Paraconsistent Logic and Query Answering in Inconsistent Databases

C. A. Middelburg
ORCID: https://orcid.org/0000-0002-8725-0197

Informatics Institute, Faculty of Science, University of Amsterdam,
Science Park 900, 1098 XH Amsterdam, the Netherlands
C.A.Middelburg@uva.nl

Abstract. This paper concerns the paraconsistent logic $\mathrm{LPQ}^{\supset,\mathsf{F}}$ and an application of it in the area of relational database theory. The notions of a relational database, a query applicable to a relational database, and a consistent answer to a query with respect to a possibly inconsistent relational database are considered from the perspective of this logic. This perspective enables among other things the definition of a consistent answer to a query with respect to a possibly inconsistent database without resort to database repairs. In a previous paper, $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is presented with a sequent-style natural deduction proof system. In this paper, a sequent calculus proof system is presented because it is common to use a sequent calculus proof system as the basis of proof search procedures and such procedures may form the core of algorithms for computing consistent answers to queries.

Keywords: relational database; inconsistent database; consistent query answering; paraconsistent logic; sequent calculus

1 Introduction

In the area of relational database theory, rather often the view is taken in which a database is a theory of first-order classical logic, a query is a formula, and query answering amounts to proving in first-order classical logic that a formula is a logical consequence of a theory. In [21], the term *proof-theoretic view* is introduced for this view and various arguments in favor of this view are given. In work on query answering in inconsistent databases based on this view, resort to (consistent) repairs of inconsistent databases is considered unavoidable to come to a notion of a consistent answer to a possibly inconsistent database (see e.g. [1]). The reason for this is that in classical logic every formula is a logical consequence of an inconsistent theory.

In [4], the resort to repairs is avoided by switching from first-order classical logic to first-order minimal logic, a logic in which not every formula is a logical consequence of an inconsistent theory. By some shortcomings in [4], there has been no follow-up of this work. The main shortcoming is that a semantics with respect to which the presented proof system is sound and complete is not given.

By that, it remains unclear how the work fits the existing (concrete or abstract) views on what is a database. Actually, there exists a Kripke semantics of the proposional fragment (see e.g. [10]), but that semantics seems difficult to relate the existing views on what is a database.

This paper considers consistent query answering from the perspective of $\mathrm{LPQ}^{\supset,\mathsf{F}}$, another first-order logic in which not every formula is a logical consequence of an inconsistent theory. A sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ and a three-valued semantics with respect to which the proof system is sound and complete are given. The notions of a relational database, a query applicable to a relational database, and an answer to a query with respect to a relational database are defined in the setting of $\mathrm{LPQ}^{\supset,\mathsf{F}}$. The definitions concerned are based on those given in [21]. Two notions of a consistent answer to a query with respect to a possibly inconsistent relational database are introduced. One of them is reminiscent of the notion of a consistent answer from [4] and the other is essentially the same as the notion of a consistent answer from [1].

Proof search procedures may form the core of algorithms for computing consistent answers to queries. It is common to use a sequent calculus proof system as the basis of proof search procedures. That is why a sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is presented in this paper. The proof system of first-order minimal logic presented in [4] is a natural deduction proof system. A natural deduction proof system can also be used as the basis of a proof search procedure, but it is not so widely known how this can be done. The lack of any remark about a proof search procedure for first-order minimal logic is sometimes considered a shortcoming in [4] as well.

A logic is called a paraconsistent logic if in the logic not every formula is a logical consequence of an inconsistent theory. In [20], Priest proposed the paraconsistent propositional logic LP (Logic of Paradox) and its first-order extension LPQ. $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is LPQ enriched with a falsity constant and an implication connective for which the standard deduction theorem holds. $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is essentially the same as $\mathrm{J}_3^*{=}$ [11] and LP° [19]. In [15], a sequent-style natural deduction proof system for $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is presented. Several main properties of the logical consequence relation and the logical equivalence relation of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ are also treated in that paper.

In $\mathrm{LPQ}^{\supset,\mathsf{F}}$, for every inconsistent theory Γ in which the falsity constant F does not occur, for every formula A that does not have function symbols, predicate symbols or free variables in common with Γ, A is a logical consequence of Γ only if A is a logical consequence of the empty theory. In minimal logic, for every inconsistent theory Γ, for every formula A, $\neg A$ is a logical consequence of Γ. Therefore, $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is considered a genuine paraconsistent logic and minimal logic is not considered a genuine paraconsistent logic (cf. [18]). Moreover, the properties of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ treated in [15] indicate among other things that the logical consequence relation and the logical equivalence relation of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ are very close to those of classical logic. That is why the choice has been made to consider in this paper query answering in inconsistent databases from the perspective of $\mathrm{LPQ}^{\supset,\mathsf{F}}$.

The structure of this paper is as follows. First, the language of $\mathrm{LPQ}^{\supset,\mathsf{F}}$, a sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$, and a three-valued semantics of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ are presented (Sections 2, 3, and 4). Next, relational databases and query answering in possibly inconsistent relational databases are considered from the perspective of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ (Sections 5 and 6). After that, examples of query answering are given (Section 7) and some remaining remarks about consistent query answering are made (Section 8). Finally, some concluding remarks are made (Section 9).

In order to make this paper self-contained, large parts of Sections 2 and 4 have been copied near verbatim or slightly modified from [15].

2 The Language of $\mathrm{LPQ}^{\supset,\mathsf{F}}$

In this section the language of the paraconsistent logic $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is described. First the notion of a signature is introduced and then the terms and formulas of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ are defined for a fixed but arbitrary signature. Moreover, some relevant notational conventions and abbreviations are presented and some remarks about free variables and substitution are made. In coming sections, the proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ and the interpretation of the terms and formulas of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ are defined for a fixed but arbitrary signature.

Signatures It is assumed that the following has been given: (a) a countably infinite set $\mathcal{V}$ of *variables*, (b) for each $n \in \mathbb{N}$, a countably infinite set F_n of *function symbols of arity* n, and, (c) for each $n \in \mathbb{N}$, a countably infinite set P_n of *predicate symbols of arity* n. It is also assumed that all these sets and the set $\{=\}$ are mutually disjoint. We write SYM for the set $\mathcal{V} \cup \bigcup\{F_n \mid n \in \mathbb{N}\} \cup \bigcup\{P_n \mid n \in \mathbb{N}\}$.

Function symbols of arity 0 are also known as *constant symbols* and predicate symbols of arity 0 are also known as *proposition symbols*.

A *signature* Σ is a subset of $SYM \setminus \mathcal{V}$. We write $\mathrm{F}_n(\Sigma)$ and $\mathrm{P}_n(\Sigma)$, where Σ is a signature and $n \in \mathbb{N}$, for the sets $\Sigma \cap F_n$ and $\Sigma \cap P_n$, respectively.

The language of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ will be defined for a fixed but arbitrary signature Σ. This language will be called the language of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ over Σ or shortly the language of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$. The corresponding proof system and interpretation will be called the proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ and the interpretation of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$.

Terms and formulas The language of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ consists of terms and formulas. They are constructed according to the formation rules given below.

The set of all *terms of* $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$, written $\mathcal{T}(\Sigma)$, is inductively defined by the following formation rules:

1. if $x \in \mathcal{V}$, then $x \in \mathcal{T}(\Sigma)$;
2. if $c \in \mathrm{F}_0(\Sigma)$, then $c \in \mathcal{T}(\Sigma)$;
3. if $f \in \mathrm{F}_{n+1}(\Sigma)$ and $t_1, \ldots, t_{n+1} \in \mathcal{T}(\Sigma)$, then $f(t_1, \ldots, t_{n+1}) \in \mathcal{T}(\Sigma)$.

The set of all *closed terms of* $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ is the subset of $\mathcal{F}(\Sigma)$ inductively defined by the formation rules 2 and 3.

The set of all *formulas of* $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$, written $\mathcal{F}(\Sigma)$, is inductively defined by the following formation rules:

1. $\mathsf{F} \in \mathcal{F}(\Sigma)$;
2. if $p \in \mathrm{P}_0(\Sigma)$, then $p \in \mathcal{F}(\Sigma)$;
3. if $P \in \mathrm{P}_{n+1}(\Sigma)$ and $t_1, \ldots, t_{n+1} \in \mathcal{T}(\Sigma)$, then $P(t_1, \ldots, t_{n+1}) \in \mathcal{F}(\Sigma)$;
4. if $t_1, t_2 \in \mathcal{T}(\Sigma)$, then $t_1 = t_2 \in \mathcal{F}(\Sigma)$;
5. if $A \in \mathcal{F}(\Sigma)$, then $\neg A \in \mathcal{F}(\Sigma)$;
6. if $A_1, A_2 \in \mathcal{F}(\Sigma)$, then $A_1 \wedge A_2$, $A_1 \vee A_2$, $A_1 \supset A_2 \in \mathcal{F}(\Sigma)$;
7. if $x \in \mathcal{V}$ and $A \in \mathcal{F}(\Sigma)$, then $\forall x \bullet A$, $\exists x \bullet A \in \mathcal{F}(\Sigma)$.

The set of all *atomic formulas of* $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ is the subset of $\mathcal{F}(\Sigma)$ inductively defined by the formation rules 1–4. The set of all *literals of* $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ is the subset of $\mathcal{F}(\Sigma)$ inductively defined by the formation rules 1–5.

For the connectives $\neg$, $\wedge$, $\vee$, and $\supset$ and the quantifiers $\forall$ and $\exists$, the classical truth-conditions and falsehood-conditions are retained. Except for implications, a formula is classified as both-true-and-false exactly when it cannot be classified as true or false by these conditions.

We write $e_1 \equiv e_2$, where e_1 and e_2 are terms from $\mathcal{T}(\Sigma)$ or formulas from $\mathcal{F}(\Sigma)$, to indicate that e_1 is syntactically equal to e_2.

Notational conventions and abbreviations The following will sometimes be used without mentioning (with or without decoration): x as a meta-variable ranging over all variables from $\mathcal{V}$, t as a meta-variable ranging over all terms from $\mathcal{T}(\Sigma)$, A as a meta-variable ranging over all formulas from $\mathcal{F}(\Sigma)$, and Γ as a meta-variable ranging over all finite sets of formulas from $\mathcal{F}(\Sigma)$.

The string representation of terms and formulas suggested by the formation rules given above can lead to syntactic ambiguities. Parentheses are used to avoid such ambiguities. The need to use parentheses is reduced by ranking the precedence of the logical connectives $\neg$, $\wedge$, $\vee$, $\supset$. The enumeration presents this order from the highest precedence to the lowest precedence. Moreover, the scope of the quantifiers extends as far as possible to the right and $\forall x_1 \bullet \cdots \forall x_n \bullet A$ is usually written as $\forall x_1, \ldots, x_n \bullet A$.

The following abbreviation is used: T stands for $\neg\mathsf{F}$.

Free variables and substitution Free variables of a term or formula and substitution for variables in a term or formula are defined in the usual way.

Let x be a variable from $\mathcal{V}$, t be a term from $\mathcal{T}(\Sigma)$, and e be a term from $\mathcal{T}(\Sigma)$ or a formula from $\mathcal{F}(\Sigma)$. Then we write $[x := t]e$ for the result of substituting the term t for the free occurrences of the variable x in e, avoiding (by means of renaming of bound variables) free variables becoming bound in t.

3 A Proof System of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$

In this section, a sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ is presented. This means that the inference rules have sequents as premises and conclusions. First, the notion of a sequent is introduced. Then, the inference rules of the proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ are presented. After that, the notion of a derivation of a sequent from a set of sequents and the notion of a proof of a sequent are introduced. An extension of the proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ which can serve as a proof system for first-order classical logic is also described.

Sequents In $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$, a *sequent* is an expression of the form $\Gamma \Rightarrow \Delta$, where Γ and Δ are finite sets of formulas from $\mathcal{F}(\Sigma)$. We write Γ, Γ' for $\Gamma \cup \Gamma'$ and A, where A is a formula from $\mathcal{F}(\Sigma)$, for $\{A\}$ on both sides of a sequent. Moreover, we write $\Rightarrow \Delta$ instead of $\emptyset \Rightarrow \Delta$.

A sequent $\Gamma \Rightarrow \Delta$ states that the logical consequence relation that is defined in Section 4 holds between Γ and Δ. Informally speaking, that logical consequence relation holds between Γ and Δ if, whenever every formula from Γ is not false, at least one formula from Δ is not false. If a sequent $\Gamma \Rightarrow \Delta$ can be proved by means of the rules of inference given below, then that logical consequence relation holds between Γ and Δ.

Rules of inference The sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ consists of the inference rules given in Table 1. In this table, x and y are meta-variables ranging over all variables from $\mathcal{V}$, t, t_1, and t_2 are meta-variables ranging over all terms from $\mathcal{T}(\Sigma)$, A, A_1, and A_2 are meta-variables ranging over all formulas from $\mathcal{F}(\Sigma)$, and Γ and Δ are meta-variables ranging over all finite sets of formulas from $\mathcal{F}(\Sigma)$.

Derivations and proofs In $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$, a *derivation of a sequent* $\Gamma \Rightarrow \Delta$ *from a finite set of sequents* $\mathcal{H}$ is a finite sequence $\langle s_1, \ldots, s_n \rangle$ of sequents such that s_n equals $\Gamma \Rightarrow \Delta$ and, for each $i \in \{1, \ldots, n\}$, one of the following conditions holds:

- $s_i \in \mathcal{H}$;
- s_i is the conclusion of an instance of some inference rule from the proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ whose premises are among $s_1, \ldots, s_{i-1}$.

A *proof of a sequent* $\Gamma \Rightarrow \Delta$ is a derivation of $\Gamma \Rightarrow \Delta$ from the empty set of sequents. A sequent $\Gamma \Rightarrow \Delta$ is said to be *provable* if there exists a proof of $\Gamma \Rightarrow \Delta$.

Let Γ and Δ be sets of formulas from $\mathcal{F}(\Sigma)$. Then Δ is *derivable* from Γ, written $\Gamma \vdash \Delta$, iff there exist finite sets $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ such that the sequent $\Gamma' \Rightarrow \Delta'$ is provable.

An inference rule that does not belong to the inference rules of some proof system is called a *derived inference rule* if there exists a derivation of the conclu-

Table 1. Sequent calculus proof system of LPQ$^{\supset,\mathsf{F}}(\Sigma)$

$$\boxed{\mathrm{I}}\ \frac{}{A, \Gamma \Rightarrow \Delta, A}\ *$$

$$\boxed{\mathsf{F}\text{-L}}\ \frac{}{\mathsf{F}, \Gamma \Rightarrow \Delta}$$

$$\boxed{\neg\text{-R}}\ \frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}\ \dagger$$

$$\boxed{\wedge\text{-L}}\ \frac{A_1, A_2, \Gamma \Rightarrow \Delta}{A_1 \wedge A_2, \Gamma \Rightarrow \Delta}$$

$$\boxed{\wedge\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, A_1 \quad \Gamma \Rightarrow \Delta, A_2}{\Gamma \Rightarrow \Delta, A_1 \wedge A_2}$$

$$\boxed{\vee\text{-L}}\ \frac{A_1, \Gamma \Rightarrow \Delta \quad A_2, \Gamma \Rightarrow \Delta}{A_1 \vee A_2, \Gamma \Rightarrow \Delta}$$

$$\boxed{\vee\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, A_1, A_2}{\Gamma \Rightarrow \Delta, A_1 \vee A_2}$$

$$\boxed{\supset\text{-L}}\ \frac{\Gamma \Rightarrow \Delta, A_1 \quad A_2, \Gamma \Rightarrow \Delta}{A_1 \supset A_2, \Gamma \Rightarrow \Delta}$$

$$\boxed{\supset\text{-R}}\ \frac{A_1, \Gamma \Rightarrow \Delta, A_2}{\Gamma \Rightarrow \Delta, A_1 \supset A_2}$$

$$\boxed{\forall\text{-L}}\ \frac{[x := t]A, \Gamma \Rightarrow \Delta}{\forall x \bullet A, \Gamma \Rightarrow \Delta}$$

$$\boxed{\forall\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, [x := y]A}{\Gamma \Rightarrow \Delta, \forall x \bullet A}\ \ddagger$$

$$\boxed{\exists\text{-L}}\ \frac{[x := y]A, \Gamma \Rightarrow \Delta}{\exists x \bullet A, \Gamma \Rightarrow \Delta}\ \ddagger$$

$$\boxed{\exists\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, [x := t]A}{\Gamma \Rightarrow \Delta, \exists x \bullet A}$$

$$\boxed{\neg\mathsf{F}\text{-R}}\ \frac{}{\Gamma \Rightarrow \Delta, \neg\mathsf{F}}$$

$$\boxed{\neg\neg\text{-L}}\ \frac{A, \Gamma \Rightarrow \Delta}{\neg\neg A, \Gamma \Rightarrow \Delta}$$

$$\boxed{\neg\neg\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \neg\neg A}$$

$$\boxed{\neg\wedge\text{-L}}\ \frac{\neg A_1, \Gamma \Rightarrow \Delta \quad \neg A_2, \Gamma \Rightarrow \Delta}{\neg(A_1 \wedge A_2), \Gamma \Rightarrow \Delta}$$

$$\boxed{\neg\wedge\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, \neg A_1, \neg A_2}{\Gamma \Rightarrow \Delta, \neg(A_1 \wedge A_2)}$$

$$\boxed{\neg\vee\text{-L}}\ \frac{\neg A_1, \neg A_2, \Gamma \Rightarrow \Delta}{\neg(A_1 \vee A_2), \Gamma \Rightarrow \Delta}$$

$$\boxed{\neg\vee\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, \neg A_1 \quad \Gamma \Rightarrow \Delta, \neg A_2}{\Gamma \Rightarrow \Delta, \neg(A_1 \vee A_2)}$$

$$\boxed{\neg\supset\text{-L}}\ \frac{A_1, \neg A_2, \Gamma \Rightarrow \Delta}{\neg(A_1 \supset A_2), \Gamma \Rightarrow \Delta}$$

$$\boxed{\neg\supset\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, A_1 \quad \Gamma \Rightarrow \Delta, \neg A_2}{\Gamma \Rightarrow \Delta, \neg(A_1 \supset A_2)}$$

$$\boxed{\neg\forall\text{-L}}\ \frac{\neg[x := y]A, \Gamma \Rightarrow \Delta}{\neg\forall x \bullet A, \Gamma \Rightarrow \Delta}\ \ddagger$$

$$\boxed{\neg\forall\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, \neg[x := t]A}{\Gamma \Rightarrow \Delta, \neg\forall x \bullet A}$$

$$\boxed{\neg\exists\text{-L}}\ \frac{\neg[x := t]A, \Gamma \Rightarrow \Delta}{\neg\exists x \bullet A, \Gamma \Rightarrow \Delta}$$

$$\boxed{\neg\exists\text{-R}}\ \frac{\Gamma \Rightarrow \Delta, \neg[x := y]A}{\Gamma \Rightarrow \Delta, \neg\exists x \bullet A}\ \ddagger$$

$$\boxed{=\text{-Refl}}\ \frac{t = t, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

$$\boxed{=\text{-Repl}}\ \frac{[x := t_1]A, \Gamma \Rightarrow \Delta}{t_1 = t_2, [x := t_2]A, \Gamma \Rightarrow \Delta}\ *$$

$*$ restriction: A is a literal.

$\dagger$ restriction: A is an atomic formula.

$\ddagger$ restriction: y is not free in Γ, y is not free in Δ, y is not free in A unless $x \equiv y$.

sion from the premises, using the inference rules of that proof system, for each instance of the rule.

Let the set $\Gamma_{=}$ of *equality axioms* be the subset of $\mathcal{F}(\Sigma)$ consisting of the following formulas:

- $\forall x \bullet x = x$;
- $c = c$ for every $c \in \mathrm{F}_0(\Sigma)$;
- $\forall x_1, y_1, \ldots, x_{n+1}, y_{n+1} \bullet$
  $$x_1 = y_1 \wedge \ldots \wedge x_{n+1} = y_{n+1} \supset f(x_1, \ldots, x_{n+1}) = f(y_1, \ldots, y_{n+1})$$
  for every $f \in \mathrm{F}_{n+1}(\Sigma)$, for every $n \in \mathbb{N}$;
- $p \supset p$ for every $p \in \mathrm{P}_0(\Sigma)$;
- $\forall x_1, y_1, \ldots, x_{n+1}, y_{n+1} \bullet$
  $$x_1 = y_1 \wedge \ldots \wedge x_{n+1} = y_{n+1} \wedge P(x_1, \ldots, x_{n+1}) \supset P(y_1, \ldots, y_{n+1})$$
  for every $P \in \mathrm{P}_{n+1}(\Sigma)$, for every $n \in \mathbb{N}$.

Then the sequent $\Gamma \Rightarrow \Delta$ is provable iff $\Gamma_{=}, \Gamma \Rightarrow \Delta$ is provable without using the inference rules =-Refl and =-Repl. This can easily be proved in the same way as Proposition 7.4 from [22] is proved.

In [15], a proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ formulated as a sequent-style natural deduction system is given.

A proof system of CL(Σ) We use the name CL here to denote a version of classical logic that has the same logical constants, connectives, and quantifiers as $\mathrm{LPQ}^{\supset,\mathsf{F}}$.

In CL, the same assumptions about symbols are made as in $\mathrm{LPQ}^{\supset,\mathsf{F}}$ and the notion of a signature is defined as in $\mathrm{LPQ}^{\supset,\mathsf{F}}$. The languages of $\mathrm{CL}(\Sigma)$ and $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ are the same. A sound and complete sequent calculus proof system of $\mathrm{CL}(\Sigma)$ can be obtained by adding the following inference rule to the sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$:[1]

$$\boxed{\neg\text{-L}}\ \frac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta}$$

4 Truth and Logical Consequence in $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$

The proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ is based on the logical consequence relation on sets of formulas of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ defined in this section: a sequent $\Gamma \Rightarrow \Delta$ is provable iff the logical consequence relation holds between Γ and Δ. This relation is defined in terms of the truth value of formulas of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$. The truth value of formulas is defined relative to a structure and an assignment. First, the notion of a structure and the notion of an assignment are introduced. Next, the truth value of formulas and the logical consequence relation on sets of formulas are defined.

[1] If we replace the inference rule ¬-R by the inference rule ¬-L in the sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$, then we obtain a sound and complete proof system of the paracomplete analogue of $\mathrm{LPQ}^{\supset,\mathsf{F}}$. The propositional part of that logic ($\mathrm{K3}^{\supset,\mathsf{F}}$) is studied in e.g. [16].

Structures The terms from $\mathcal{T}(\Sigma)$ and the formulas from $\mathcal{F}(\Sigma)$ are interpreted in structures which consist of a non-empty domain of individuals and an interpretation of every symbol in the signature Σ and the equality symbol. The domain of truth values consists of three values: t (*true*), f (*false*), and b (*both true and false*).

A structure $\mathbf{A}$ of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ consists of:

- a set $\mathcal{U}^{\mathbf{A}}$, the *domain of* $\mathbf{A}$, such that $\mathcal{U}^{\mathbf{A}} \neq \emptyset$ and $\mathcal{U}^{\mathbf{A}} \cap \{\mathsf{t}, \mathsf{f}, \mathsf{b}\} = \emptyset$;
- for each $c \in \mathrm{F}_0(\Sigma)$,
  an element $c^{\mathbf{A}} \in \mathcal{U}^{\mathbf{A}}$;
- for each $n \in \mathbb{N}$, for each $f \in \mathrm{F}_{n+1}(\Sigma)$,
  a function $f^{\mathbf{A}} : {\mathcal{U}^{\mathbf{A}}}^{n+1} \to \mathcal{U}^{\mathbf{A}}$;
- for each $p \in \mathrm{P}_0(\Sigma)$,
  an element $p^{\mathbf{A}} \in \{\mathsf{t}, \mathsf{f}, \mathsf{b}\}$;
- for each $n \in \mathbb{N}$, for each $P \in \mathrm{P}_{n+1}(\Sigma)$,
  a function $P^{\mathbf{A}} : {\mathcal{U}^{\mathbf{A}}}^{n+1} \to \{\mathsf{t}, \mathsf{f}, \mathsf{b}\}$;
- a function $=^{\mathbf{A}} : {\mathcal{U}^{\mathbf{A}}}^2 \to \{\mathsf{t}, \mathsf{f}, \mathsf{b}\}$ such that, for each $d \in \mathcal{U}^{\mathbf{A}}$,
  $=^{\mathbf{A}} (d, d) = \mathsf{t}$ or $=^{\mathbf{A}} (d, d) = \mathsf{b}$.

Instead of $w^{\mathbf{A}}$ we write w when it is clear from the context that the interpretation of symbol w in structure $\mathbf{A}$ is meant.

Assignments An assignment in a structure $\mathbf{A}$ of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ assigns elements from $\mathcal{U}^{\mathbf{A}}$ to the variables from $\mathcal{V}$. The interpretation of the terms from $\mathcal{T}(\Sigma)$ and the formulas from $\mathcal{F}(\Sigma)$ in $\mathbf{A}$ is given with respect to an assignment α in $\mathbf{A}$.

Let $\mathbf{A}$ be a structure of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$. Then an *assignment in* $\mathbf{A}$ is a function $\alpha : \mathcal{V} \to \mathcal{U}^{\mathbf{A}}$. For every assignment α in $\mathbf{A}$, variable $x \in \mathcal{V}$, and element $d \in \mathcal{U}^{\mathbf{A}}$, we write $\alpha(x \to d)$ for the assignment α' in $\mathbf{A}$ such that $\alpha'(x) = d$ and $\alpha'(y) = \alpha(y)$ if $y \not\equiv x$.

Valuations and models The valuation of the terms from $\mathcal{T}(\Sigma)$ is given by a function mapping term t, structure $\mathbf{A}$ and assignment α in $\mathbf{A}$ to the element of $\mathcal{U}^{\mathbf{A}}$ that is the value of t in $\mathbf{A}$ under assignment α. Similarly, the valuation of the formulas from $\mathcal{F}(\Sigma)$ is given by a function mapping formula A, structure $\mathbf{A}$ and assignment α in $\mathbf{A}$ to the element of $\{\mathsf{t}, \mathsf{f}, \mathsf{b}\}$ that is the truth value of A in $\mathbf{A}$ under assignment α. We write $[\![t]\!]^{\mathbf{A}}_{\alpha}$ and $[\![A]\!]^{\mathbf{A}}_{\alpha}$ for these valuations.

The valuation functions for the terms from $\mathcal{T}(\Sigma)$ and the formulas from $\mathcal{F}(\Sigma)$ are inductively defined in Table 2. In this table, x is a meta-variable ranging over all variables from $\mathcal{V}$, c is a meta-variable ranging over all function symbols from $\mathrm{F}_0(\Sigma)$, f is a meta-variable ranging over all function symbols from $\mathrm{F}_{n+1}(\Sigma)$ (where n is understood from the context), p is a meta-variable ranging over all predicate symbols from $\mathrm{P}_0(\Sigma)$, P is a meta-variable ranging over all predicate symbols from $\mathrm{P}_{n+1}(\Sigma)$ (where n is understood from the context), $t_1, \ldots, t_{n+1}$

Table 2. Valuations of terms and formulas of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$

$$
\begin{aligned}
[\![x]\!]^{\mathbf{A}}_{\alpha} &= \alpha(x)\;, \\
[\![c]\!]^{\mathbf{A}}_{\alpha} &= c^{\mathbf{A}}\;, \\
[\![f(t_1,\ldots,t_{n+1})]\!]^{\mathbf{A}}_{\alpha} &= f^{\mathbf{A}}([\![t_1]\!]^{\mathbf{A}}_{\alpha},\ldots,[\![t_{n+1}]\!]^{\mathbf{A}}_{\alpha}) \\
[\![\mathsf{F}]\!]^{\mathbf{A}}_{\alpha} &= \mathsf{f}\;, \\
[\![p]\!]^{\mathbf{A}}_{\alpha} &= p^{\mathbf{A}}\;, \\
[\![P(t_1,\ldots,t_{n+1})]\!]^{\mathbf{A}}_{\alpha} &= P^{\mathbf{A}}([\![t_1]\!]^{\mathbf{A}}_{\alpha},\ldots,[\![t_{n+1}]\!]^{\mathbf{A}}_{\alpha})\;, \\
[\![t_1 = t_2]\!]^{\mathbf{A}}_{\alpha} &= \;=^{\mathbf{A}}([\![t_1]\!]^{\mathbf{A}}_{\alpha},[\![t_2]\!]^{\mathbf{A}}_{\alpha})\;, \\
[\![\neg A]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} & \text{if } [\![A]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f} \\ \mathsf{f} & \text{if } [\![A]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t} \\ \mathsf{b} & \text{otherwise,} \end{cases} \\
[\![A_1 \land A_2]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} & \text{if } [\![A_1]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t} \text{ and } [\![A_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t} \\ \mathsf{f} & \text{if } [\![A_1]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f} \text{ or } [\![A_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f} \\ \mathsf{b} & \text{otherwise,} \end{cases} \\
[\![A_1 \lor A_2]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} & \text{if } [\![A_1]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t} \text{ or } [\![A_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t} \\ \mathsf{f} & \text{if } [\![A_1]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f} \text{ and } [\![A_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f} \\ \mathsf{b} & \text{otherwise,} \end{cases} \\
[\![A_1 \supset A_2]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} & \text{if } [\![A_1]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f} \text{ or } [\![A_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t} \\ \mathsf{f} & \text{if } [\![A_1]\!]^{\mathbf{A}}_{\alpha} \neq \mathsf{f} \text{ and } [\![A_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f} \\ \mathsf{b} & \text{otherwise,} \end{cases} \\
[\![\forall x \bullet A]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} & \text{if, for all } d \in \mathcal{U}^{\mathbf{A}},\ [\![A]\!]^{\mathbf{A}}_{\alpha(x \to d)} = \mathsf{t} \\ \mathsf{f} & \text{if, for some } d \in \mathcal{U}^{\mathbf{A}},\ [\![A]\!]^{\mathbf{A}}_{\alpha(x \to d)} = \mathsf{f} \\ \mathsf{b} & \text{otherwise.} \end{cases} \\
[\![\exists x \bullet A]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} & \text{if, for some } d \in \mathcal{U}^{\mathbf{A}},\ [\![A]\!]^{\mathbf{A}}_{\alpha(x \to d)} = \mathsf{t} \\ \mathsf{f} & \text{if, for all } d \in \mathcal{U}^{\mathbf{A}},\ [\![A]\!]^{\mathbf{A}}_{\alpha(x \to d)} = \mathsf{f} \\ \mathsf{b} & \text{otherwise.} \end{cases}
\end{aligned}
$$

are meta-variables ranging over all terms from $\mathcal{T}(\Sigma)$, and A, A_1, and A_2 are meta-variables ranging over all formulas from $\mathcal{F}(\Sigma)$.

The following theorem is a decidability result concerning valuations of formulas in structures with a finite domain.

Theorem 1. *Let* $\mathbf{A}$ *be a structure of* $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ *such that* $\mathcal{U}^{\mathbf{A}}$ *is finite, and let* α *be an assignment in* $\mathbf{A}$*. Then, for all* $A \in \mathcal{F}(\Sigma)$*,* $[\![A]\!]^{\mathbf{A}}_{\alpha} \in \{\mathsf{t},\mathsf{b}\}$ *is decidable.*

Proof. This is easy to prove by induction on the structure of A. $\square$

Let Γ be a set of formulas from $\mathcal{F}(\Sigma)$. Then a *model of* Γ is a structure $\mathbf{A}$ of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ such that, for all assignments α in $\mathbf{A}$, for all $A \in \Gamma$, $[\![A]\!]^{\mathbf{A}}_{\alpha} \in \{\mathsf{t},\mathsf{b}\}$.

Logical consequence Let Γ and Δ be sets of formulas from $\mathcal{F}(\Sigma)$. Then Δ *is a logical consequence of* Γ, written $\Gamma \vDash \Delta$, iff for all structures $\mathbf{A}$ of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$, for all assignments α in $\mathbf{A}$, $[\![A]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f}$ for some $A \in \Gamma$ or $[\![A']\!]^{\mathbf{A}}_{\alpha} \in \{\mathsf{t}, \mathsf{b}\}$ for some $A' \in \Delta$.

The sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ presented in Section 3 is sound and complete with respect to logical consequence as defined above.

Theorem 2. *Let Γ and Δ be finite sets of formulas from $\mathcal{F}(\Sigma)$. Then $\Gamma \vdash \Delta$ iff $\Gamma \vDash \Delta$.*

Proof. In the proof of this theorem use is made of the fact that a sound and complete sequent calculus proof system for LP°, a logic similar to $\mathrm{LPQ}^{\supset,\mathsf{F}}$, is available in [19]. The differences between the two logics are:

- the proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ does not include a cut rule and the proof system of LP° includes a cut rule, but the latter proof system has the cut-elimination property;
- the $\neg$-R rule and the Repl rule from the proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ differ from the $\neg$-R rule and the Repl rule from the proof system of LP°, but in either proof systems the $\neg$-R rule and the Repl rule from the other proof system are derived inference rules;
- the logical symbols of LP° include the *consistency* connective $\circ$ and the logical symbols of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ do not include this logical symbol, but formulas with it as outermost operator can be defined as abbreviations of formulas in $\mathrm{LPQ}^{\supset,\mathsf{F}}$ as follows: $\circ A$ stands for $(A \supset \mathsf{F}) \lor (\neg A \supset \mathsf{F})$;
- the logical symbols of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ include F, $\supset$, $\land$, and $\forall$ and the logical symbols of LP° do not include these logical symbols, but formulas with them as outermost operator can be defined as abbreviations of formulas in LP° as follows: F stands for $A \land \neg A \land \circ A$ where A is an arbitrary atomic formula, $A_1 \supset A_2$ stands for $(\neg A_1 \land \circ A_1) \lor A_2$, $A_1 \land A_2$ stands for $\neg(\neg A_1 \lor \neg A_2)$, and $\forall x \bullet A$ stands for $\neg \exists x \bullet \neg A$.

For each formula of one of the two logics which is defined above as an abbreviation of a formula in the other logic, the valuation of the former formula in the former logic is the same as the valuation of the latter formula in the latter logic. Moreover, the first two differences mentioned above have no effect on the sequents that can be proved. Therefore, the sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is sound and complete if, for each logical symbol missing in one of the logics, the inference rules for that symbol in the proof system of the other logic become derived inference rules in the proof system of the former logic when the formulas with that symbol as outermost operator are taken for abbreviations of formulas as defined above. It is a routine matter to prove this. $\square$

A non-standard, indirect proof of soundness and completeness is outlined above. This proof outline clarifies why $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is called 'essentially the same as' LP° in Section 1. Moreover, it follows from this proof outline that the admissibility of the structural inference rules of cut and weakening in LP° carries over to $\mathrm{LPQ}^{\supset,\mathsf{F}}$.

A direct proof of soundness and completeness can be given along the same lines as in the proof of Theorem 1 from [17].

There are two minor differences between $\mathrm{LPQ}^{\supset,\mathsf{F}}$ and LP° that are not mentioned in the proof outline above. The first difference is that a predicate symbol is interpreted in LP° as what is sometimes called a paraconsistent relation (see e.g. [2]) and in $\mathrm{LPQ}^{\supset,\mathsf{F}}$ as what may be called the characteristic function of such a relation. However, this difference is nullified in the valuation of formulas. The second difference is that in LP° signatures are restricted to signatures Σ for which $\mathrm{P}_0(\Sigma) = \emptyset$. By consulting the soundness and completeness proofs in [19], it becomes immediately clear that, as expected, this restriction can be removed without effect on the soundness and completeness.

Abbreviations From Section 5 on, we use $\circ A$ and $A_1 \to A_2$ as abbreviations for formulas in $\mathrm{LPQ}^{\supset,\mathsf{F}}$. These abbreviations are defined as follows: $\circ A$ stands for $(A \supset \mathsf{F}) \lor (\neg A \supset \mathsf{F})$ and $A_1 \to A_2$ stands for $(A_1 \supset A_2) \land (\neg A_2 \supset \neg A_1)$. It follows from these definitions that:

$$
\begin{aligned}
[\![\circ A]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} & \text{if } [\![A]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t} \text{ or } [\![A]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f} \\ \mathsf{f} & \text{otherwise,} \end{cases} \\
[\![A_1 \to A_2]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} & \text{if } [\![A_1]\!]^{\mathbf{A}}_{\alpha} = \mathsf{f} \text{ or } [\![A_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t} \\ \mathsf{b} & \text{if } [\![A_1]\!]^{\mathbf{A}}_{\alpha} = \mathsf{b} \text{ and } [\![A_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{b} \\ \mathsf{f} & \text{otherwise,} \end{cases}
\end{aligned}
$$

5 Relational Databases Viewed through $\mathrm{LPQ}^{\supset,\mathsf{F}}$

In this section, relational databases are considered from the perspective of $\mathrm{LPQ}^{\supset,\mathsf{F}}$. A relational database can be considered from a logical point of view in two different ways: either as a model of a logical theory (the model-theoretic view) or as a logical theory (the proof-theoretic view). Here, the second viewpoint is taken. In the definition of the notion of a relational database, use is made of the notions of a relational language and a relational theory. The latter two notions are defined first. The definitions given in this section are based on those given in [21]. However, types are ignored for the sake of simplicity (cf. [12,23]).

Relational languages The pair $(\Sigma, \mathcal{F}(\Sigma))$, where Σ is a signature, is called the *language of* $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$. If Σ satisfies particular conditions, then the language of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ is considered a relational language.

Let Σ be a signature. Then the language $R = (\Sigma, \mathcal{F}(\Sigma))$ of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ is a *relational language* iff it satisfies the following conditions:

- $\mathrm{F}_0(\Sigma)$ is non-empty and finite;
- $\bigcup\{\mathrm{F}_{n+1}(\Sigma) \mid n \in \mathbb{N}\}$ is empty;
- $\mathrm{P}_0(\Sigma)$ is empty;
- $\bigcup\{\mathrm{P}_{n+1}(\Sigma) \mid n \in \mathbb{N}\}$ is finite.

Relational theories Below, we will introduce the notion of a relational theory. In the definition of a relational theory, use is made of four auxiliary notions, namely the notions of an atomic fact, a domain closure axiom, a unique name axiom set, and a completion axiom. These auxiliary notions are defined first.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language. Then an *atomic fact for* R is a formula from $\mathcal{F}(\Sigma)$ of the form $P(c_1, \ldots, c_{n+1})$, where $P \in \mathrm{P}_{n+1}(\Sigma)$ and $c_1, \ldots, c_{n+1} \in \mathrm{F}_0(\Sigma)$.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language. Then the *equality consistency axiom for* R is the formula

$$\forall x, x' \bullet \circ(x = x') \ .$$

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language and let $c_1, \ldots, c_n$ be all members of $\mathrm{F}_0(\Sigma)$. Then the *domain closure axiom for* R is the formula

$$\forall x \bullet (x = c_1 \lor \ldots \lor x = c_n)$$

and the *unique name axiom set for* R is the set of formulas

$$\{\neg(c_i = c_j) \mid 1 \leq i < j \leq n\} \ .$$

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, let $\Lambda \subseteq \mathcal{F}(\Sigma)$ be a finite set of atomic facts for R, and let $P \in \mathrm{P}_{n+1}(\Sigma)$ ($n \in \mathbb{N}$). Suppose that there exist formulas in Λ in which P occurs and let $P(c^1_1, \ldots, c^1_{n+1}), \ldots, P(c^m_1, \ldots, c^m_{n+1})$ be all formulas from Λ in which P occurs. Then the P*-completion axiom for* Λ is the formula

$$\begin{aligned}&\forall x_1, \ldots, x_{n+1} \bullet P(x_1, \ldots, x_{n+1}) \to \\ &\qquad x_1 = c^1_1 \land \ldots \land x_{n+1} = c^1_{n+1} \ \lor\ \ldots\ \lor\ x_1 = c^m_1 \land \ldots \land x_{n+1} = c^m_{n+1} \ .\end{aligned}$$

Suppose that there does not exist a formula in Λ in which P occurs. Then the P*-completion axiom for* Λ is the formula

$$\forall x_1, \ldots, x_{n+1} \bullet P(x_1, \ldots, x_{n+1}) \to \mathsf{F} \ .$$

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language. Then the *relational structure axioms for* R, written $\mathit{RSA}(R)$, is the set of all formulas $A \in \mathcal{F}(\Sigma)$ for which one of the following holds:

- A is the equality consistency axiom for R;
- A is the domain closure axiom for R;
- A is an element of the unique name axiom set for R.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, and let $\Lambda \subseteq \mathcal{F}(\Sigma)$ be a finite set of atomic facts for R. Then the *relational theory for* R *with basis* Λ, written $\mathit{RT}(R, \Lambda)$, is the set of all formulas $A \in \mathcal{F}(\Sigma)$ for which one of the following holds:

- $A \in RSA(R)$;
- $A \in \Lambda$;
- A is the P-completion axiom for Λ for some $P \in \bigcup\{\mathrm{P}_{n+1}(\Sigma) \mid n \in \mathbb{N}\}$.

A set $\Theta \subseteq \mathcal{F}(\Sigma)$ is called a *relational theory for* R if $\Theta = RT(R, \Lambda)$ for some finite set $\Lambda \subseteq \mathcal{F}(\Sigma)$ of atomic facts for R. The elements of this unique Λ are called the *atomic facts of* Θ.

The following theorem is a decidability result concerning provability of sequents $\Gamma \Rightarrow A$ where Γ includes the relational structure axioms for some relational language.

Theorem 3. *Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, and let Γ be a finite subset of $\mathcal{F}(\Sigma)$ such that $RSA(R) \subseteq \Gamma$. Then it is decidable whether, for a formula $A \in \mathcal{F}(\Sigma)$, $\Gamma \Rightarrow A$ is provable.*

Proof. Because $RSA(R) \subseteq \Gamma$, it is sufficient to consider only structures that are models of $RSA(R)$. The domains of these structures have the same finite cardinality. Because in addition there are finitely many predicate symbols in Σ, there exist moreover only finitely many of these structures.

Clearly, it is sufficient to consider only the restrictions of assignments to the set of all variables that occur free in $\Gamma \cup \{A\}$. Because the set of all variables that occur free in $\Gamma \cup \{A\}$ is finite and the domain of the structures to be considered is finite, there exist only finitely many such restrictions and those restrictions are finite.

It follows easily from the above-mentioned finiteness properties and Theorems 1 and 2 that it is decidable whether, for a formula $A \in \mathcal{F}(\Sigma)$, $\Gamma \Rightarrow A$ is provable. $\square$

Relational databases Having defined the notions of an relational language and a relational theory, we are ready to define the notion of a relational database in the setting of $\mathrm{LPQ}^{\supset,\mathsf{F}}$.

A *relational database* DB is a triple (R, Θ, Ξ), where:

- $R = (\Sigma, \mathcal{F}(\Sigma))$ is a relational language;
- Θ is a relational theory for R;
- Ξ is a finite subset of $\mathcal{F}(\Sigma)$.

Θ is called the *relational theory of DB* and Ξ is called the *set of integrity constraints of DB*.

The set Ξ of integrity constraints of a relational database $DB = (R, \Theta, \Xi)$ can be seen as a set of assumptions about the relational theory of the relational database Θ. If the relational theory agrees with these assumptions, then the relational database is called consistent.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, and let $DB = (R, \Theta, \Xi)$ be a relational database. Then DB *is consistent* iff, for each $A \in \mathcal{F}(\Sigma)$ such that A is an atomic fact for R or A is of the form $\neg A'$ where A' is an atomic fact for R:

$$\Theta \Rightarrow A \text{ is provable only if } \Theta, \Xi \Rightarrow \circ A \text{ is provable.}$$

Notice that, if DB is not consistent, $\Theta, \Xi \Rightarrow A$ is provable with the sequent calculus proof system of $\mathrm{CL}(\Sigma)$ for all $A \in \mathcal{F}(\Sigma)$. However, the sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ rules out such an explosion.

Models of relational theories The models of relational theories for a relational language $R = (\Sigma, \mathcal{F}(\Sigma))$ are structures of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ of a special kind.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language. Then a *relational structure for* R is a structure $\mathbf{A}$ of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$ such that:

- for all $d_1, d_2 \in \mathcal{U}^{\mathbf{A}}$, $=^{\mathbf{A}}(d_1, d_2) \in \{\mathsf{t}, \mathsf{f}\}$;
- for all $d \in \mathcal{U}^{\mathbf{A}}$, there exists a $c \in \mathrm{F}_0(\Sigma)$ such that $=^{\mathbf{A}}(d, c^{\mathbf{A}}) = \mathsf{t}$;
- for all $c_1, c_2 \in \mathrm{F}_0(\Sigma)$, $=^{\mathbf{A}}(c_1{}^{\mathbf{A}}, c_2{}^{\mathbf{A}}) = \mathsf{t}$ only if $c_1 \equiv c_2$.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, and let $\mathbf{A}$ be a structure of $\mathrm{LPQ}^{\supset,\mathsf{F}}(\Sigma)$. Then $\mathbf{A}$ is a relational structure for R iff, for all assignments α in $\mathbf{A}$, for all $A \in \mathit{RSA}(R)$, $[\![A]\!]^{\mathbf{A}}_{\alpha} \in \{\mathsf{t}, \mathsf{b}\}$. Moreover, let Θ be a relational theory for R. Then all models of Θ are relational structures for R because $\mathit{RSA}(R) \subseteq \Theta$. Θ does not have a unique model up to isomorphism. Θ's predicate completion axioms fail to enforce a unique model up to isomorphism. However, identification of t and b in the models of Θ yields uniqueness up to isomorphism.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, and let $\mathbf{A}$ be a relational structure for R. Then we write $\nabla\mathbf{A}$ for the relational structure $\mathbf{A}'$ for R such that:

- $\mathcal{U}^{\mathbf{A}'} = \mathcal{U}^{\mathbf{A}}$;
- for each $c \in \mathrm{F}_0(\Sigma)$, $c^{\mathbf{A}'} = c^{\mathbf{A}}$;
- for each $n \in \mathbb{N}$, for each $P \in \mathrm{P}_{n+1}(\Sigma)$, for each $d_1, \ldots, d_{n+1} \in \mathcal{U}^{\mathbf{A}'}$, $P^{\mathbf{A}'}(d_1, \ldots, d_{n+1}) = \begin{cases} \mathsf{t} & \text{if } P^{\mathbf{A}}(d_1, \ldots, d_{n+1}) \in \{\mathsf{t}, \mathsf{b}\} \\ \mathsf{f} & \text{otherwise;} \end{cases}$
- for each $d_1, d_2 \in \mathcal{U}^{\mathbf{A}'}$, $=^{\mathbf{A}'}(d_1, d_2) = {=^{\mathbf{A}}}(d_1, d_2)$.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, let Θ be a relational theory for R, and let $\mathbf{A}$ be a model of Θ. Then $\nabla\mathbf{A}$, i.e. $\mathbf{A}$ with t and b identified, is in essence a relational database as originally introduced in [8].

Theorem 4. *Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, let Θ be a relational theory for R, and let $\mathbf{A}$ and $\mathbf{A}'$ be models of Θ. Then $\nabla\mathbf{A}$ and $\nabla\mathbf{A}'$ are isomorphic relational structures.*

Proof. The proof goes in almost the same way as the proof of part 1 of Theorem 3.1 from [21]. The only point of attention is that it may be the case that, for some $P \in \mathrm{P}_{n+1}(\Sigma)$ and $c_1, \ldots, c_{n+1} \in \mathrm{F}_0(\Sigma)$ ($n \in \mathbb{N}$), either $[\![P(c_1, \ldots, c_{n+1})]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t}$ and $[\![P(c_1, \ldots, c_{n+1})]\!]^{\mathbf{A}'}_{\alpha} = \mathsf{b}$ or $[\![P(c_1, \ldots, c_{n+1})]\!]^{\mathbf{A}}_{\alpha} = \mathsf{b}$ and $[\![P(c_1, \ldots, c_{n+1})]\!]^{\mathbf{A}'}_{\alpha} = \mathsf{t}$. But, if this is the case, $[\![P(c_1, \ldots, c_{n+1})]\!]^{\nabla\mathbf{A}}_{\alpha} = \mathsf{t}$ and $[\![P(c_1, \ldots, c_{n+1})]\!]^{\nabla\mathbf{A}'}_{\alpha} = \mathsf{t}$. □

Theorem 5. *Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, and let $\mathbf{A}$ be a relational structure for R. Then there exists a relational theory Θ for R such that $\mathbf{A}$ is a model of Θ.*

Proof. The proof goes in the same way as the proof of part 2 of Theorem 3.1 from [21]. □

6 Query Answering Viewed through $\mathrm{LPQ}^{\supset,\mathsf{F}}$

In this section, queries applicable to a relational database and their answers are considered from the perspective of $\mathrm{LPQ}^{\supset,\mathsf{F}}$. As a matter of fact, the queries introduced below are closely related to the relational-calculus-oriented queries originally originally introduced in [9].

Queries As to be expected in the current setting, a query applicable to a relational database involves a formula of $\mathrm{LPQ}^{\supset,\mathsf{F}}$.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language. Then a *query for* R is an expression of the form $(x_1, \ldots, x_n) \bullet A$, where:

- $x_1, \ldots, x_n \in \mathcal{V}$;
- $A \in \mathcal{F}(\Sigma)$ and all variables that are free in A are among $x_1, \ldots, x_n$.

Let $DB = (R, \Theta, \Xi)$ be a relational database. Then a query is *applicable to* DB iff it is a query for R.

Answers Answering a query with respect to a consistent relational database amounts to looking for closed instances of the formula concerned that are logical consequences of a relational theory. The main issue concerning query answering is how to deal with inconsistent relational databases.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, let $DB = (R, \Theta, \Xi)$ be a relational database, and let $(x_1, \ldots, x_n) \bullet A$ be a query that is applicable to DB. Then an *answer to* $(x_1, \ldots, x_n) \bullet A$ *with respect to* DB is a $(c_1, \ldots, c_n) \in \mathrm{F}_0(\Sigma)^n$ for which $\Theta \Rightarrow [x_1 := c_1] \ldots [x_n := c_n]A$ is provable.

The above definition of an answer to a query with respect to a database does not take into account the integrity constraints of the database concerned.

Consistent answers The definition of a consistent answer given below is based on the following:

- the observation that the formula that corresponds to an answer, being a logical consequence of the relational theory of the database, is also a logical consequence of one or more sets of atomic facts and negations of atomic facts that are logical consequences of the relational theory of the database;
- the idea that in the case of a consistent answer there must be such a set that does not contain an atomic fact or negation of an atomic fact that causes the database to be inconsistent.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language. Then an *semi-atomic fact for* R is a formula from $\mathcal{F}(\Sigma)$ of the form $P(c_1, \ldots, c_{n+1})$ or the form $\neg P(c_1, \ldots, c_{n+1})$, where $P \in \mathrm{P}_{n+1}(\Sigma)$ and $c_1, \ldots, c_{n+1} \in \mathrm{F}_0(\Sigma)$.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, let $DB = (R, \Theta, \Xi)$ be a relational database, and let $(x_1, \ldots, x_n) \bullet A$ be a query that is applicable to DB. Then a *consistent answer to* $(x_1, \ldots, x_n) \bullet A$ *with respect to* DB is a $(c_1, \ldots, c_n) \in \mathrm{F}_0(\Sigma)^n$ for which there exists a $\Phi \subseteq \{A' \mid A' \text{ is a semi-atomic fact for } R\}$ such that:

– for all $A' \in \Phi$, $\Theta \Rightarrow A'$ is provable and $\Theta, \Xi \Rightarrow \circ A'$ is provable;
– $\Phi, RSA(R) \Rightarrow [x_1 := c_1] \ldots [x_n := c_n]A$ is provable.

The above definition of a consistent answer to a query with respect to a database is reminiscent of the definition of a consistent answer to a query with respect to a database given in [4]. It simply accepts that a database is inconsistent and excludes the source or sources of the inconsistency from being used in consistent query answering.

Strongly consistent answers The definition of a strongly consistent answer given below is not so tolerant of inconsistency and makes use of consistent repairs of the database. The idea is that an answer is strongly consistent if it is an answer with respect to any minimally repaired version of the original database.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, and let $\Lambda \subseteq \mathcal{F}(\Sigma)$ be a finite set of atomic facts for R. Then, following [1], the binary relation $\leq_\Lambda$ on the set of all finite sets of atomic facts for R is defined by:

$$\Lambda' \leq_\Lambda \Lambda'' \text{ iff } (\Lambda \setminus \Lambda') \cup (\Lambda' \setminus \Lambda) \subseteq (\Lambda \setminus \Lambda'') \cup (\Lambda'' \setminus \Lambda) .$$

Intuitively, $\Lambda' \leq_\Lambda \Lambda''$ indicates that the extent to which Λ' differs from Λ is less than the extent to which Λ'' differs from Λ.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, let $\Lambda \subseteq \mathcal{F}(\Sigma)$ be a finite set of atomic facts for R, and let Ξ is a finite subset of $\mathcal{F}(\Sigma)$. Then Λ *is consistent with* Ξ iff for all semi-atomic facts A for R, $RT(R, \Lambda) \Rightarrow A$ is provable only if $RT(R, \Lambda), \Xi \Rightarrow \neg A$ is not provable. We write $Con(\Xi)$ for the set of all finite sets of atomic facts for R that are consistent with Ξ.

Let $R = (\Sigma, \mathcal{F}(\Sigma))$ be a relational language, let $\Lambda \subseteq \mathcal{F}(\Sigma)$ be a finite set of atomic facts for R, let $DB = (R, RT(R, \Lambda), \Xi)$ be a relational database, and let $(x_1, \ldots, x_n) \bullet A$ be a query that is applicable to DB. Then a *strongly consistent answer to* $(x_1, \ldots, x_n) \bullet A$ *with respect to* DB is a $(c_1, \ldots, c_n) \in \mathrm{F}_0(\Sigma)^n$ such that, for each Λ' that is $\leq_\Lambda$-minimal in $Con(\Xi)$, $RT(R, \Lambda') \Rightarrow [x_1 := c_1] \ldots [x_n := c_n]A$ is provable. The elements of $Con(\Xi)$ that are $\leq_\Lambda$-minimal in $Con(\Xi)$ are called the *repairs of* Λ.

The above definition of a strongly consistent answer to a query with respect to a database is essentially the same as the definition of a consistent answer to a query with respect to a database given in [1]. It represents, presumably, the first view on what the repairs of an inconsistent database are. Other views have been taken in e.g. [14,13,6,5,3].

Decidability The following theorem concerns the decidability of being an answer to a query.

Theorem 6. *Let* $R = (\Sigma, \mathcal{F}(\Sigma))$ *be a relational language, let* $DB = (R, \Theta, \Xi)$ *be a relational database, and let* $(x_1, \ldots, x_n) \bullet A$ *be a query applicable to* DB. *Then it is decidable whether, for* $(c_1, \ldots, c_n) \in \mathrm{F}_0(\Sigma)^n$*:*

– $(c_1, \ldots, c_n)$ *is an answer to* $(x_1, \ldots, x_n) \bullet A$ *with respect to DB;*
– $(c_1, \ldots, c_n)$ *is a consistent answer to* $(x_1, \ldots, x_n) \bullet A$ *with respect to DB;*
– $(c_1, \ldots, c_n)$ *is a strongly consistent answer to* $(x_1, \ldots, x_n) \bullet A$ *with respect to DB.*

Proof. Each of these decidability results follows immediately from Theorem 3 and the definition of the kind of answer concerned. $\square$

As a corollary of Theorem 6, we have that the set of answers to a query, the set of consistent answers to a query, and the set of strongly consistent answers to a query are computable.

7 Examples of Query Answering

For a given database and query applicable to that database, the set of all answers, the set of all consistent answers, and the set of all strongly consistent answers may be different. The examples of query answering given below illustrate this. The examples are kept extremely simple so that readers that are not initiated in the sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ can understand the remarks made about the provability of sequents.

Example 1 Consider the relational database whose relational language, say R, has constant symbols a and b and unary predicate symbols P and Q, whose relational theory is the relational theory of which $P(a)$, $P(b)$, and $Q(a)$ are the atomic facts, and whose only integrity constraint is $\forall x \bullet \neg(P(x) \land Q(x))$. Moreover, consider the query $x \bullet P(x)$. Clearly, the set of answers is $\{a, b\}$.

The sets of semi-atomic formulas that are logical consequences of the relational theory and do not cause the database to be inconsistent are $\{P(b), \neg Q(b)\}$ and all its subsets. We have:

– $P(b), \neg Q(b), RSA(R) \Rightarrow P(a)$ is not provable;
– $P(b), \neg Q(b), RSA(R) \Rightarrow P(b)$ is provable.

Hence, the set of consistent answers is $\{b\}$.

The repairs of $\{P(a), P(b), Q(a)\}$ are $\{P(a), P(b)\}$ and $\{P(b), Q(a)\}$. We have:

– $RT(R, \{P(b), Q(a)\}) \Rightarrow P(a)$ is not provable;
– $RT(R, \{P(a), P(b)\}) \Rightarrow P(b)$ is provable;
– $RT(R, \{P(b), Q(a)\}) \Rightarrow P(b)$ is provable.

Hence, the set of strongly consistent answers is $\{b\}$.

In this example, the set of all answers differs from the set of all consistent answers and the set of all strongly consistent answers, but the set of all consistent answers and the set of all strongly consistent answers are the same. The repairs of the database are obtained by deletion of atomic facts.

Example 2 Consider the relational database whose relational language, say R, has constant symbols a, b, and c and unary predicate symbols P and Q, whose relational theory is the relational theory of which $P(a)$, $P(b)$, $Q(a)$, and $Q(c)$ are the atomic facts, and whose only integrity constraint is $\forall x \bullet P(x) \rightarrow Q(x)$. Moreover, consider the query $x \bullet P(x)$. Clearly, the set of answers is $\{a, b\}$.

The sets of semi-atomic formulas that are logical consequences of the relational theory and do not cause the database to be inconsistent are $\{P(a), \neg P(c), Q(a), \neg Q(b), Q(c)\}$ and all its subsets. We have:

- $P(a), \neg P(c), Q(a), \neg Q(b), Q(c), RSA(R) \Rightarrow P(a)$ is provable;
- $P(a), \neg P(c), Q(a), \neg Q(b), Q(c), RSA(R) \Rightarrow P(b)$ is not provable.

Hence, the set of consistent answers is $\{a\}$.

The repairs of $\{P(a), P(b), Q(a), Q(c)\}$ are $\{P(a), P(b), Q(a), Q(b), Q(c)\}$ and $\{P(a), Q(a), Q(c)\}$. We have:

- $RT(R, \{P(a), P(b), Q(a), Q(b), Q(c)\}) \Rightarrow P(a)$ is provable;
- $RT(R, \{P(a), Q(a), Q(c)\}) \Rightarrow P(a)$ is provable;
- $RT(R, \{P(a), Q(a), Q(c)\}) \Rightarrow P(b)$ is not provable.

Hence, the set of strongly consistent answers is $\{a\}$.

In this example, like in the previous example, the set of all answers differs from the set of all consistent answers and the set of all strongly consistent answers, but the set of all consistent answers and the set of all strongly consistent answers are the same. Unlike in the previous example, one of the repairs of the database is obtained by deletion of an atomic fact and the other is obtained by insertion of an atomic fact.

Example 3 Consider the relational database whose relational language, say R, has constant symbols a, b, c, d, e, f, and g and unary predicate symbol P, whose relational theory is the relational theory of which $P(a, b, c)$, $P(a, c, d)$, $P(a, c, e)$, and $P(b, f, g)$ are the atomic facts, and whose only integrity constraint is $\forall x, y, z, y', z' \bullet (P(x, y, z) \wedge P(x, y', z')) \rightarrow y = y'$. Moreover, consider the query $y \bullet \exists x, z \bullet P(x, y, z)$. Clearly, the set of answers is $\{b, c, f\}$.

The sets of semi-atomic formulas that are logical consequences of the relational theory and do not cause the database to be inconsistent include $\{P(a, b, c), P(b, f, g)\}$ and $\{P(a, c, d), P(a, c, e), P(b, f, g)\}$. We have:

- $P(a, b, c), P(b, f, g), RSA(R) \Rightarrow \exists x, z \bullet P(x, b, z)$ is provable;
- $P(a, c, d), P(a, c, e), P(b, f, g), RSA(R) \Rightarrow \exists x, z \bullet P(x, c, z)$ is provable;
- $P(a, b, c), P(b, f, g), RSA(R) \Rightarrow \exists x, z \bullet P(x, f, z)$ is provable.

Because a, d, e, and g are no answers, they cannot be consistent answers. Hence, the set of consistent answers is $\{b, c, f\}$.

The repairs of $\{P(a, b, c), P(a, c, d), P(a, c, e), P(b, f, g)\}$ are $\{P(a, b, c), P(b, f, g)\}$ and $\{P(a, c, d), P(a, c, e), P(b, f, g)\}$. We have:

- $RT(R, \{P(a,c,d), P(a,c,e), P(b,f,g)\}) \Rightarrow \exists x, z \bullet P(x,b,z)$ is not provable;
- $RT(R, \{P(a,b,c), P(b,f,g)\}) \Rightarrow \exists x, z \bullet P(x,c,z)$ is not provable;
- $RT(R, \{P(a,c,d), P(a,c,e), P(b,f,g)\}) \Rightarrow \exists x, z \bullet P(x,f,z)$ is provable;
- $RT(R, \{P(a,b,c), P(b,f,g)\}) \Rightarrow \exists x, z \bullet P(x,f,z)$ is provable.

Because a, d, e, and g are no answers, they cannot be strongly consistent answers. Hence, the set of strongly consistent answers is $\{f\}$.

In this example, unlike in the previous two examples, the set of all answers and the set of all consistent answers are the same, but the set of all consistent answers differs from the set of all strongly consistent answers. Like in the first example, the repairs of this database are obtained by deletion of atomic facts.

8 Some remarks about consistent query answering

The definition of a consistent answer to a query with respect to a database given in Section 6 simply accepts that a database is inconsistent and excludes the source or sources of inconsistency from being used in consistent query answering. Several considerations underlying this definition are mentioned in the next two paragraphs.

Seeing the extensional nature of the atomic facts of a database and the intensional nature of its integrity constraints, it is natural to consider the presence or absence of atomic facts in a database that causes inconsistency with its integrity constraints suspect and consequently not to use it in answering a query with respect to the database. The plain choice not to use the source or sources of inconsistency in answering a query does not result in additional choices to be made.

The only accepted alternative to deal with an inconsistent database is to base the answers on consistent databases, called repairs, obtained by deletion and/or addition and/or alteration of atomic facts from the inconsistent database that differ to a minimal extent from the inconsistent database. This alternative requires rather artificial choices to be made concerning, among other things, the kinds of changes (deletions, additions, alterations) that may be made to the original database and what is taken as the extent to which two databases differ.

The definition of a consistent answer to a query with respect to a database given in Section 6 is reminiscent of the definition of a consistent answer to a query with respect to a database given in [4]. That paper is, to my knowledge, the first paper in which consistent query answering in inconsistent databases is considered. The definition of consistent query answer given in that paper is based on provability in a natural deduction proof system of first-order minimal logic, a paraconsistent logic that is much less close to classical logic than $\mathrm{LPQ}^{\supset,\mathsf{F}}$.

What is missing in [4] is a semantics with respect to which the presented proof system is sound and complete. This leaves it somewhat unclear how the logical versions of the relevant notions (relational database, query, etc.) defined in that paper are related to their standard version. The Kripke semantics of the propositional fragment of minimal logic that can be found in various publications leaves this unclear as well.

The definition of a strongly consistent answer to a query with respect to a database given in this section is essentially the same as the definition of a consistent answer to a query with respect to a database given in [1]. It is, to my knowledge, the first definition of a consistent answer based on the idea that an answer is consistent if it is an answer with respect to any minimally repaired version of the original database. Different views of what is a minimally repaired version of a database are plausible. Views that differ from the original one have been considered in e.g. [14,13,6,5,3].

In [4], the definition of a consistent answer is based on the idea that a (usually large) part of an inconsistent database is consistent and that a consistent answer is simply an answer with respect to the consistent part of the database. From the viewpoint taken in [1], this means that only one repair is considered. Because there is in general more than one repair of a database, this is called a shortcoming in [7]. However, the implicit assumption that it is necessary to use the auxiliary notion of a repair in defining the notion of a consistent answer is nowhere substantiated.

9 Concluding Remarks

This paper builds heavily on the following views related to relational databases and consistent query answering:

- the proof-theoretic view of [21] on what is a relational databases, a query applicable to a relational database, and an answer to a query with respect to a consistent relational database;
- the view of [4] on what is a consistent answer to a query with respect to an inconsistent relational database;
- the view of [1] on what is a consistent answer to a query with respect to an inconsistent relational database.

The view of Reiter [21] has been combined with the view of Bry [4] as well as with the view of Arenas et al [1] and adapted to the setting of the paraconsistent logic $\mathrm{LPQ}^{\supset,\mathsf{F}}$. This has led to one coherent view on relational databases and consistent query answering expressed in a setting that is more suitable to this end than classical logic or minimal logic.

The notion of a relational theory can be generalized by allowing its basis to be a set of Horn clauses and adapting the completion axioms as sketched in [12]. This generalization gives rise to a generalization of the notion of a relational database that is generally known as the notion of a definite deductive database. The definitions of an answer, a consistent answer, and a strongly consistent answer given in this paper are also applicable to this generalization of the notion of a relational database. Further generalizion of the notion of an indefinite deductive database is a different matter.

The presented sequent calculus proof system of $\mathrm{LPQ}^{\supset,\mathsf{F}}$, which is sound and complete with respect of the given three-valued semantics of $\mathrm{LPQ}^{\supset,\mathsf{F}}$, is new.

References

1. Arenas, M., Bertossi, L., Chomicki, J.: Consistent query answers in inconsistent databases. In: PODS 1999. pp. 68–79. ACM Press (1999)
2. Bagai, R., Sunderraman, R.: A paraconsistent relational data model. International Journal of Computer Mathematics 55(1–2), 39–55 (1995)
3. Bertossi, L.: Database repairs and consistent query answering: Origins and further developments. In: PODS 2019. pp. 48–58. ACM Press (2019)
4. Bry, F.: Query answering in information systems with integrity constraints. In: Jajodia, S., List, W., McGregor, G., Strous, L. (eds.) IICIS 1997. pp. 113–130. IFIPAICT, Chapman and Hall, London (1997)
5. Calautti, M., Libkin, L., Pieris, A.: An operational approach to consistent query answering. In: PODS 2018. pp. 239–251. ACM Press (2018)
6. ten Cate, B., Fontaine, G., Kolaitis, P.G.: On the data complexity of consistent query answering. In: ICDT 2012. pp. 22–33. ACM Press (2012)
7. Chomicki, J.: Consistent query answering. In: Jajodia, S., Strous, L. (eds.) IICIS 2003. IFIP, vol. 140, pp. 219–239. Springer-Verlag (2004)
8. Codd, E.F.: A relational model for large shared data banks. Communications of the ACM 13(6), 377–387 (1970)
9. Codd, E.F.: Relational completeness of data base sublanguage. Tech. Rep. RJ 987, IBM, San Jose, CA (1972)
10. Colacito, A., de Jongh, D., Vargas, A.L.: Subminimal negation. Soft Computing 21(1), 165–174 (2017)
11. D´Ottaviano, I.M.L.: The completeness and compactness of a three-valued first-order logic. Revista Colombiana de Matemáticas 19, 77–94 (1985)
12. Gallaire, H., Minker, J., Nicolas, J.M.: Logic and databases: A deductive approach. Computing Surveys 16(2), 153–185 (1984)
13. Greco, S., Molinaro, C.: Consistent query answering over inconsistent databases. International Journal of Knowledge-based and Intelligent Engineering Systems 15(3), 119–129 (2011)
14. Lopatenko, A., Bertossi, L.: Consistent query answering by minimal-size repairs. In: DEXA 2006. pp. 558–562. IEEE (2006)
15. Middelburg, C.A.: A classical-logic view of a paraconsistent logic. `arXiv:2008.07292v6 [cs.LO]` (August 2020)
16. Middelburg, C.A.: On the strongest three-valued paraconsistent logic contained in classical logic and its dual. Journal of Logic and Computation 31(2), 597–611 (2021)
17. Middelburg, C.A.: Belnap-Dunn logic and query answering in inconsistent databases with null values. `arXiv:2303.05264v1 [cs.DB]` (March 2023)
18. Odintsov, S.P.: The lattice of extensions of the minimal logic. Siberian Advances in Mathematics 17(2), 112–143 (2007)
19. Picollo, L.: Truth in a logic of formal inconsistency: How classical can it get? Logic Journal of the IGPL 28(5), 771–806 (2020)
20. Priest, G.: The logic of paradox. Journal of Philosophical Logic 8(1), 219–241 (1979)
21. Reiter, R.: Towards a logical reconstruction of relational database theory. In: Brodie, M.L., Mylopoulos, J., Schmidt, J.W. (eds.) On Conceptual Modelling, pp. 191–238. Springer-Verlag (1984)
22. Takeuti, G.: Proof Theory. North-Holland, Amsterdam (1975)
23. Vardi, M.Y.: Querying logical databases. Journal of Computer and System Sciences 33(2), 142–160 (1986)