mathsf{id}[\vec{x}_1\mapsto\vec{a}](subd_q^\tau(1,d))$; and
  (b) if N_d contains $R(\underline{\vec{a}},\vec{b})$ or $R(\underline{\vec{a}},\vec{c})$, then for each repair J of $J_1\cup J_2$ such that $N_d\subseteq J$, $J\not\models\mathsf{id}[\vec{x}_1\mapsto\vec{a}](subd_q^\tau(1,d))$.

If this construction succeeds, then the root vertex $R(\underline{\vec{x}_1},\vec{y}_1)$ is reifiable. Indeed, by property (P1), there exists a repair J of $J_1\cup J_2$ (and hence of I) such that $N_h\subseteq J$. Notice that $q=subd_q^\tau(1,h)$. Assume $J\models\mathsf{id}[\vec{x}_1\mapsto\vec{a}](q)$. Let $R(\underline{\vec{a}},\vec{b}),R(\underline{\vec{a}},\vec{c})$ be key-equal atoms, one from J_1 and the other from J_2. By property (P2b), $N_h\subseteq J$ and $J\models\mathsf{id}[\vec{x}_1\mapsto\vec{a}](q)$ imply that N_h contains neither $R(\underline{\vec{a}},\vec{b})$ nor $R(\underline{\vec{a}},\vec{c})$.Then, by property (P2a), $J_1\models\mathsf{id}[\vec{x}_1\mapsto\vec{a}](q)$ and $J_2\models\mathsf{id}[\vec{x}_1\mapsto\vec{a}](q)$. Since $\vec{a}$ is arbitrary, it follows that $\mathsf{Reifies}(q,\vec{x}_1,J)\subseteq\mathsf{Reifies}(q,\vec{x}_1,J_1)$ and $\mathsf{Reifies}(q,\vec{x}_1,J)\subseteq\mathsf{Reifies}(q,\vec{x}_1,J_2)$. Then, by Lemma 1, $R(\underline{\vec{x}_1},\vec{y}_1)$ is reifiable.

Let $q'=\langle R(\underline{\vec{x}_2},\vec{y}_2),\ldots,R(\underline{\vec{x}_m},\vec{y}_m)\rangle$, the tail of q. We still need to show that $\theta(q')$ is key-rooted for every valuation θ of $\vec{x}_1\vec{y}_1$. Let i be a child of 1 in the number join tree τ with root 1. The construction in the proof of Lemma 5 allows to build a number join tree (call it τ_i) for the ordered rule $sub_q^\tau(i)$. By Lemma 4, τ_i is also a number join tree for $\theta(sub_q^\tau(i))$. By condition (4) in the theorem's statement and by Theorem 2, the ordered rule $\theta(sub_q^\tau(i))$ is key-rooted. By Corollary 1, $\theta(q')$ is key-rooted.

In the remainder of the proof, we show the construction of N_d for each $d\in\{0,\ldots,h\}$. The construction runs by induction on increasing d. Since $N_0=\{\}$, N_0 trivially satisfies properties (P1) and (P2b); N_0 satisfies property (P2a) because $subd_q^\tau(1,0)=R(\underline{\vec{x}_1},\vec{y}_1)$ and $\vec{x}_1\vec{y}_1$ is a sequence

of distinct variables (condition (1) in the theorem's statement).

Construction of N_{d+1} from N_d: For all key-equal atoms $R(\underline{\vec{a}}, \vec{b}), R(\underline{\vec{a}}, \vec{c})$, one from J_1 and the other from J_2, and both not in N_d, whenever

$$J_1 \cup J_2 \not\models_{\mathsf{sure}} \mathsf{id}[\vec{x}_1 \mapsto \vec{a}](subd_q^\tau(1, d+1)),$$

then select $A \in \{R(\underline{\vec{a}}, \vec{b}), R(\underline{\vec{a}}, \vec{c})\}$ such that

$$\left[\!\!\left[\begin{matrix} J_1 \cup J_2 \\ A \end{matrix} \right]\!\!\right] \not\models_{\mathsf{sure}} \mathsf{id}[\vec{x}_1 \mapsto \vec{a}](subd_q^\tau(1, d+1)) \tag{5}$$

and include $A \in N_{d+1}$. Henceforth, assume w.l.o.g. that $A = R(\underline{\vec{a}}, \vec{b})$.

Note that if $J_1 \cup J_2 \models_{\mathsf{sure}} \mathsf{id}[\vec{x}_i \mapsto \vec{a}](subd_q^\tau(1, d+1))$, then N_{d+1} contains neither $R(\underline{\vec{a}}, \vec{b})$ nor $R(\underline{\vec{a}}, \vec{c})$.

In this way, N_{d+1} obviously satisfies properties (P1) and (P2a). To show that N_{d+1} satisfies property (P2b), assume $R(\underline{\vec{a}}, \vec{b}) \in N_{d+1}$. Let J be a repair of $J_1 \cup J_2$ such that $N_{d+1} \subseteq J$. Two cases can occur.

1. Case $R(\underline{\vec{a}}, \vec{b}) \in N_d$. By the induction hypothesis,

   $J \not\models \mathsf{id}[\vec{x}_1 \mapsto \vec{a}](subd_q^\tau(1, d))$.

   Obviously, $J \not\models \mathsf{id}[\vec{x}_1 \mapsto \vec{a}](subd_q^\tau(1, d+1))$.
2. Case $R(\underline{\vec{a}}, \vec{b}) \notin N_d$. Then, by condition (5) in our construction

   $$\left[\!\!\left[\begin{matrix} J_1 \cup J_2 \\ R(\underline{\vec{a}}, \vec{b}) \end{matrix} \right]\!\!\right] \not\models_{\mathsf{sure}} \mathsf{id}[\vec{x}_1 \mapsto \vec{a}](subd_q^\tau(1, d+1)).$$

   Consequently, we can assume the existence of a child j of 1 in the number join tree τ with root 1 such that

   $$\left[\!\!\left[\begin{matrix} J_1 \cup J_2 \\ R(\underline{\vec{a}}, \vec{b}) \end{matrix} \right]\!\!\right] \not\models_{\mathsf{sure}} \mathsf{id}[\vec{x}_1\vec{y}_1 \mapsto \vec{a}\vec{b}](subd_q^\tau(j, d)).$$

   Assume $\mathsf{id}[\vec{x}_1\vec{y}_1 \mapsto \vec{a}\vec{b}](\vec{x}_j) = \vec{e}$. Since $\mathsf{vars}(\vec{x}_1\vec{y}_1) \cap \mathsf{vars}(\vec{x}_j\vec{y}_j) = \mathsf{vars}(\vec{x}_j)$ by condition (4) in the theorem's statement,

   $$J_1 \cup J_2 \not\models_{\mathsf{sure}} \mathsf{id}[\vec{x}_j \mapsto \vec{e}](subd_q^\tau(j, d)). \tag{6}$$

   Consider three mutually exclusive and exhaustive cases:
   (a) N_d contains an R-atom with primary key value $\vec{e}$. By the induction hypothesis,

   $J \not\models \mathsf{id}[\vec{x}_1 \mapsto \vec{e}](subd_q^\tau(1, d))$.

   By conditions (2) and (3) in the theorem's statement, $subd_q^\tau(1, d)$ and $subd_q^\tau(j, d)$ are the same up to a renaming of variables, hence

   $J \not\models \mathsf{id}[\vec{x}_j \mapsto \vec{e}](subd_q^\tau(j, d))$.

   It follows $J \not\models \mathsf{id}[\vec{x}_1 \mapsto \vec{a}](subd_q^\tau(1, d+1))$.
   (b) $J_1 \cup J_2$ contains no R-atom with primary key value $\vec{e}$. Obviously, $J \not\models \mathsf{id}[\vec{x}_1 \mapsto \vec{a}](subd_q^\tau(1, d+1))$.
   (c) $J_1 \cup J_2$ contains an R-atom with primary key value $\vec{e}$, but N_d does not. By the induction hypothesis,

   $J_1 \cup J_2 \models_{\mathsf{sure}} \mathsf{id}[\vec{x}_1 \mapsto \vec{e}](subd_q^\tau(1, d))$,

   which contradicts Eq. (6). We conclude by contradiction that this case cannot occur. □

Proof of Theorem 4. The proof runs by induction on the length of q. The proof is trivial if $|q| = 0$; next assume that $|q| > 0$.

Let I be a database. Let J_1 and J_2 be two repairs of I. We are going to show the existence of a repair J of $J_1 \cup J_2$ such that for every valuation θ of $\vec{x}_1$, if $J \models \theta(q)$, then $J_1 \models \theta(q)$ and $J_2 \models \theta(q)$. By Lemma 8, we can assume w.l.o.g. that for every atom $R_i(\underline{\vec{x}_i}, \vec{y}_i) \in q$ and for every $R_i(\underline{\vec{a}}, \vec{b}) \in J_1 \cup J_2$, $\mathsf{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}]$ is well defined.

For each $i \in \{1, \ldots, m\}$, we construct a set $N(i) \subseteq J_1 \cup J_2$ such that:

(P1) $N(i)$ is consistent.
(P2) For all key-equal atoms $R_i(\underline{\vec{a}}, \vec{b}), R_i(\underline{\vec{a}}, \vec{c})$, one from J_1 and the other from J_2 (possibly $\vec{b} = \vec{c}$),
   (a) if $N(i)$ contains neither $R_i(\underline{\vec{a}}, \vec{b})$ nor $R_i(\underline{\vec{a}}, \vec{c})$, then $J_1 \cup J_2 \models_{\mathsf{sure}} \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$; and
   (b) if $N(i)$ contains $R_i(\underline{\vec{a}}, \vec{b})$ or $R_i(\underline{\vec{a}}, \vec{c})$, then for each repair J of $J_1 \cup J_2$ such that $N(i) \subseteq J$, $J \not\models \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$.

Then, by Lemma 1, $R_1(\underline{\vec{x}_1}, \vec{y}_1)$ is reifiable; the argumentation is the same as in the proof of Theorem 3.

Let $q' = \langle R(\underline{\vec{x}_2}, \vec{y}_2), \ldots, R(\underline{\vec{x}_m}, \vec{y}_m)\rangle$, the tail of q. We still need to show that $\theta(q')$ is key-rooted for every valuation θ of $\vec{x}_1\vec{y}_1$. Let i be a child of 1 in the number join tree τ with root 1. The construction in the proof of Lemma 5 allows to build a number join tree (call it τ_i) for the ordered rule $sub_q^\tau(i)$. By Lemma 4, τ_i is also a number join tree for $\theta(sub_q^\tau(i))$. It can now be easily checked that the ordered rule $\theta(sub_q^\tau(i))$, which is of smaller length than q, satisfies the conditions of the theorem's statement, hence $\theta(sub_q^\tau(i))$ is key-rooted by the induction hypothesis. By Corollary 1, $\theta(q')$ is key-rooted.

The construction of each $N(i)$ runs by induction on decreasing i, that is, in decreasing depth of the number join tree τ with root 1. Thus, if i is the parent of j, then $N(j)$ is computed before $N(i)$. The construction is specified next.

Construction of $N(i)$ for leaf vertex i: *Let* $N(i) = \{\}$.

By our assumption in the beginning of the proof, for every $R_i(\underline{\vec{a}}, \vec{b}) \in J_1 \cup J_2$, $\{R_i(\underline{\vec{a}}, \vec{b})\} \models R_i(\underline{\vec{x}_i}, \vec{y}_i)$. Obviously, if i is a leaf vertex, then $N(i)$ satisfies properties (P1) and (P2).

For the induction step, we assume that $i \in \{1, \ldots, m\}$ is an internal node. Assume that the children of i are $i+1, \ldots, i+k, \ldots, i+l$ such that for all $j \in \{1+1, \ldots, i+l\}$:

1. if $j \in \{i+1, \ldots, i+k\}$, then $\mathsf{vars}(\vec{x}_i) \subseteq \mathsf{vars}(\vec{x}_j)$ and $\mathsf{vars}(\vec{x}_i\vec{y}_i) \cap \mathsf{vars}(\vec{x}_j\vec{y}_j) \not\subseteq \mathsf{vars}(\vec{x}_j)$.
2. if $j \in \{i+k+1, \ldots, i+l\}$, then $\mathsf{vars}(\vec{x}_i\vec{y}_i) \cap \mathsf{vars}(\vec{x}_j\vec{y}_j) \subseteq \mathsf{vars}(\vec{x}_j)$.

This is illustrated in Fig. 15. We construct $N(i)$ as the smallest set containing $N(i+1),\ldots,N(i+l)$ and the atoms specified next.

Construction of $N(i)$ for internal vertex i: For all key-equal atoms $R_i(\underline{\vec{a}},\vec{b}), R_i(\underline{\vec{a}},\vec{c})$, one from J_1 and the other from J_2, whenever

$$J_1 \cup J_2 \not\models_{\mathsf{sure}} \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i)),$$

then:

(I1) Select $A \in \{R_i(\underline{\vec{a}},\vec{b}), R_i(\underline{\vec{a}},\vec{c})\}$ such that

$$\left[\!\!\left[\begin{matrix} J_1 \cup J_2 \\ A \end{matrix} \right]\!\!\right] \not\models_{\mathsf{sure}} \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$$

and include $A \in N(i)$. Henceforth, assume w.l.o.g. that $A = R_i(\underline{\vec{a}},\vec{b})$ (the other case is symmetrical).

(I2) Construct a repair $J_{\vec{a}}$ of $\left[\!\!\left[\begin{matrix} J_1 \cup J_2 \\ R_i(\underline{\vec{a}},\vec{b}) \end{matrix} \right]\!\!\right]$ such that $N(j) \subseteq J_{\vec{a}}$ for each child j of i and $J_{\vec{a}} \not\models \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$.

(I3) Select a child $j_{\vec{a}}$ of i such that

$$J_{\vec{a}} \not\models \mathsf{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}](sub_q^\tau(j_{\vec{a}})). \tag{7}$$

For each pair of distinct key-equal atoms $R_{j_{\vec{a}}}(\underline{\vec{d}},\vec{e})$, $R_{j_{\vec{a}}}(\underline{\vec{d}},\vec{f}) \in (J_1 \cup J_2)\backslash N(j_{\vec{a}})$, if $R_{j_{\vec{a}}}(\underline{\vec{d}},\vec{e}) \in J_{\vec{a}}$ and

$$J_{\vec{a}} \cup \{R_{j_{\vec{a}}}(\underline{\vec{d}},\vec{f})\} \models \mathsf{id}[\vec{x}_i\vec{y}_i\vec{x}_{j_{\vec{a}}}\vec{y}_{j_{\vec{a}}} \mapsto \vec{a}\vec{b}\vec{d}\vec{f}](sub_q^\tau(j_{\vec{a}})),$$

then add $R_{j_{\vec{a}}}(\underline{\vec{d}},\vec{e}) \in N(i)$.

Note that if $J_1 \cup J_2 \models_{\mathsf{sure}} \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$, then neither $R_i(\underline{\vec{a}},\vec{b})$ nor $R_i(\underline{\vec{a}},\vec{c})$ is included in $N(i)$.

Obviously, for any internal vertex i, $N(i)$ satisfies property (P2a). The construction is illustrated by the following example.

Example 13. Let $q = \langle R(\underline{x},z), S(\underline{x,y},z)\rangle$. Let

$$J_1 \cup J_2 = \{R(\underline{a},c), R(\underline{a},d), S(\underline{a,1},c), S(\underline{a,1},d), S(\underline{a,2},c), S(\underline{a,2},d)\}.$$

We have $N(2) = \{\}$. Next, depending on which of $R(\underline{a},c)$ or $R(\underline{a},d)$ is selected in step (I1), either

- $N(1) = \{R(\underline{a},c), S(\underline{a,1},d), S(\underline{a,2},d)\}$, or
- $N(1) = \{R(\underline{a},d), S(\underline{a,1},c), S(\underline{a,2},c)\}$.

The S-atoms in $N(1)\backslash N(2)$ are added in step (I3).

We show the existence of $J_{\vec{a}}$ in step (I2). Since

$$\left[\!\!\left[\begin{matrix} J_1 \cup J_2 \\ R_i(\underline{\vec{a}},\vec{b}) \end{matrix} \right]\!\!\right] \not\models_{\mathsf{sure}} \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i)),$$

we can assume a repair $G_{\vec{a}}$ of $J_1 \cup J_2$ such that $R_i(\underline{\vec{a}},\vec{b}) \in G_{\vec{a}}$ and

$$G_{\vec{a}} \not\models \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i)). \tag{8}$$

Let $J_{\vec{a}}$ be the repair of $\left[\!\!\left[\begin{matrix} J_1 \cup J_2 \\ R_i(\underline{\vec{a}},\vec{b}) \end{matrix} \right]\!\!\right]$ such that $N(j) \subseteq J_{\vec{a}}$ for each child j of i and $J_{\vec{a}}\backslash(\bigcup_{j=i+1}^{i+l} N(j)) \subseteq G_{\vec{a}}$. Thus, $J_{\vec{a}}$ contains $N(j)$ for each child j of i and is "completed" with atoms from $G_{\vec{a}}$.

We show $J_{\vec{a}} \not\models \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$. Assume, on the contrary,

$$J_{\vec{a}} \models \mathsf{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i)). \tag{9}$$

By the *Connectedness Condition*, for all children j_1, j_2 of i such that $j_1 \neq j_2$, the ordered rules $\mathsf{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}](sub_q^\tau(j_1))$ and $\mathsf{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}](sub_q^\tau(j_2))$ have no variables in common. Then, by Eq. (8), we can assume a child h of i such that

$$G_{\vec{a}} \not\models \mathsf{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}](sub_q^\tau(h)). \tag{10}$$

On the other hand, by Eq. (9), $J_{\vec{a}} \models \mathsf{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}](sub_q^\tau(h))$. Then, we can assume $R_h(\underline{\vec{d}},\vec{e}) \in J_{\vec{a}}$ such that:

- $J_{\vec{a}} \models \mathsf{id}[\vec{x}_i\vec{y}_i\vec{x}_h\vec{y}_h \mapsto \vec{a}\vec{b}\vec{d}\vec{e}](sub_q^\tau(h))$, and
- $G_{\vec{a}} \not\models \mathsf{id}[\vec{x}_i\vec{y}_i\vec{x}_h \mapsto \vec{a}\vec{b}\vec{d}](sub_q^\tau(h))$.

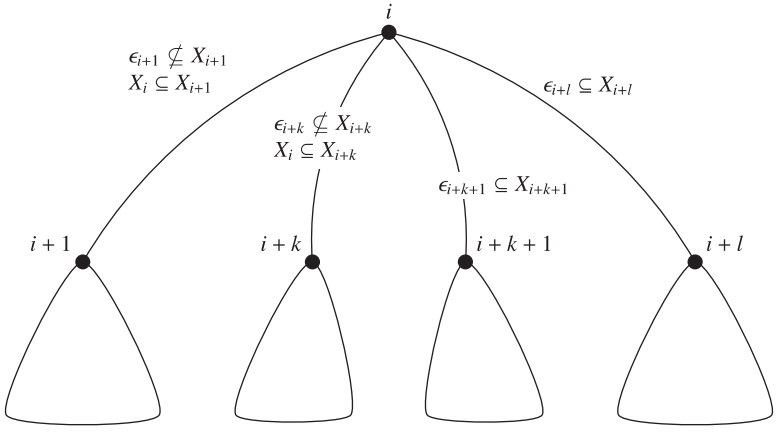

Fig. 15. Vertex numbering for an internal vertex i in the number join tree τ for $\langle R_1(\underline{\vec{x}_1},\vec{y}_1),\ldots,R_m(\underline{\vec{x}_m},\vec{y}_m)\rangle$, used in the proof of Theorem 4. We write X_i as a shorthand for $\mathsf{vars}(\vec{x}_i)$. Every edge label ε_i is the set of variables that occur in both the parent and the child node.

Then, $J_{\vec{a}}$ is a repair of $J_1 \cup J_2$ such that $N(h) \subseteq J_{\vec{a}}$ and

$J_{\vec{a}} \vDash \mathrm{id}[\vec{x}_h \mapsto \vec{d}](sub_q^\tau(h))$.

Since $N(h)$ satisfies property (P2b) by the induction hypothesis, $R_h(\underline{\vec{d}}, \vec{e}) \notin N(h)$. By the construction of $J_{\vec{a}}$, $R_h(\underline{\vec{d}}, \vec{e}) \in G_{\vec{a}}$ and $N(h)$ contains no atom that is key-equal to $R_h(\underline{\vec{d}}, \vec{e})$. Since $N(h)$ satisfies property (P2a) by the induction hypothesis, $G_{\vec{a}} \vDash \mathrm{id}[\vec{x}_h \mapsto \vec{d}](sub_q^\tau(h))$, hence $G_{\vec{a}} \vDash \mathrm{id}[\vec{x}_h\vec{y}_h \mapsto \vec{d}\vec{e}](sub_q^\tau(h))$. Since $R_h(\underline{\vec{d}}, \vec{e}) \in J_{\vec{a}}$ and $\mathrm{id}[\vec{x}_i\vec{y}_i\vec{x}_h\vec{y}_h \mapsto \vec{a}\vec{b}\vec{d}\vec{e}]$ is well defined, it follows that $\mathrm{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}]$ and $\mathrm{id}[\vec{x}_h\vec{y}_h \mapsto \vec{d}\vec{e}]$ agree on $\mathrm{vars}(\vec{x}_i\vec{y}_i) \cap \mathrm{vars}(\vec{x}_h\vec{y}_h)$. Consequently, $G_{\vec{a}} \vDash \mathrm{id}[\vec{x}_i\vec{y}_i\vec{x}_h \mapsto \vec{a}\vec{b}\vec{d}](sub_q^\tau(h))$, contradicting (10). We conclude by contradiction $J_{\vec{a}} \nvDash \mathrm{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$.

In step (I3), the integer $j_{\vec{a}}$ satisfying (7) exists, because by the *Connectedness Condition*, for all children j_1, j_2 of i such that $j_1 \neq j_2$, $\mathrm{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}](sub_q^\tau(j_1))$ and $\mathrm{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}](sub_q^\tau(j_2))$ have no variables in common.

We now show that the addition of some $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e}) \in N(i)$ in step (I3) of the above construction cannot possibly result in $N(i)$ becoming inconsistent (property (P1)). Assume, on the contrary, that (I3) also adds $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{f}) \in N(i)$ with $\vec{f} \neq \vec{e}$. Then, we can assume the existence of an atom $R_i(\underline{\vec{a}'}, \vec{b}')$ in step (I1), a repair $J_{\vec{a}'}$ in step (I2), and an integer $j_{\vec{a}'} = j_{\vec{a}}$ in step (I3) such that $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{f}) \in J_{\vec{a}'}$ and

$J_{\vec{a}'} \cup \{R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e})\} \vDash \mathrm{id}[\vec{x}_i\vec{y}_i\vec{x}_{j_{\vec{a}}}\vec{y}_{j_{\vec{a}}} \mapsto \vec{a}'\vec{b}'\vec{d}\vec{e}](sub_q^\tau(j_{\vec{a}}))$.

It can be easily seen that the condition for adding $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e})$ to $N(i)$ in step (I3) requires $j_{\vec{a}} \in \{i+1, \ldots, i+k\}$. It follows $\mathrm{vars}(\vec{x}_i) \subseteq \mathrm{vars}(\vec{x}_{j_{\vec{a}}})$. Then, since the valuations $\mathrm{id}[\vec{x}_i\vec{y}_i\vec{x}_{j_{\vec{a}}}\vec{y}_{j_{\vec{a}}} \mapsto \vec{a}\vec{b}\vec{d}\vec{f}]$ and $\mathrm{id}[\vec{x}_i\vec{y}_i\vec{x}_{j_{\vec{a}}}\vec{y}_{j_{\vec{a}}} \mapsto \vec{a}'\vec{b}'\vec{d}\vec{e}]$ agree on $\vec{x}_{j_{\vec{a}}}$, they must agree on $\vec{x}_i$, hence $\vec{a} = \vec{a}'$, so $J_{\vec{a}} = J_{\vec{a}'}$. Then, $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e})$, $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{f}) \in J_{\vec{a}}$, a contradiction. It is correct to conclude by contradiction that $N(i)$ satisfies property (P1) for any internal vertex i.

Finally, we show that $N(i)$ satisfies property (P2b). Assume a repair J of $J_1 \cup J_2$ such that $R_i(\underline{\vec{a}}, \vec{b}) \in N(i) \subseteq J$. We need to show $J \nvDash \mathrm{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$. Assume, on the contrary, $J \vDash \mathrm{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$.

By (I2) and (I3), since $R_i(\underline{\vec{a}}, \vec{b}) \in N(i)$, there exists a repair $J_{\vec{a}}$ of $J_1 \cup J_2$ and a child $j_{\vec{a}}$ of i such that $R_i(\underline{\vec{a}}, \vec{b}) \in J_{\vec{a}}$ and $J_{\vec{a}} \nvDash \mathrm{id}[\vec{x}_i\vec{y}_i \mapsto \vec{a}\vec{b}](sub_q^\tau(j_{\vec{a}}))$.

Since $J \vDash \mathrm{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$, we can assume key-equal atoms $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e})$, $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{f})$, one from J_1 and the other from J_2, such that $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{f}) \in J$ and

$J \vDash \mathrm{id}[\vec{x}_i\vec{y}_i\vec{x}_{j_{\vec{a}}}\vec{y}_{j_{\vec{a}}} \mapsto \vec{a}\vec{b}\vec{d}\vec{f}](sub_q^\tau(j_{\vec{a}}))$.

Hence, $\mathrm{id}[\vec{x}_i\vec{y}_i\vec{x}_{j_{\vec{a}}}\vec{y}_{j_{\vec{a}}} \mapsto \vec{a}\vec{b}\vec{d}\vec{f}]$ is well defined. Since $N(j_{\vec{a}})$ satisfies property (P2b) by the induction hypothesis and since $N(j_{\vec{a}}) \subseteq N(i) \subseteq J$ and $J \vDash \mathrm{id}[\vec{x}_{j_{\vec{a}}} \mapsto \vec{d}](sub_q^\tau(j_{\vec{a}}))$, it follows $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{f}) \notin N(j_{\vec{a}})$ and $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e}) \notin N(j_{\vec{a}})$. Then, since $N(j_{\vec{a}})$ satisfies property (P2a) by the induction hypothesis,

$J_1 \cup J_2 \vDash_{\mathsf{sure}} \mathrm{id}[\vec{x}_{j_{\vec{a}}} \mapsto \vec{d}](sub_q^\tau(j_{\vec{a}}))$.

Since $(J_{\vec{a}} \setminus \{R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e})\}) \cup \{R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{f})\}$ is a repair of $J_1 \cup J_2$,

$(J_{\vec{a}} \setminus \{R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e})\}) \cup \{R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{f})\} \vDash \mathrm{id}[\vec{x}_{j_{\vec{a}}} \mapsto \vec{d}](sub_q^\tau(j_{\vec{a}}))$.

Since $R_i(\underline{\vec{a}}, \vec{b}) \in J_{\vec{a}}$ and since $\mathrm{id}[\vec{x}_i\vec{y}_i\vec{x}_{j_{\vec{a}}}\vec{y}_{j_{\vec{a}}} \mapsto \vec{a}\vec{b}\vec{d}\vec{f}]$ is well defined,

$J_{\vec{a}} \cup \{R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{f})\} \vDash \mathrm{id}[\vec{x}_i\vec{y}_i\vec{x}_{j_{\vec{a}}}\vec{y}_{j_{\vec{a}}} \mapsto \vec{a}\vec{b}\vec{d}\vec{f}](sub_q^\tau(j_{\vec{a}}))$.

Hence, it must be the case that $\vec{e} \neq \vec{f}$ and that $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e}) \in J_{\vec{a}}$ was included in $N(i)$ during step (I3). So $R_{j_{\vec{a}}}(\underline{\vec{d}}, \vec{e}) \in J$, a contradiction. We conclude by contradiction that $J \nvDash \mathrm{id}[\vec{x}_i \mapsto \vec{a}](sub_q^\tau(i))$. Consequently, $N(i)$ satisfies property (P2b) for any internal vertex i. □

Appendix E. Proof of Theorem 5

Proof of Theorem 5 (*First item*). Let I be a database. Let $A = \{x \mid \exists y(R(x,y) \in I)\}$. For each $x \in A$, we define $succ(x) = \{y \mid R(x,y) \in I\}$. Construct a maximal sequence $C_0 \subsetneq C_1 \subsetneq C_2 \subsetneq \cdots$ where $C_0 = \{\}$ and for each $i > 0$, $C_i = C_{i-1} \cup \{x\}$ for some $x \in A$ such that $c \in succ(x) \subseteq C_{i-1} \cup \{c\}$. Thus, $x \in C_i$ implies $R(\underline{x}, c) \in I$ $(i \geqslant 0)$. Let C_m be the last element in this sequence. Obviously, $m \leqslant |I|$ and C_m can be computed in polynomial time.

We show that

$I \vDash_{\mathsf{sure}} q$ if and only if for some $x \in A$, $\quad succ(x) \subseteq C_m$.

Note that for $x \in A$, $succ(x)$ is nonempty, hence $succ(x) \subseteq C_m$ implies $m \geqslant 1$.

Example 14. For the database $I = \{R(\underline{a}, b), R(\underline{a}, d), R(\underline{b}, c), R(\underline{b}, d), R(\underline{b}, e), R(\underline{d}, c), R(\underline{d}, e), R(\underline{e}, c)\}$, we have $A = \{a, b, d, e\}$. We obtain $C_0 = \{\}$, $C_1 = \{e\}$, $C_2 = \{d, e\}$, $C_3 = \{b, d, e\}$, and this sequence is maximal (i.e. $m = 3$). Since $succ(a) = \{b, d\} \subseteq C_3$, $I \vDash_{\mathsf{sure}} q$.

$\boxed{\Rightarrow}$ Assume for all $x \in A$, $succ(x) \nsubseteq C_m$. Construct a repair J of I as follows:

- for each $x \in C_m$, $R(\underline{x}, c) \in J$; and
- for each $x \in A \setminus C_m$, J contains $R(\underline{x}, y) \in I$ for some y with $y \notin C_m \cup \{c\}$. Assume that no such y exists. Then, for each $x \in A \setminus C_m$, $succ(x) \subseteq C_m \cup \{c\}$. Since $succ(x) \nsubseteq C_m$ by the premise, $c \in succ(x)$. But then the sequence can be extended with $C_{m+1} = C_m \cup \{x\}$, contradicting maximality of the original sequence.

We show $J \nvDash q$. Assume, on the contrary, $J \vDash q$. Then, we can assume a valuation θ such that $R(\theta(x), \theta(y)), R(\theta(y), c) \in J$. From $R(\theta(y), c) \in J$, it follows $\theta(y) \in C_m$. From $R(\theta(x), \theta(y)) \in J$ and $\theta(y) \in C_m$, it follows $\theta(y) = c$. Our construction obviously guarantees that for every a in some C_i, $succ(a) \subseteq C_i \cup \{c\}$. Then, since $c \in C_m$, $succ(c) \subseteq C_m \cup$

$\{c\} = C_m$, contradicting our assumption that $succ(x) \notin C_m$ for each $x \in A$. We conclude by contradiction $J \not\models q$.

Since $J \not\models q$, $I \not\models_{\mathsf{sure}} q$

$\Leftarrow$ Assume the existence of $a \in A$ such that $succ(a) \subseteq C_m$.

Let $M = \{R(\underline{x}, y) \in I \mid x \in C_m\}$. We will show in the next paragraph that for every repair J of M, either $J \models q$ or $J = \{R(\underline{x}, c) \in I \mid x \in C_m\}$. Then, since for every repair K of I, $K \cap M$ is a repair of M, two cases can occur:

1. $K \cap M \models q$. Obviously, $K \models q$.
2. $K \cap M = \{R(\underline{x}, c) \in I \mid x \in C_m\}$. Then, $K \models R(\underline{a}, y), R(\underline{y}, c)$.

It follows $I \models_{\mathsf{sure}} q$.

It remains to be shown that for every repair J of M, either $J \models q$ or $J = \{R(\underline{x}, c) \in I \mid x \in C_m\}$. For each $i \in \{0, \ldots, m\}$, let $M_i = \{R(\underline{x}, y) \in I \mid x \in C_i\}$. We show that for every repair J of M_i, either $J \models q$ or $J = \{R(\underline{x}, c) \in I \mid x \in C_i\}$. Then, the desired result follows by choosing $i = m$. The proof runs by induction on increasing i. The base case $i = 0$ is obvious. For the induction step, assume $C_{k+1} = C_k \cup \{b\}$. Assume w.l.o.g. $M_{k+1} = M_k \cup \{R(\underline{b}, c), R(\underline{b}, a_1), \ldots, R(\underline{b}, a_n)\}$ where $a_1, \ldots, a_n \in C_k$. Let J be a repair of M_{k+1}. Clearly, for some repair J' of M_k, for some $e \in \{c, a_1, \ldots, a_n\}$, $J = J' \cup \{R(\underline{b}, e)\}$. By the induction hypothesis, two cases can occur:

1. $J' \models q$. It follows $J \models q$.
2. $J' = \{R(\underline{x}, c) \in I \mid x \in C_k\}$. If $e = c$, then $J = \{R(\underline{x}, c) \in I \mid x \in C_{k+1}\}$. If $e = a_i \in C_k$ for some $i \in \{1, \ldots, n\}$, then $R(\underline{e}, c), R(\underline{b}, e) \in J$, hence $J \models q$. $\square$

Proof of Theorem 5 (*Second item*). The proof is based on an Ehrenfeucht–Fraïssé game [19]. Assume a vocabulary with constant symbols c, α, β, ε, δ, χ, τ. Suppose there is a FO sentence ψ checking membership of CQA(q). Let d be the quantifier depth of ψ. We exhibit two databases $\mathfrak{A}$ and $\mathfrak{B}$ that are undistinguishable by Ehrenfeucht–Fraïssé games of length d such that $\mathfrak{A} \models_{\mathsf{sure}} q$ and $\mathfrak{B} \not\models_{\mathsf{sure}} q$. Consequently, $\mathfrak{A}$ and $\mathfrak{B}$ are undistinguishable using sentences of quantifier depth d, a contradiction.

The directed graphs in Fig. 13 show the databases $\mathfrak{A}$ and $\mathfrak{B}$. An edge from a_1 to a_2 means that the atom $R(\underline{a_1}, a_2)$ is in the database; the interpretation of constant symbols is indicated in the graphs.

Both databases $\mathfrak{A}$ and $\mathfrak{B}$ contain a long path from β to α, denoted $[\beta, \alpha]_{\mathfrak{A}}$ and $[\beta, \alpha]_{\mathfrak{B}}$, respectively; there is an edge from every element on the path to c. Furthermore, there are edges from δ to χ, and from τ to ε. The difference between both databases is that τ precedes χ on $[\beta, \alpha]_{\mathfrak{A}}$, while τ succeeds χ on $[\beta, \alpha]_{\mathfrak{B}}$.

In every repair, no vertex can have more than one outgoing edge. Fig. 14 shows a repair of $\mathfrak{B}$ that falsifies q, because no path of length 2 ends in c. On the other hand, it can be verified that every repair of $\mathfrak{A}$ has a path of length 2 ending in c.

Distances on the path $[\beta, \alpha]_{\mathfrak{A}}$ are defined as usual: for all vertices v, w on $[\beta, \alpha]_{\mathfrak{A}}$ such that v precedes w on $[\beta, \alpha]_{\mathfrak{A}}$, we write $d_{\mathfrak{A}}(v, w) = p$ if the directed subpath from v to w contains exactly p edges. The distance function $d_{\mathfrak{B}}$ between vertices on $[\beta, \alpha]_{\mathfrak{B}}$ is defined analogously.

We assume that $\mathfrak{A}$ and $\mathfrak{B}$ are chosen sufficiently large such that:

$$d_{\mathfrak{A}}(\beta, \tau) > 3^d, \quad d_{\mathfrak{B}}(\beta, \chi) > 3^d,$$
$$d_{\mathfrak{A}}(\tau, \chi) > 3^d, \quad d_{\mathfrak{B}}(\chi, \tau) > 3^d,$$
$$d_{\mathfrak{A}}(\chi, \alpha) > 3^d, \quad d_{\mathfrak{B}}(\tau, \alpha) > 3^d.$$

We specify the winning strategy for the duplicator. In particular, we show that the duplicator can play in such a way such that the following holds for each round i ($i \geqslant 0$).

Let

$$\vec{a} = (a_{-6}, a_{-5}, \ldots, a_0, a_1, \ldots, a_i)$$

and

$$\vec{b} = (b_{-6}, b_{-5}, \ldots, b_0, b_1, \ldots, b_i),$$

where

$$a_{-6} = b_{-6} = c, \quad a_{-5} = b_{-5} = \alpha, \quad a_{-4} = b_{-4} = \beta,$$
$$a_{-3} = b_{-3} = \delta, \quad a_{-2} = b_{-2} = \varepsilon, \quad a_{-1} = b_{-1} = \chi,$$
$$a_0 = b_0 = \tau,$$

and $a_1, \ldots, a_i$ are the i moves in $\mathfrak{A}$, and $b_1, \ldots, b_i$ are the i moves in $\mathfrak{B}$. Then, for $-6 \leqslant j, l \leqslant i$,

(C1) $a_j = a_l \iff b_j = b_l$.
(C2) for each a_j on $[\beta, \alpha]_{\mathfrak{A}}$ with $a_j \neq \alpha$,

$$\begin{array}{ll} \text{(a)} & d_{\mathfrak{A}}(a_j, succ^i_{\mathfrak{A}}(a_j)) > 3^{d-i} \\ & \Downarrow \\ & d_{\mathfrak{B}}(b_j, succ^i_{\mathfrak{B}}(b_j)) > 3^{d-i} \\ \text{(b)} & succ^i_{\mathfrak{A}}(a_j) = a_l \\ & \text{and } d_{\mathfrak{A}}(a_j, a_l) \leqslant 3^{d-i} \\ & \Downarrow \\ & succ^i_{\mathfrak{B}}(b_j) = b_l \\ & \text{and } d_{\mathfrak{B}}(b_j, b_l) = d_{\mathfrak{A}}(a_j, a_l) \end{array} \tag{11}$$

where for a_j on $[\beta, \alpha]_{\mathfrak{A}}$ such that $a_j \neq \alpha$, $succ^i_{\mathfrak{A}}(a_j)$ denotes the vertex among $\{a_{-6}, \ldots, a_i\}$ that is the nearest successor of a_j on the path $[\beta, \alpha]_{\mathfrak{A}}$. That is, $succ^i_{\mathfrak{A}}(a_j) = a_l$ if

1. $[\beta, \alpha]_{\mathfrak{A}}$ contains a subpath of length $\geqslant 1$ from a_j to a_l; and
2. whenever $-6 \leqslant m \leqslant i$ and a_m lies on the subpath from a_j to a_l, then either $a_m = a_j$ or $a_m = a_l$.

Since α is the last vertex on the path $[\beta, \alpha]_{\mathfrak{A}}$, the successor function $succ^i_{\mathfrak{A}}$ is not defined in α. The successor function $succ^i_{\mathfrak{B}}$ on vertices in $\mathfrak{B}$ is defined analogously. For example, $succ^0_{\mathfrak{A}}(\chi) = \alpha$ and $succ^0_{\mathfrak{B}}(\chi) = \tau$.

The predecessor function $prec^i_{\mathfrak{A}}$ (and $prec^i_{\mathfrak{B}}$) is defined symmetrically: for a_j on $[\beta, \alpha]_{\mathfrak{A}}$ such that $a_j \neq \beta$, we write $prec^i_{\mathfrak{A}}(a_j) = a_l$ if and only if $succ^i_{\mathfrak{A}}(a_l) = a_j$.

We indicate that in condition (C2) of (11), $succ^i_{\mathfrak{B}}(b_j)$ is well defined whenever $succ^i_{\mathfrak{A}}(a_j)$ is well defined. Assume a_j is on $[\beta,\alpha]_{\mathfrak{A}}$ and $a_j \neq \alpha$. Then, $a_j \neq \delta$, $a_j \neq \varepsilon$, and $a_j \neq c$. By condition (C1) of (11), $b_j \neq \alpha$, $b_j \neq \delta$, $b_j \neq \varepsilon$, and $b_j \neq c$. Since b_j is on $[\beta,\alpha]_{\mathfrak{B}}$ with $b_j \neq \alpha$, $succ^i_{\mathfrak{B}}(b_j)$ is well defined.

Propositions 3 and 4 give equivalent ways of expressing (11). These symmetries will be exploited for shortening the proof.

Proposition 3. *Condition* (11) *is equivalent to*: *for* $-6 \leqslant j, l \leqslant i$,

(C1′) $a_j = a_l \iff b_j = b_l$.
(C2′) *for each* a_j *on* $[\beta,\alpha]_{\mathfrak{A}}$ *with* $a_j \neq \beta$,
- (a) *if* $d_{\mathfrak{A}}(prec^i_{\mathfrak{A}}(a_j), a_j) > 3^{d-i}$, *then* $d_{\mathfrak{B}}(prec^i_{\mathfrak{B}}(b_j), b_j) > 3^{d-i}$.
- (b) *if* $prec^i_{\mathfrak{A}}(a_j) = a_l$ *and* $d_{\mathfrak{A}}(a_l, a_j) \leqslant 3^{d-i}$, *then* $prec^i_{\mathfrak{B}}(b_j) = b_l$ *and* $d_{\mathfrak{B}}(b_l, b_j) = d_{\mathfrak{A}}(a_l, a_j)$.

Proof. We show that (11) implies (C1′) and (C2′) (the opposite implication is symmetrical). (11) implies (C1′) is trivial. We next show (11) implies (C2′). Let a_j on $[\beta,\alpha]_{\mathfrak{A}}$ such that $a_j \neq \beta$.

(11) ⇒(C2′a) Assume $d_{\mathfrak{A}}(prec^i_{\mathfrak{A}}(a_j), a_j) > 3^{d-i}$. Let $prec^i_{\mathfrak{B}}(b_j) = b_l$ and $succ^i_{\mathfrak{A}}(a_l) = a_m$. Thus, $succ^i_{\mathfrak{B}}(b_l) = b_j$. Assume $d_{\mathfrak{A}}(a_l, a_m) \leqslant 3^{d-i}$. By condition (C2b) in (11), $b_m = b_j$. By condition (C1) in (11), $a_m = a_j$. Then $succ^i_{\mathfrak{A}}(a_l) = a_j$, hence $prec^i_{\mathfrak{A}}(a_j) = a_l$. Consequently, $d_{\mathfrak{A}}(a_l, a_m) = d_{\mathfrak{A}}(prec^i_{\mathfrak{A}}(a_j), a_j)$, a contradiction.
We conclude by contradiction $d_{\mathfrak{A}}(a_l, a_m) > 3^{d-i}$. By condition (C2a) in (11), $d_{\mathfrak{B}}(b_l, succ^i_{\mathfrak{B}}(b_l)) > 3^{d-i}$. Hence, $d_{\mathfrak{B}}(prec^i_{\mathfrak{B}}(b_j), b_j) > 3^{d-i}$.

(11) ⇒(C2′b) Condition (C2′b) follows immediately from condition (C2b) in (11). □

Proposition 4. *Condition* (11) *is equivalent to*: *for* $-6 \leqslant j, l \leqslant i$,

(C1″) $a_j = a_l \iff b_j = b_l$.
(C2″) *for each* b_j *on* $[\beta,\alpha]_{\mathfrak{B}}$ *with* $b_j \neq \alpha$,
- (a) *if* $d_{\mathfrak{B}}(b_j, succ^i_{\mathfrak{B}}(b_j)) > 3^{d-i}$, *then* $d_{\mathfrak{A}}(a_j, succ^i_{\mathfrak{A}}(a_j)) > 3^{d-i}$.
- (b) *if* $succ^i_{\mathfrak{B}}(b_j) = b_l$ *and* $d_{\mathfrak{B}}(b_j, b_l) \leqslant 3^{d-i}$, *then* $succ^i_{\mathfrak{A}}(a_j) = a_l$ *and* $d_{\mathfrak{A}}(a_j, a_l) = d_{\mathfrak{B}}(b_j, b_l)$.

Proof. We show that (11) implies (C1″) and (C2″) (the opposite implication is symmetrical). (11) implies (C1″) is trivial. We next show (11) implies (C2″). Let b_j on $[\beta,\alpha]_{\mathfrak{B}}$ and $b_j \neq \alpha$.

(11) ⇒(C2″a) Clearly, condition (C2″a) follows from condition (C2b) in (11).

(11) ⇒(C2″b) Let $succ^i_{\mathfrak{B}}(b_j) = b_l$ and $d_{\mathfrak{B}}(b_j, b_l) \leqslant 3^{d-i}$. Let m be an integer satisfying $succ^i_{\mathfrak{A}}(a_j) = a_m$. If $d_{\mathfrak{A}}(a_j, a_m) > 3^{d-i}$, then by condition (C2a) in (11), $d_{\mathfrak{B}}(b_j, b_l) > 3^{d-i}$, a contradiction. We conclude by contradiction that $d_{\mathfrak{A}}(a_j, a_m) \leqslant 3^{d-i}$. Then, by condition (C2b) in (11), $b_l = b_m$ and $d_{\mathfrak{B}}(b_j, b_m) = d_{\mathfrak{A}}(a_j, a_m)$. By condition (C1) in (11), $a_m = a_l$. □

We now show that the conditions in (11) can be preserved by the duplicator. The base case $i = 0$ is immediate from Fig. 13 and our assumption about the size of $\mathfrak{A}$ and $\mathfrak{B}$. For the induction step, assume that the spoiler is making his $(i+1)$st move in $\mathfrak{A}$ (the case $\mathfrak{B}$ is symmetrical because of Proposition 4). If the spoiler plays a_j, $j \leqslant i$, the response of the duplicator is b_j, and the three conditions in (11) are trivially preserved.

Otherwise, the spoiler's move a_{i+1} must fall on $[\beta,\alpha]_{\mathfrak{A}}$ and $\beta \neq a_{i+1} \neq \alpha$. Let $-6 \leqslant p, s \leqslant i$ such that $a_p = prec^{i+1}_{\mathfrak{A}}(a_{i+1})$ and $a_s = succ^{i+1}_{\mathfrak{A}}(a_{i+1})$. Clearly, $succ^i_{\mathfrak{A}}(a_p) = a_s$. Then for $-6 \leqslant j \leqslant i+1$ such that a_j on $[\beta,\alpha]_{\mathfrak{A}}$ with $a_j \neq \alpha$,

$$succ^{i+1}_{\mathfrak{A}}(a_j) = \begin{cases} succ^i_{\mathfrak{A}}(a_j) & \text{if } j \neq p \text{ and } j \neq i+1, \\ a_{i+1} & \text{if } j = p, \\ a_s & \text{if } j = i+1. \end{cases}$$

We consider the case $d_{\mathfrak{A}}(a_p, a_{i+1}) \leqslant d_{\mathfrak{A}}(a_{i+1}, a_s)$ (the case $d_{\mathfrak{A}}(a_p, a_{i+1}) > d_{\mathfrak{A}}(a_{i+1}, a_s)$ is symmetrical because of Proposition 3).

We show hereafter that the duplicator can (and will) choose b_{i+1} such that

$$succ^{i+1}_{\mathfrak{B}}(b_{i+1}) = succ^i_{\mathfrak{B}}(b_p), \tag{12}$$

or equivalently, $b_{i+1} = succ^{i+1}_{\mathfrak{B}}(b_p)$. This is illustrated in Fig. 16. Note that possibly $succ^i_{\mathfrak{B}}(b_p) \neq b_s$.

Hence, for all $-6 \leqslant j \leqslant i+1$ such that b_j on $[\beta,\alpha]_{\mathfrak{B}}$ with $b_j \neq \alpha$,

$$succ^{i+1}_{\mathfrak{B}}(b_j) = \begin{cases} succ^i_{\mathfrak{B}}(b_j) & \text{if } j \neq p \text{ and } j \neq i+1, \\ b_{i+1} & \text{if } j = p, \\ succ^i_{\mathfrak{B}}(b_p) & \text{if } j = i+1. \end{cases}$$

We show that condition (C2) of (11) is satisfied for every a_j with $j \neq p$ and $j \neq i+1$. To this end, let $-6 \leqslant j \leqslant i+1$ such that $j \neq p$ and $j \neq i+1$. Then, $succ^{i+1}_{\mathfrak{A}}(a_j) = succ^i_{\mathfrak{A}}(a_j)$ and $succ^{i+1}_{\mathfrak{B}}(b_j) = succ^i_{\mathfrak{B}}(b_j)$.

1. To show that condition (C2a) in (11) is true for a_j, assume $d_{\mathfrak{A}}(a_j, succ^{i+1}_{\mathfrak{A}}(a_j)) > 3^{d-(i+1)}$.
   Two cases can occur:
   - (a) $d_{\mathfrak{A}}(a_j, succ^i_{\mathfrak{A}}(a_j)) > 3^{d-i}$. By the induction hypothesis $d_{\mathfrak{B}}(b_j, succ^i_{\mathfrak{B}}(b_j)) > 3^{d-i}$, hence $d_{\mathfrak{B}}(b_j, succ^{i+1}_{\mathfrak{B}}(b_j)) > 3^{d-(i+1)}$.
   - (b) $d_{\mathfrak{A}}(a_j, succ^i_{\mathfrak{A}}(a_j)) \leqslant 3^{d-i}$. Let $succ^i_{\mathfrak{A}}(a_j) = a_l$. By the induction hypothesis, $succ^i_{\mathfrak{B}}(b_j) = b_l$ and $d_{\mathfrak{B}}(b_j, b_l) = d_{\mathfrak{A}}(a_j, a_l)$. Hence, $d_{\mathfrak{B}}(b_j, succ^i_{\mathfrak{B}}(b_j)) = d_{\mathfrak{A}}(a_j, a_l)$. Since $d_{\mathfrak{A}}(a_j, a_l) > 3^{d-(i+1)}$ by our initial assumption, it follows $d_{\mathfrak{B}}(b_j, succ^{i+1}_{\mathfrak{B}}(b_j)) > 3^{d-(i+1)}$.

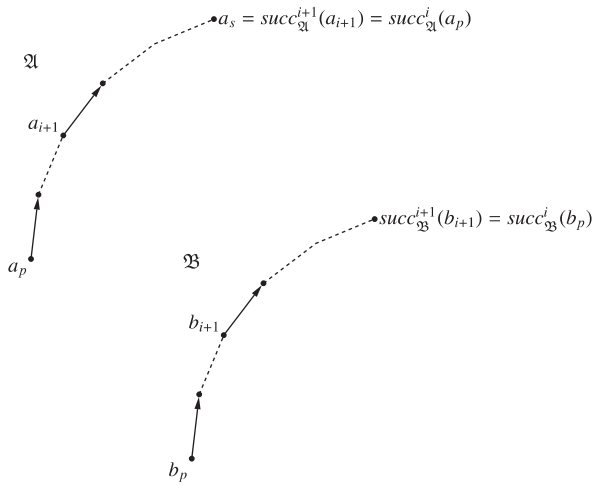

Fig. 16. Moves in round $i+1$.

2. To show that condition (C2b) in (11) is true for a_j, assume $succ^{i+1}_{\mathfrak{A}}(a_j) = a_l$ and $d_{\mathfrak{A}}(a_j, a_l) \leqslant 3^{d-(i+1)}$. Hence, $succ^{i}_{\mathfrak{A}}(a_j) = a_l$ and $d_{\mathfrak{A}}(a_j, a_l) \leqslant 3^{d-i}$. By the induction hypothesis, $succ^{i}_{\mathfrak{B}}(b_j) = b_l$ and $d_{\mathfrak{B}}(b_j, b_l) = d_{\mathfrak{A}}(a_j, a_l)$. It follows $succ^{i+1}_{\mathfrak{B}}(b_j) = b_l$ and $d_{\mathfrak{B}}(b_j, b_l) = d_{\mathfrak{A}}(a_j, a_l)$.

Thus, hereafter it suffices to show that the duplicator can play in such a way that condition (C2) of (11) is true for a_p and a_{i+1}. We distinguish two cases:

Case $d_{\mathfrak{A}}(a_p, a_{i+1}) \leqslant 3^{d-(i+1)}$: The duplicator picks in $\mathfrak{B}$ the (unique) element b_{i+1} satisfying $d_{\mathfrak{B}}(b_p, b_{i+1}) = d_{\mathfrak{A}}(a_p, a_{i+1})$. We distinguish again two cases:

1. $d_{\mathfrak{A}}(a_p, a_s) > 3^{d-i}$. Then, by the induction hypothesis, $d_{\mathfrak{B}}(b_p, succ^{i}_{\mathfrak{B}}(b_p)) > 3^{d-i}$. Since $d_{\mathfrak{B}}(b_p, b_{i+1}) \leqslant 3^{d-(i+1)} < 3^{d-i}$, equation (12) is true.
   It is immediate that condition (C2) of (11) is satisfied for the vertex a_p. We show that condition (C2) of (11) is also satisfied for a_{i+1}:
   (a) Assume $d_{\mathfrak{A}}(a_{i+1}, a_s) > 3^{d-(i+1)}$. Since $d_{\mathfrak{B}}(b_p, succ^{i}_{\mathfrak{B}}(b_p)) > 3^{d-i} = 3 * 3^{d-(i+1)}$ and $d_{\mathfrak{B}}(b_p, b_{i+1}) \leqslant 3^{d-(i+1)}$, it is correct to conclude $d_{\mathfrak{B}}(b_{i+1}, succ^{i+1}_{\mathfrak{B}}(b_{i+1})) > 2 * 3^{d-(i+1)}$.
   (b) Assume $d_{\mathfrak{A}}(a_{i+1}, a_s) \leqslant 3^{d-(i+1)}$. Since $d_{\mathfrak{A}}(a_p, a_{i+1}) \leqslant d_{\mathfrak{A}}(a_{i+1}, a_s)$ by our initial assumption, it follows $d_{\mathfrak{A}}(a_p, a_s) \leqslant 2 * 3^{d-(i+1)} < 3^{d-i}$, a contradiction. We conclude by contradiction that this case cannot occur.
2. $d_{\mathfrak{A}}(a_p, a_s) \leqslant 3^{d-i}$. Then, $succ^{i}_{\mathfrak{B}}(b_p) = b_s$ and $d_{\mathfrak{B}}(b_p, b_s) = d_{\mathfrak{A}}(a_p, a_s)$ by the induction hypothesis. Then, the path from a_{i+1} to a_s is isomorphic to the path from b_{i+1} to $b_s = succ^{i+1}_{\mathfrak{B}}(b_{i+1})$. Condition (12) is obviously satisfied. It is immediate that condition (C2) of (11) is satisfied for the vertices a_p and a_{i+1}.

Case $d_{\mathfrak{A}}(a_p, a_{i+1}) > 3^{d-(i+1)}$: We first show

$$d_{\mathfrak{B}}(b_p, succ^{i}_{\mathfrak{B}}(b_p)) > 2 * 3^{d-(i+1)} + 1.$$

1. If $d_{\mathfrak{A}}(a_p, a_s) > 3^{d-i}$, then, by the induction hypothesis, $d_{\mathfrak{B}}(b_p, succ^{i}_{\mathfrak{B}}(b_p)) > 3^{d-i}$. The desired result follows because $3^{d-i} \geqslant 2 * 3^{d-(i+1)} + 1$.
2. If $d_{\mathfrak{A}}(a_p, a_s) \leqslant 3^{d-i}$, then $succ^{i}_{\mathfrak{B}}(b_p) = b_s$ and $d_{\mathfrak{B}}(b_p, b_s) = d_{\mathfrak{A}}(a_p, a_s)$ by the induction hypothesis. By our initial assumptions that $3^{d-(i+1)} < d_{\mathfrak{A}}(a_p, a_{i+1}) \leqslant d_{\mathfrak{A}}(a_{i+1}, a_s)$, it follows $d_{\mathfrak{A}}(a_p, a_s) > 2 * 3^{d-(i+1)} + 1$. Hence, $d_{\mathfrak{B}}(b_p, b_s) > 2 * 3^{d-(i+1)} + 1$.

Thus, the duplicator can choose in $\mathfrak{B}$ an element b_{i+1} satisfying

$$d_{\mathfrak{B}}(b_p, b_{i+1}) > 3^{d-(i+1)},$$
$$d_{\mathfrak{B}}(b_{i+1}, succ^{i+1}_{\mathfrak{B}}(b_{i+1})) > 3^{d-(i+1)},$$

where $succ^{i+1}_{\mathfrak{B}}(b_{i+1}) = succ^{i}_{\mathfrak{B}}(b_p)$. Condition (12) is obviously satisfied. It is immediate that condition (C2) of (11) is satisfied for a_p and a_{i+1}.

Finally, we show that $(\vec{a}, \vec{b})$ defines a partial isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$. That is, for $-6 \leqslant j, l \leqslant i$,

1. $a_j = a_l \iff b_j = b_l$. This is condition (C1) of (11).
2. For every $e \in \{c, \alpha, \beta, \delta, \varepsilon, \chi, \tau\}$, $a_j = e \iff b_j = e$. This is immediate from the choice of $(a_{-6}, \ldots, a_0)$ and $(b_{-6}, \ldots, b_0)$.
3. $R(a_j, a_l) \in \mathfrak{A} \iff R(b_j, b_l) \in \mathfrak{B}$. We show $\Rightarrow$ (the opposite direction is symmetrical using Proposition 4). The implication holds obviously if one of a_j or a_l is not on $[\beta, \alpha]_{\mathfrak{A}}$. Assume next that a_j and a_l are both on $[\beta, \alpha]_{\mathfrak{A}}$. Then, b_j and b_l are both on $[\beta, \alpha]_{\mathfrak{B}}$. Since $d_{\mathfrak{A}}(a_j, a_l) = 1 \leqslant 3^{d-i}$, it follows from condition (C2b) of (11) that $d_{\mathfrak{B}}(b_j, b_l) = 1$, hence $R(b_j, b_l) \in \mathfrak{B}$.

Thus we have shown that the duplicator can win a d-round Ehrenfeucht–Fraïssé game on $\mathfrak{A}$ and $\mathfrak{B}$. □

References

[1] M. Arenas, L.E. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: PODS, ACM Press, New York, 1999, pp. 68–79.
[2] J. Chomicki, Consistent query answering: five easy pieces, in: T. Schwentick, D. Suciu (Eds.), ICDT, Lecture Notes in Computer Science, vol. 4353, Springer, Berlin, 2007, pp. 1–17.
[3] A. Fuxman, R.J. Miller, First-order query rewriting for inconsistent databases, J. Comput. Syst. Sci. 73 (4) (2007) 610–635.
[4] J. Wijsen, On the consistent rewriting of conjunctive queries under primary key constraints, in: M. Arenas, M.I. Schwartzbach (Eds.), DBPL, Lecture Notes in Computer Science, vol. 4797, Springer, Berlin, 2007, pp. 112–126.
[5] C. Beeri, R. Fagin, D. Maier, M. Yannakakis, On the desirability of acyclic database schemes, J. ACM 30 (3) (1983) 479–513.
[6] A. Fuxman, R.J. Miller, Towards inconsistency management in data integration systems, in: S. Kambhampati, C.A. Knoblock (Eds.), IIWeb, 2003, pp. 143–148.
[7] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, Reading, MA, 1995.
[8] J. Lin, A.O. Mendelzon, Merging databases under constraints, Int. J. Cooperative Inf. Syst. 7 (1) (1998) 55–76.
[9] J. Wijsen, Database repairing using updates, ACM Trans. Database Syst. 30 (3) (2005) 722–768.
[10] A. Fuxman, E. Fazli, R.J. Miller, Conquer: efficient management of inconsistent databases, in: F. Özcan (Ed.), SIGMOD Conference, ACM, New York, 2005, pp. 155–166.
[11] L. Grieco, D. Lembo, R. Rosati, M. Ruzzi, Consistent query answering under key and exclusion dependencies: algorithms and

experiments, in: O. Herzog, H.-J. Schek, N. Fuhr, A. Chowdhury, W. Teiken (Eds.), CIKM, ACM, New York, 2005, pp. 792–799.
[12] D. Lembo, R. Rosati, M. Ruzzi, On the first-order reducibility of unions of conjunctive queries over inconsistent databases, in: T. Grust, H. Höpfner, A. Illarramendi, S. Jablonski, M. Mesiti, S. Müller, P.-L. Patranjan, K.-U. Sattler, M. Spiliopoulou, J. Wijsen (Eds.), EDBT Workshops, Lecture Notes in Computer Science, vol. 4254, Springer, Berlin, 2006, pp. 358–374.
[13] A. Fuxman, R.J. Miller, First-order query rewriting for inconsistent databases, in: T. Eiter, L. Libkin (Eds.), ICDT, Lecture Notes in Computer Science, vol. 3363, Springer, Berlin, 2005, pp. 337–351.
[14] A. Calí, D. Lembo, R. Rosati, On the decidability and complexity of query answering over inconsistent and incomplete databases, in: PODS, ACM, New York, 2003, pp. 260–271.
[15] J. Chomicki, J. Marcinkowski, Minimal-change integrity maintenance using tuple deletions, Inf. Comput. 197 (1–2) (2005) 90–121.
[16] G. Gottlob, N. Leone, F. Scarcello, Hypertree decompositions and tractable queries, J. Comput. Syst. Sci. 64 (3) (2002) 579–627.
[17] J. Wijsen, Consistent query answering under primary keys: a characterization of tractable queries, in: R. Fagin (Ed.), ICDT, ACM, New York, 2009.
[18] G. Gottlob, N. Leone, F. Scarcello, Hypertree decompositions and tractable queries, J. Comput. Syst. Sci. 64 (3) (2002) 579–627.
[19] L. Libkin, Elements of Finite Model Theory, Springer, Berlin, 2004.