



UvA-DARE (Digital Academic Repository)

Paraconsistent logic and query answering in inconsistent databases

Middelburg, C. A.

DOI

[10.1080/11663081.2024.2312776](https://doi.org/10.1080/11663081.2024.2312776)

Publication date

2024

Document Version

Final published version

Published in

Journal of Applied Non-Classical Logics

License

Article 25fa Dutch Copyright Act (<https://www.openaccess.nl/en/policies/open-access-in-dutch-copyright-law-taverne-amendment>)

[Link to publication](#)

Citation for published version (APA):

Middelburg, C. A. (2024). Paraconsistent logic and query answering in inconsistent databases. *Journal of Applied Non-Classical Logics*, 34(1), 133-154. <https://doi.org/10.1080/11663081.2024.2312776>

General rights


It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.



Paraconsistent logic and query answering in inconsistent databases

C. A. Middelburg 

Informatics Institute, Faculty of Science, University of Amsterdam, Amsterdam, The Netherlands

ABSTRACT

This paper concerns the paraconsistent logic $LPQ^{\supset,F}$ 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 an earlier paper, $LPQ^{\supset,F}$ is presented with a sequent-style natural deduction proof system. In this paper, a sequent calculus proof system is presented instead because such proof systems are generally considered more suitable as the basis of proof search procedures than natural deduction proof systems and proof search procedures can serve as the core of algorithms for computing consistent answers to queries.

ARTICLE HISTORY

Received 27 September 2022
Accepted 22 December 2023


KEYWORDS

Relational database;
inconsistent database;
consistent query answering;
paraconsistent logic; sequent
calculus

1. Introduction

In the area of relational database theory, often the view is taken in which a database is a theory of first-order classical logic, a query is a formula of first-order classical logic, and query answering amounts to proving in first-order classical logic that a formula is a logical consequence of a theory. In Reiter (1984), the term *proof-theoretic view* is introduced for this view and various arguments in favour 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. Arenas et al., 1999). The reason for this is that in classical logic every formula is a logical consequence of an inconsistent theory.

In Bry (1997), 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 Bry (1997), there has been no follow-up of this work. A 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

CONTACT C. A. Middelburg  C.A.Middelburg@uva.nl

a database. Actually, there exists a Kripke semantics of the propositional fragment (see e.g. Colacito et al., 2017), 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 $\text{LPQ}^{\supset, F}$, another first-order logic in which not every formula is a logical consequence of an inconsistent theory. A sequent calculus proof system of $\text{LPQ}^{\supset, F}$ 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 $\text{LPQ}^{\supset, F}$. The definitions concerned are based on those given in Reiter (1984). 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 Bry (1997) and the other is essentially the same as the notion of a consistent answer from Arenas et al. (1999).

Proof search procedures can serve as the core of algorithms for computing consistent answers to queries. Sequent calculus proof systems are generally considered more suitable as the basis of proof search procedures than Hilbert proof systems and natural deduction proof systems (for the main reasons, see e.g. Negri & von Plato, 2001). That is why a sequent calculus proof system of $\text{LPQ}^{\supset, F}$ is presented in this paper. The proof system of first-order minimal logic presented in Bry (1997) is a natural deduction proof system. Natural deduction proof systems can of course also be used as the basis of proof search procedures, but much less is known about how this can be done. The lack of any remark about proof search procedures for first-order minimal logic is sometimes considered a shortcoming in Bry (1997) as well.

A logic is called a paraconsistent logic if in the logic not every formula is a logical consequence of an inconsistent theory. Priest (1979) proposed the paraconsistent propositional logic LP (Logic of Paradox) and its first-order extension LPQ. $\text{LPQ}^{\supset, F}$ is LPQ enriched with a falsity constant and an implication connective for which the standard deduction theorem holds. $\text{LPQ}^{\supset, F}$ is essentially the same as $J^*_3 = (D'Ottaviano, 1985)$ and LP° (Picollo, 2020). In Middelburg (2020), a sequent-style natural deduction proof system for $\text{LPQ}^{\supset, F}$ is presented. Several main properties of the logical consequence relation and the logical equivalence relation of $\text{LPQ}^{\supset, F}$ are also treated in that paper.

In $\text{LPQ}^{\supset, F}$, 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, $\text{LPQ}^{\supset, F}$ is considered a genuine paraconsistent logic and minimal logic is not considered a genuine paraconsistent logic (cf. Odintsov, 2007). Moreover, the properties of $\text{LPQ}^{\supset, F}$ treated in Middelburg (2020) indicate among other things that the logical consequence relation and the logical equivalence relation of $\text{LPQ}^{\supset, F}$ 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 $\text{LPQ}^{\supset, F}$.

The structure of this paper is as follows. First, the language of $\text{LPQ}^{\supset, F}$, a sequent calculus proof system of $\text{LPQ}^{\supset, F}$, and a three-valued semantics of $\text{LPQ}^{\supset, F}$ are presented

(Sections 2–4). Next, relational databases and query answering in possibly inconsistent relational databases are considered from the perspective of $\text{LPQ}^{\supset, F}$ (Sections 5 and 6). After that, examples of query answering are given and some remaining remarks about consistent query answering are made ((Sections 7 and 8). Finally, some concluding remarks are made (Section 9).

2. The language of $\text{LPQ}^{\supset, F}$

This section concerns the language of the first-order paraconsistent logic $\text{LPQ}^{\supset, F}$. First the notion of a signature is introduced and then the terms and formulas of $\text{LPQ}^{\supset, F}$ are defined relative to a signature. Moreover, some relevant notational conventions and abbreviations are presented and some remarks about free variables and substitution are made. In coming sections, many other notions relevant to $\text{LPQ}^{\supset, F}$ are also defined relative to a signature.

Signatures

It is assumed that the following has been given: (a) a countably infinite set \mathcal{Var} of variables, (b) for each $n \in \mathbb{N}$, a countably infinite set \mathcal{Func}_n of function symbols of arity n , and, (c) for each $n \in \mathbb{N}$, a countably infinite set \mathcal{Pred}_n of predicate symbols of arity n . It is also assumed that all these sets and the set $\{=, \neg, \wedge, \vee, \supset, \forall, \exists\}$ are mutually disjoint.

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 $\bigcup\{\mathcal{Func}_n \mid n \in \mathbb{N}\} \cup \bigcup\{\mathcal{Pred}_n \mid n \in \mathbb{N}\}$. We write $\mathcal{Func}_n(\Sigma)$ and $\mathcal{Pred}_n(\Sigma)$, where Σ is a signature and $n \in \mathbb{N}$, for the sets $\Sigma \cap \mathcal{Func}_n$ and $\Sigma \cap \mathcal{Pred}_n$, respectively.

Below the language of $\text{LPQ}^{\supset, F}$ will be defined relative to a signature Σ . This language will be called the language of $\text{LPQ}^{\supset, F}$ over Σ or shortly the language of $\text{LPQ}^{\supset, F}(\Sigma)$. The corresponding proof system and logical consequence relation will be called the proof system of $\text{LPQ}^{\supset, F}(\Sigma)$ and the logical consequence relation of $\text{LPQ}^{\supset, F}(\Sigma)$.

Terms and formulas

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

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

- (1) if $x \in \mathcal{Var}$, then $x \in \mathcal{Term}(\Sigma)$;
- (2) if $c \in \mathcal{Func}_0(\Sigma)$, then $c \in \mathcal{Term}(\Sigma)$;
- (3) if $f \in \mathcal{Func}_{n+1}(\Sigma)$ and $t_1, \dots, t_{n+1} \in \mathcal{Term}(\Sigma)$, then $f(t_1, \dots, t_{n+1}) \in \mathcal{Term}(\Sigma)$.

The set of all *closed terms* of $\text{LPQ}^{\supset, F}(\Sigma)$ is the subset of $\mathcal{Term}(\Sigma)$ that can be formed by applying formation rules 2 and 3 only.

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

- (1) $F \in \mathcal{F}orm(\Sigma)$;
- (2) if $p \in \mathcal{P}red_0(\Sigma)$, then $p \in \mathcal{F}orm(\Sigma)$;
- (3) if $P \in \mathcal{P}red_{n+1}(\Sigma)$ and $t_1, \dots, t_{n+1} \in \mathcal{T}erm(\Sigma)$, then $P(t_1, \dots, t_{n+1}) \in \mathcal{F}orm(\Sigma)$;
- (4) if $t_1, t_2 \in \mathcal{T}erm(\Sigma)$, then $t_1 = t_2 \in \mathcal{F}orm(\Sigma)$;
- (5) if $A \in \mathcal{F}orm(\Sigma)$, then $\neg A \in \mathcal{F}orm(\Sigma)$;
- (6) if $A_1, A_2 \in \mathcal{F}orm(\Sigma)$, then $A_1 \wedge A_2, A_1 \vee A_2, A_1 \supset A_2 \in \mathcal{F}orm(\Sigma)$;
- (7) if $x \in \mathcal{V}ar$ and $A \in \mathcal{F}orm(\Sigma)$, then $\forall x \cdot A, \exists x \cdot A \in \mathcal{F}orm(\Sigma)$.

The set of all *atomic formulas* of $\text{LPQ}^{\supset, \text{F}}(\Sigma)$ is the subset of $\mathcal{F}orm(\Sigma)$ that can be formed by applying formation rules 1–4 only.

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 the classical truth-conditions and falsehood-conditions.

We write $e_1 \equiv e_2$, where e_1 and e_2 are terms from $\mathcal{T}erm(\Sigma)$ or formulas from $\mathcal{F}orm(\Sigma)$, 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 $\mathcal{V}ar$, t as a meta-variable ranging over all terms from $\mathcal{T}erm(\Sigma)$, A as a meta-variable ranging over all formulas from $\mathcal{F}orm(\Sigma)$, and Γ as a meta-variable ranging over all finite sets of formulas from $\mathcal{F}orm(\Sigma)$.

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 $\forall x_1 \cdot \dots \cdot \forall x_n \cdot A$ is usually written as $\forall x_1, \dots, x_n \cdot A$.

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

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 $\mathcal{V}ar$, t be a term from $\mathcal{T}erm(\Sigma)$, and e be a term from $\mathcal{T}erm(\Sigma)$ or a formula from $\mathcal{F}orm(\Sigma)$. 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 $\text{LPQ}^{\supset, \text{F}}(\Sigma)$

In this section, a sequent calculus proof system of $\text{LPQ}^{\supset, \text{F}}(\Sigma)$ 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 $\text{LPQ}^{\supset, \text{F}}(\Sigma)$ 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 $\text{LPQ}^{\supset, \text{F}}(\Sigma)$ which can serve as a proof system for first-order classical logic is also described.

Sequents

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

Informally speaking, a sequent $\Gamma \Rightarrow \Delta$ expresses that, if every formula from Γ is not false, at least one formula from Δ is not false.

Rules of inference

The sequent calculus proof system of $\text{LPQ}^{\supset, \text{F}}(\Sigma)$ consists of the inference rules given in Table 1. In this table, x and y are meta-variables ranging over all variables from $\mathcal{V}\text{ar}$, t , t_1 , and t_2 are meta-variables ranging over all terms from $\mathcal{T}\text{erm}(\Sigma)$, A , A_1 , and A_2 are meta-variables ranging over all formulas from $\mathcal{F}\text{orm}(\Sigma)$, and Γ , Γ' , Δ , and Δ' are meta-variables ranging over all finite sets of formulas from $\mathcal{F}\text{orm}(\Sigma)$.

Derivations and proofs

In $\text{LPQ}^{\supset, \text{F}}(\Sigma)$, a *derivation of a sequent* $\Gamma \Rightarrow \Delta$ from a finite set of sequents \mathcal{H} is a finite sequence $\langle s_1, \dots, s_n \rangle$ of sequents such that s_n equals $\Gamma \Rightarrow \Delta$ and, for each $i \in \{1, \dots, n\}$, 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 $\text{LPQ}^{\supset, \text{F}}(\Sigma)$ whose premises are among s_1, \dots, 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 $\Gamma \Rightarrow \Delta$.

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 conclusion 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 $\mathcal{F}\text{orm}(\Sigma)$ consisting of the following formulas:

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

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

$\boxed{\text{Id}} \frac{}{A, \Gamma \Rightarrow \Delta, A}$	$\boxed{\text{Cut}} \frac{\Gamma \Rightarrow \Delta, A \quad A, \Gamma' \Rightarrow \Delta'}{\Gamma', \Gamma \Rightarrow \Delta, \Delta'}$
$\boxed{\text{F-L}} \frac{}{F, \Gamma \Rightarrow \Delta}$	$\boxed{\neg\text{-R}} \frac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A}$
$\boxed{\wedge\text{-L}} \frac{A_1, A_2, \Gamma \Rightarrow \Delta}{A_1 \wedge A_2, \Gamma \Rightarrow \Delta}$	$\boxed{\wedge\text{-R}} \frac{\Gamma \Rightarrow \Delta, A_1 \quad \Gamma \Rightarrow \Delta, A_2}{\Gamma \Rightarrow \Delta, A_1 \wedge A_2}$
$\boxed{\vee\text{-L}} \frac{A_1, \Gamma \Rightarrow \Delta \quad A_2, \Gamma \Rightarrow \Delta}{A_1 \vee A_2, \Gamma \Rightarrow \Delta}$	$\boxed{\vee\text{-R}} \frac{\Gamma \Rightarrow \Delta, A_1, A_2}{\Gamma \Rightarrow \Delta, A_1 \vee A_2}$
$\boxed{\supset\text{-L}} \frac{\Gamma \Rightarrow \Delta, A_1 \quad A_2, \Gamma \Rightarrow \Delta}{A_1 \supset A_2, \Gamma \Rightarrow \Delta}$	$\boxed{\supset\text{-R}} \frac{A_1, \Gamma \Rightarrow \Delta, A_2}{\Gamma \Rightarrow \Delta, A_1 \supset A_2}$
$\boxed{\forall\text{-L}} \frac{[x := t]A, \Gamma \Rightarrow \Delta}{\forall x \cdot A, \Gamma \Rightarrow \Delta}$	$\boxed{\forall\text{-R}} \frac{\Gamma \Rightarrow \Delta, [x := y]A}{\Gamma \Rightarrow \Delta, \forall x \cdot A} \dagger$
$\boxed{\exists\text{-L}} \frac{[x := y]A, \Gamma \Rightarrow \Delta}{\exists x \cdot A, \Gamma \Rightarrow \Delta} \dagger$	$\boxed{\exists\text{-R}} \frac{\Gamma \Rightarrow \Delta, [x := t]A}{\Gamma \Rightarrow \Delta, \exists x \cdot A}$
$\boxed{\neg\neg\text{-L}} \frac{A, \Gamma \Rightarrow \Delta}{\neg\neg A, \Gamma \Rightarrow \Delta}$	$\boxed{\neg\neg\text{-R}} \frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \neg\neg A}$
$\boxed{\neg\wedge\text{-L}} \frac{\neg A_1, \Gamma \Rightarrow \Delta \quad \neg A_2, \Gamma \Rightarrow \Delta}{\neg(A_1 \wedge A_2), \Gamma \Rightarrow \Delta}$	$\boxed{\neg\wedge\text{-R}} \frac{\Gamma \Rightarrow \Delta, \neg A_1, \neg A_2}{\Gamma \Rightarrow \Delta, \neg(A_1 \wedge A_2)}$
$\boxed{\neg\vee\text{-L}} \frac{\neg A_1, \neg A_2, \Gamma \Rightarrow \Delta}{\neg(A_1 \vee A_2), \Gamma \Rightarrow \Delta}$	$\boxed{\neg\vee\text{-R}} \frac{\Gamma \Rightarrow \Delta, \neg A_1 \quad \Gamma \Rightarrow \Delta, \neg A_2}{\Gamma \Rightarrow \Delta, \neg(A_1 \vee A_2)}$
$\boxed{\neg\supset\text{-L}} \frac{A_1, \neg A_2, \Gamma \Rightarrow \Delta}{\neg(A_1 \supset A_2), \Gamma \Rightarrow \Delta}$	$\boxed{\neg\supset\text{-R}} \frac{\Gamma \Rightarrow \Delta, A_1 \quad \Gamma \Rightarrow \Delta, \neg A_2}{\Gamma \Rightarrow \Delta, \neg(A_1 \supset A_2)}$
$\boxed{\neg\forall\text{-L}} \frac{\neg[x := y]A, \Gamma \Rightarrow \Delta}{\neg\forall x \cdot A, \Gamma \Rightarrow \Delta} \dagger$	$\boxed{\neg\forall\text{-R}} \frac{\Gamma \Rightarrow \Delta, \neg[x := t]A}{\Gamma \Rightarrow \Delta, \neg\forall x \cdot A}$
$\boxed{\neg\exists\text{-L}} \frac{\neg[x := t]A, \Gamma \Rightarrow \Delta}{\neg\exists x \cdot A, \Gamma \Rightarrow \Delta}$	$\boxed{\neg\exists\text{-R}} \frac{\Gamma \Rightarrow \Delta, \neg[x := y]A}{\Gamma \Rightarrow \Delta, \neg\exists x \cdot A} \dagger$
$\boxed{=\text{-Refl}} \frac{t = t, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$	$\boxed{=\text{-Repl}} \frac{[x := t_1]A, \Gamma \Rightarrow \Delta}{t_1 = t_2, [x := t_2]A, \Gamma \Rightarrow \Delta}$

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

Then the sequent $\Gamma \Rightarrow \Delta$ is provable iff $\Gamma_{=}, \Gamma \Rightarrow \Delta$ is provable without using the inference rules $=\text{-Refl}$ and $=\text{-Repl}$. This can easily be proved in the same way as Proposition 7.4 from Takeuti (1975) is proved.

In Middelburg (2020), a proof system of $\text{LPQ}^{\supset, \text{F}}$ formulated as a sequent-style natural deduction system is given.

A proof system of $\text{CL}(\Sigma)$

We use the name CL here to denote a version of classical logic that has the same logical constants, connectives, and quantifiers as $\text{LPQ}^{\supset, \text{F}}$.

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

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

4. Truth and the logical consequence relation of $\text{LPQ}^{\supset, F}(\Sigma)$

The proof system of $\text{LPQ}^{\supset, F}(\Sigma)$ is based on the logical consequence relation of $\text{LPQ}^{\supset, F}(\Sigma)$ defined in this section: a sequent $\Gamma \Rightarrow \Delta$ is provable iff the logical consequence relation holds between Γ and Δ . The logical consequence relation is defined in terms of the valuation of formulas of $\text{LPQ}^{\supset, F}(\Sigma)$. The valuation of formulas of $\text{LPQ}^{\supset, F}(\Sigma)$ 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 valuation of formulas of $\text{LPQ}^{\supset, F}(\Sigma)$ and the logical consequence relation of $\text{LPQ}^{\supset, F}(\Sigma)$ are defined.

Structures

The valuation of terms and formulas from $\mathcal{F}\text{orm}(\Sigma)$ is defined relative to a 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 $\text{LPQ}^{\supset, F}(\Sigma)$ consists of:

- a set $\mathcal{U}^{\mathbf{A}}$, the *domain* of \mathbf{A} , such that $\mathcal{U}^{\mathbf{A}} \neq \emptyset$ and $\mathcal{U}^{\mathbf{A}} \cap \{t, f, b\} = \emptyset$;
- for each $c \in \mathcal{F}\text{unc}_0(\Sigma)$,
an element $c^{\mathbf{A}} \in \mathcal{U}^{\mathbf{A}}$;
- for each $n \in \mathbb{N}$, for each $f \in \mathcal{F}\text{unc}_{n+1}(\Sigma)$,
a function $f^{\mathbf{A}} : \mathcal{U}^{\mathbf{A}^{n+1}} \rightarrow \mathcal{U}^{\mathbf{A}}$;
- for each $p \in \mathcal{P}\text{red}_0(\Sigma)$,
an element $p^{\mathbf{A}} \in \{t, f, b\}$;
- for each $n \in \mathbb{N}$, for each $P \in \mathcal{P}\text{red}_{n+1}(\Sigma)$,
a function $P^{\mathbf{A}} : \mathcal{U}^{\mathbf{A}^{n+1}} \rightarrow \{t, f, b\}$;
- a function $=^{\mathbf{A}} : \mathcal{U}^{\mathbf{A}^2} \rightarrow \{t, f, b\}$ such that, for all $d_1, d_2 \in \mathcal{U}^{\mathbf{A}}$,
 $=^{\mathbf{A}}(d_1, d_2) \in \{t, b\}$ iff $d_1 = d_2$.

Assignments

An assignment in a structure \mathbf{A} of $\text{LPQ}^{\supset, F}(\Sigma)$ assigns elements from $\mathcal{U}^{\mathbf{A}}$ to the variables from $\mathcal{V}\text{ar}$. The valuation of terms and formulas from $\mathcal{F}\text{orm}(\Sigma)$ in \mathbf{A} is defined relative to an assignment α in \mathbf{A} .

Let \mathbf{A} be a structure of $\text{LPQ}^{\supset, F}(\Sigma)$. Then an *assignment in \mathbf{A}* is a function $\alpha : \mathcal{V}\text{ar} \rightarrow \mathcal{U}^{\mathbf{A}}$. For every assignment α in \mathbf{A} , variable $x \in \mathcal{V}\text{ar}$, and element $d \in \mathcal{U}^{\mathbf{A}}$, we write $\alpha(x \rightarrow d)$ for the assignment α' in \mathbf{A} such that $\alpha'(x) = d$ and $\alpha'(y) = \alpha(y)$ if $y \neq x$.

Table 2. Valuation of terms and formulas of $\text{LPQ}^{\supset, \text{f}}(\Sigma)$.

$\llbracket x \rrbracket_{\alpha}^{\mathbf{A}}$	=	$\alpha(x)$,
$\llbracket c \rrbracket_{\alpha}^{\mathbf{A}}$	=	$c^{\mathbf{A}}$,
$\llbracket f(t_1, \dots, t_{n+1}) \rrbracket_{\alpha}^{\mathbf{A}}$	=	$f^{\mathbf{A}}(\llbracket t_1 \rrbracket_{\alpha}^{\mathbf{A}}, \dots, \llbracket t_{n+1} \rrbracket_{\alpha}^{\mathbf{A}})$
$\llbracket f \rrbracket_{\alpha}^{\mathbf{A}}$	=	f ,
$\llbracket p \rrbracket_{\alpha}^{\mathbf{A}}$	=	$p^{\mathbf{A}}$,
$\llbracket P(t_1, \dots, t_{n+1}) \rrbracket_{\alpha}^{\mathbf{A}}$	=	$P^{\mathbf{A}}(\llbracket t_1 \rrbracket_{\alpha}^{\mathbf{A}}, \dots, \llbracket t_{n+1} \rrbracket_{\alpha}^{\mathbf{A}})$,
$\llbracket t_1 = t_2 \rrbracket_{\alpha}^{\mathbf{A}}$	=	$=^{\mathbf{A}}(\llbracket t_1 \rrbracket_{\alpha}^{\mathbf{A}}, \llbracket t_2 \rrbracket_{\alpha}^{\mathbf{A}})$,
$\llbracket \neg A \rrbracket_{\alpha}^{\mathbf{A}}$	=	$\begin{cases} \text{t} & \text{if } \llbracket A \rrbracket_{\alpha}^{\mathbf{A}} = \text{f} \\ \text{f} & \text{if } \llbracket A \rrbracket_{\alpha}^{\mathbf{A}} = \text{t} \\ \text{b} & \text{otherwise,} \end{cases}$
$\llbracket A_1 \wedge A_2 \rrbracket_{\alpha}^{\mathbf{A}}$	=	$\begin{cases} \text{t} & \text{if } \llbracket A_1 \rrbracket_{\alpha}^{\mathbf{A}} = \text{t} \text{ and } \llbracket A_2 \rrbracket_{\alpha}^{\mathbf{A}} = \text{t} \\ \text{f} & \text{if } \llbracket A_1 \rrbracket_{\alpha}^{\mathbf{A}} = \text{f} \text{ or } \llbracket A_2 \rrbracket_{\alpha}^{\mathbf{A}} = \text{f} \\ \text{b} & \text{otherwise,} \end{cases}$
$\llbracket A_1 \vee A_2 \rrbracket_{\alpha}^{\mathbf{A}}$	=	$\begin{cases} \text{t} & \text{if } \llbracket A_1 \rrbracket_{\alpha}^{\mathbf{A}} = \text{t} \text{ or } \llbracket A_2 \rrbracket_{\alpha}^{\mathbf{A}} = \text{t} \\ \text{f} & \text{if } \llbracket A_1 \rrbracket_{\alpha}^{\mathbf{A}} = \text{f} \text{ and } \llbracket A_2 \rrbracket_{\alpha}^{\mathbf{A}} = \text{f} \\ \text{b} & \text{otherwise,} \end{cases}$
$\llbracket A_1 \supset A_2 \rrbracket_{\alpha}^{\mathbf{A}}$	=	$\begin{cases} \text{t} & \text{if } \llbracket A_1 \rrbracket_{\alpha}^{\mathbf{A}} = \text{f} \text{ or } \llbracket A_2 \rrbracket_{\alpha}^{\mathbf{A}} = \text{t} \\ \text{f} & \text{if } \llbracket A_1 \rrbracket_{\alpha}^{\mathbf{A}} \neq \text{f} \text{ and } \llbracket A_2 \rrbracket_{\alpha}^{\mathbf{A}} = \text{f} \\ \text{b} & \text{otherwise,} \end{cases}$
$\llbracket \forall x \cdot A \rrbracket_{\alpha}^{\mathbf{A}}$	=	$\begin{cases} \text{t} & \text{if, for all } d \in \mathcal{U}^{\mathbf{A}}, \llbracket A \rrbracket_{\alpha(x \rightarrow d)}^{\mathbf{A}} = \text{t} \\ \text{f} & \text{if, for some } d \in \mathcal{U}^{\mathbf{A}}, \llbracket A \rrbracket_{\alpha(x \rightarrow d)}^{\mathbf{A}} = \text{f} \\ \text{b} & \text{otherwise.} \end{cases}$
$\llbracket \exists x \cdot A \rrbracket_{\alpha}^{\mathbf{A}}$	=	$\begin{cases} \text{t} & \text{if, for some } d \in \mathcal{U}^{\mathbf{A}}, \llbracket A \rrbracket_{\alpha(x \rightarrow d)}^{\mathbf{A}} = \text{t} \\ \text{f} & \text{if, for all } d \in \mathcal{U}^{\mathbf{A}}, \llbracket A \rrbracket_{\alpha(x \rightarrow d)}^{\mathbf{A}} = \text{f} \\ \text{b} & \text{otherwise.} \end{cases}$

Valuations and models

The valuation of the terms from $\text{Term}(\Sigma)$ 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 $\text{Form}(\Sigma)$ is given by a function mapping formula A , structure \mathbf{A} and assignment α in \mathbf{A} to the element of $\{\text{t}, \text{f}, \text{b}\}$ that is the truth value of A in \mathbf{A} under assignment α . We write $\llbracket t \rrbracket_{\alpha}^{\mathbf{A}}$ and $\llbracket A \rrbracket_{\alpha}^{\mathbf{A}}$ for these valuations.

The valuation functions for the terms from $\text{Term}(\Sigma)$ and the formulas from $\text{Form}(\Sigma)$ are inductively defined in Table 2. In this table, x is a meta-variable ranging over all variables from \mathcal{Var} , c is a meta-variable ranging over all function symbols from $\text{Func}_0(\Sigma)$, f is a meta-variable ranging over all function symbols from $\text{Func}_{n+1}(\Sigma)$ (where n is understood from the context), p is a meta-variable ranging over all predicate symbols from $\text{Pred}_0(\Sigma)$, P is a meta-variable ranging over all predicate symbols from $\text{Pred}_{n+1}(\Sigma)$ (where n is understood from the context), t_1, \dots, t_{n+1} are meta-variables ranging over all terms from $\text{Term}(\Sigma)$, and A, A_1 , and A_2 are meta-variables ranging over all formulas from $\text{Form}(\Sigma)$.

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

Theorem 4.1: Let \mathbf{A} be a structure of $\text{LPQ}^{\supset, \text{F}}(\Sigma)$ such that $\mathcal{U}^{\mathbf{A}}$ is finite, and let α be an assignment in \mathbf{A} . Then it is decidable whether, for a given $A \in \mathcal{F}\text{orm}(\Sigma)$, $\llbracket A \rrbracket_{\alpha}^{\mathbf{A}} \in \{t, b\}$.

Proof: This is easy to prove by induction on the structure of A . ■

Let Γ be a finite set of formulas from $\mathcal{F}\text{orm}(\Sigma)$. Then a *model* of Γ is a structure \mathbf{A} of $\text{LPQ}^{\supset, \text{F}}(\Sigma)$ such that, for all assignments α in \mathbf{A} , for all $A \in \Gamma$, $\llbracket A \rrbracket_{\alpha}^{\mathbf{A}} \in \{t, b\}$.

Logical consequence

Let Γ and Δ be finite sets of formulas from $\mathcal{F}\text{orm}(\Sigma)$. Then Δ is a *logical consequence* of Γ , written $\Gamma \models \Delta$, iff for all structures \mathbf{A} of $\text{LPQ}^{\supset, \text{F}}(\Sigma)$, for all assignments α in \mathbf{A} , $\llbracket A \rrbracket_{\alpha}^{\mathbf{A}} = f$ for some $A \in \Gamma$ or $\llbracket A' \rrbracket_{\alpha}^{\mathbf{A}} \in \{t, b\}$ for some $A' \in \Delta$.

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

Theorem 4.2: Let Γ and Δ be finite sets of formulas from $\mathcal{F}\text{orm}(\Sigma)$. Then $\Gamma \Rightarrow \Delta$ is provable in the sequent calculus proof system of $\text{LPQ}^{\supset, \text{F}}(\Sigma)$ iff $\Gamma \models \Delta$.

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 $\text{LPQ}^{\supset, \text{F}}$, is available in Picollo (2020). The differences between the two logics are:

- the \rightarrow -R rule and the $=$ -Repl rule from the proof system of $\text{LPQ}^{\supset, \text{F}}$ differ from the corresponding rules from the proof system of LP° , but in either proof system the rules from the other proof system are both derived inference rules;
- the logical symbols of LP° include the *consistency* connective \circ and the logical symbols of $\text{LPQ}^{\supset, \text{F}}$ do not include this logical symbol, but formulas with it as outermost operator can be defined as abbreviations of formulas in $\text{LPQ}^{\supset, \text{F}}$ as follows: $\circ A$ stands for $(A \supset F) \vee (\neg A \supset F)$;
- the logical symbols of $\text{LPQ}^{\supset, \text{F}}$ include F , \supset , \wedge , 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 $A \wedge \neg A \wedge \circ A$ where A is an arbitrary atomic formula, $A_1 \supset A_2$ stands for $(\neg A_1 \wedge \circ A_1) \vee A_2$, $A_1 \wedge A_2$ stands for $\neg(\neg A_1 \vee \neg A_2)$, 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 difference mentioned above has no effect on the sequents that can be proved. Therefore, the sequent calculus proof system of $\text{LPQ}^{\supset, \text{F}}$ 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. ■

A non-standard, indirect proof of soundness and completeness is outlined above. This proof outline clarifies why $\text{LPQ}^{\supset, \text{F}}$ is called ‘essentially the same as’ LP° in Section 1.

Moreover, it follows from this proof outline that the cut elimination property of LP° carries over to $LPQ^{\supset, F}$.

There are two minor differences between $LPQ^{\supset, F}$ and LP° that are not mentioned in the proof outline above. The first difference is that a predicate symbol is interpreted in structures of LP° as what is sometimes called a paraconsistent relation (see e.g. Bagai & Sunderraman, 1995) and in structures of $LPQ^{\supset, F}$ as what may be called the characteristic function of such a relation. However, this difference is nullified in the valuation of formulas of the form $P(t_1, \dots, t_{n+1})$. The second difference is that in LP° signatures are restricted to signatures Σ for which $\mathcal{Pred}_0(\Sigma) = \emptyset$. By consulting the soundness and completeness proofs in Picollo (2020), it becomes immediately clear that, as expected, this restriction can be removed without effect on the soundness and completeness.

Abbreviations

From Section 5 onwards, we use $\circ A$ and $A_1 \rightarrow A_2$ as abbreviations for formulas in $LPQ^{\supset, F}$. These abbreviations are defined as follows: $\circ A$ stands for $(A \supset F) \vee (\neg A \supset F)$ and $A_1 \rightarrow A_2$ stands for $(A_1 \supset A_2) \wedge (\neg A_2 \supset \neg A_1)$. It follows from these definitions that:

$$\begin{aligned} \llbracket \circ A \rrbracket_\alpha^A &= \begin{cases} t & \text{if } \llbracket A \rrbracket_\alpha^A = t \text{ or } \llbracket A \rrbracket_\alpha^A = f \\ f & \text{otherwise,} \end{cases} \\ \llbracket A_1 \rightarrow A_2 \rrbracket_\alpha^A &= \begin{cases} t & \text{if } \llbracket A_1 \rrbracket_\alpha^A = f \text{ or } \llbracket A_2 \rrbracket_\alpha^A = t \\ b & \text{if } \llbracket A_1 \rrbracket_\alpha^A = b \text{ and } \llbracket A_2 \rrbracket_\alpha^A = b \\ f & \text{otherwise,} \end{cases} \end{aligned}$$

5. Relational databases viewed through $LPQ^{\supset, F}$

In this section, relational databases are considered from the perspective of $LPQ^{\supset, F}$. 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 Reiter (1984). However, types are ignored for the sake of simplicity (cf. Gallaire et al., 1984; Vardi, 1986).

Relational languages

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

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

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

- $\mathcal{P}red_0(\Sigma)$ is empty;
- $\bigcup\{\mathcal{P}red_{n+1}(\Sigma) \mid n \in \mathbb{N}\}$ 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 a number of auxiliary notions. These auxiliary notions are defined first.

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language. Then an *atomic fact for R* is a formula from $\mathcal{F}orm(\Sigma)$ of the form $P(c_1, \dots, c_{n+1})$, where $P \in \mathcal{P}red_{n+1}(\Sigma)$ and $c_1, \dots, c_{n+1} \in \mathcal{F}unc_0(\Sigma)$.

The *equality consistency axiom* is the formula

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

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language and let c_1, \dots, c_n be all members of $\mathcal{F}unc_0(\Sigma)$. Then the *domain closure axiom for R* is the formula

$$\forall x(x = c_1 \vee \dots \vee x = c_n)$$

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 $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, let $\Lambda \subseteq \mathcal{F}orm(\Sigma)$ be a finite set of atomic facts for R , and let $P \in \mathcal{P}red_{n+1}(\Sigma)$ ($n \in \mathbb{N}$). Suppose that there exist formulas in Λ in which P occurs and let $P(c_1^1, \dots, c_{n+1}^1), \dots, P(c_1^m, \dots, c_{n+1}^m)$ be all formulas from Λ in which P occurs. Then the *P-completion axiom for Λ* is the formula

$$\begin{aligned} &\forall x_1, \dots, x_{n+1} P(x_1, \dots, x_{n+1}) \rightarrow \\ &x_1 = c_1^1 \wedge \dots \wedge x_{n+1} = c_{n+1}^1 \vee \dots \vee x_1 = c_1^m \wedge \dots \wedge x_{n+1} = c_{n+1}^m. \end{aligned}$$

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

$$\forall x_1, \dots, x_{n+1} \bullet P(x_1, \dots, x_{n+1}) \rightarrow \text{F}.$$

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

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

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

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

A set $\Theta \subseteq \mathcal{F}\text{orm}(\Sigma)$ is called a *relational theory for R* if $\Theta = \text{RT}(R, \Lambda)$ for some finite set $\Lambda \subseteq \mathcal{F}\text{orm}(\Sigma)$ 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 5.1: *Let $R = (\Sigma, \mathcal{F}\text{orm}(\Sigma))$ be a relational language, and let Γ be a finite subset of $\mathcal{F}\text{orm}(\Sigma)$ such that $\text{RSA}(R) \subseteq \Gamma$. Then it is decidable whether, for a given $A \in \mathcal{F}\text{orm}(\Sigma)$, $\Gamma \Rightarrow A$ is provable.*

Proof: Because $\text{RSA}(R) \subseteq \Gamma$, it is sufficient to consider only structures that are models of $\text{RSA}(R)$. 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 4.1 and 4.2 that it is decidable whether, for a formula $A \in \mathcal{F}\text{orm}(\Sigma)$, $\Gamma \Rightarrow A$ is provable. ■

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 $\text{LPQ}^{\supseteq, \mathcal{F}}$.

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

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

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. This is made precise in the following definition.

Let $R = (\Sigma, \mathcal{F}\text{orm}(\Sigma))$ be a relational language, and let $DB = (R, \Theta, \Xi)$ be a relational database. Then DB is *consistent* iff, for each $A \in \mathcal{F}\text{orm}(\Sigma)$ 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 $CL(\Sigma)$ for all $A' \in \mathcal{F}orm(\Sigma)$. However, the sequent calculus proof system of $LPQ^{\supset, F}(\Sigma)$ rules out such an explosion.

Models of relational theories

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

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

- for all $d_1, d_2 \in \mathcal{U}^{\mathbf{A}}$, $=^{\mathbf{A}}(d_1, d_2) \in \{t, f\}$;
- for all $d \in \mathcal{U}^{\mathbf{A}}$, there exists a $c \in \mathcal{F}unc_0(\Sigma)$ such that $=^{\mathbf{A}}(d, c^{\mathbf{A}}) = t$;
- for all $c_1, c_2 \in \mathcal{F}unc_0(\Sigma)$, $=^{\mathbf{A}}(c_1^{\mathbf{A}}, c_2^{\mathbf{A}}) = t$ only if $c_1 \equiv c_2$.

We have the following lemma.

Lemma 5.2: *Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, and let \mathbf{A} and \mathbf{A}' be relational structures for R . Then there exists a unique bijection β from $\mathcal{U}^{\mathbf{A}}$ to $\mathcal{U}^{\mathbf{A}'}$ such that, for all $c \in \mathcal{F}unc_0(\Sigma)$, $\beta(c^{\mathbf{A}}) = c^{\mathbf{A}'}$.*

Proof: This follows immediately from the definition of a relational structure for R . ■

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, and let Θ be a relational theory for R . It follows immediately from the definition of a relational structure for R that each model of $RSA(R)$ is a relational structure for R . Because $RSA(R) \subseteq \Theta$, each model of Θ is a relational structure for R as well. Despite Θ 's predicate completion axioms, Θ does not have a unique model up to isomorphism. However, identification of t and b in the models of Θ yields uniqueness up to isomorphism.

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

- $\mathcal{U}^{\mathbf{A}'} = \mathcal{U}^{\mathbf{A}}$;
- for each $c \in \mathcal{F}unc_0(\Sigma)$, $c^{\mathbf{A}'} = c^{\mathbf{A}}$;
- for each $n \in \mathbb{N}$, for each $P \in \mathcal{P}red_{n+1}(\Sigma)$, for each $d_1, \dots, d_{n+1} \in \mathcal{U}^{\mathbf{A}'}$,

$$P^{\mathbf{A}'}(d_1, \dots, d_{n+1}) = \begin{cases} t & \text{if } P^{\mathbf{A}}(d_1, \dots, d_{n+1}) \in \{t, b\} \\ f & \text{otherwise;} \end{cases}$$

- for each $d_1, d_2 \in \mathcal{U}^{\mathbf{A}'}$, $=^{\mathbf{A}'}(d_1, d_2) = =^{\mathbf{A}}(d_1, d_2)$.

We have the following lemma.

Lemma 5.3: *Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, let Θ be a relational theory for R , and let \mathbf{A} be a relational structure for R . Then \mathbf{A} is a model of Θ only if $\nabla \mathbf{A}$ is a model of Θ .*

Proof: This follows immediately from the definition of ∇ . ■

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ 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 \mathbf{t} and \mathbf{b} identified, is in essence a relational database as originally introduced by Codd (1970).

Theorem 5.4: *Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ 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 for R .*

Proof: By Lemma 5.3, $\nabla \mathbf{A}$ and $\nabla \mathbf{A}'$ are also models of Θ . Moreover, because $RSA(R) \subseteq \Theta$, $\nabla \mathbf{A}$ and $\nabla \mathbf{A}'$ are relational structures for R . It follows, by Lemma 5.2, that there exists a unique bijection β from $\mathcal{U}^{\nabla \mathbf{A}}$ to $\mathcal{U}^{\nabla \mathbf{A}'}$ such that, for all $c \in \mathcal{F}unc_0(\Sigma)$, $\beta(c^{\nabla \mathbf{A}}) = c^{\nabla \mathbf{A}'}$. Let $P \in \mathcal{P}red_{n+1}(\Sigma)$ ($n \in \mathbb{N}$). Then, because Θ includes the P -completion axiom for the set of all atomic facts for R included in Θ , $P^{\nabla \mathbf{A}}$ and $P^{\nabla \mathbf{A}'}$ are the same up to the above-mentioned bijection. Hence, $\nabla \mathbf{A}$ and $\nabla \mathbf{A}'$ are isomorphic relational structures for R . ■

Theorem 5.5: *Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ 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: Let Λ be the set of all atomic facts for R of which \mathbf{A} is a model. Then it is easy to see that, for all $P \in \bigcup \{\mathcal{P}red_{n+1}(\Sigma) \mid n \in \mathbb{N}\}$, \mathbf{A} is a model of the P -completion axiom for Λ . Moreover, it follows immediately from the definition of a relational structure for R that \mathbf{A} is a model of $RSA(R)$. Hence, \mathbf{A} is a model of the relational theory $RT(R, \Lambda)$ of R . ■

6. Query answering viewed through $LPQ^{\supset, F}$

In this section, queries applicable to a relational database and their answers are considered from the perspective of $LPQ^{\supset, F}$. As a matter of fact, the queries introduced below are closely related to the relational-calculus-oriented queries originally introduced by Codd (1972).

Queries

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

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

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

Let $DB = (R, \Theta, \mathcal{E})$ 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 $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, let $DB = (R, \Theta, \Xi)$ be a relational database, and let $(x_1, \dots, x_n) \bullet A$ be a query that is applicable to DB . Then an *answer to $(x_1, \dots, x_n) \bullet A$ with respect to DB* is a $(c_1, \dots, c_n) \in \mathcal{F}unc_0(\Sigma)^n$ for which $\Theta \Rightarrow [x_1 := c_1] \dots [x_n := c_n]A$ 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 atomic facts and negations of atomic facts that are logical consequences of that relational theory and the relevant relational structure axioms;
- the idea that in the case of a consistent answer there must be such a set of formulas that does not contain an atomic fact or a negation of an atomic fact that causes the database to be inconsistent.

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language. Then a *semi-atomic fact for R* is a formula from $\mathcal{F}orm(\Sigma)$ of the form $P(c_1, \dots, c_{n+1})$ or the form $\neg P(c_1, \dots, c_{n+1})$, where $P \in \mathcal{P}red_{n+1}(\Sigma)$ and $c_1, \dots, c_{n+1} \in \mathcal{F}unc_0(\Sigma)$.

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, let $DB = (R, \Theta, \Xi)$ be a relational database, and let $(x_1, \dots, x_n) \bullet A$ be a query that is applicable to DB . Then a *consistent answer to $(x_1, \dots, x_n) \bullet A$ with respect to DB* is a $(c_1, \dots, c_n) \in \mathcal{F}unc_0(\Sigma)^n$ 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;
- $\Phi, RSA(R) \Rightarrow [x_1 := c_1] \dots [x_n := c_n]A$ 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 Bry (1997). It simply accepts that a database is inconsistent and excludes the source or sources of the inconsistency from being used in computing consistent query answers.

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 all minimally repaired versions of the original database.

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, and let $\Lambda \subseteq \mathcal{F}orm(\Sigma)$ be a finite set of atomic facts for R . Then, following Arenas et al. (1999), the binary relation \leq_Λ 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 $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, let $\Lambda \subseteq \mathcal{F}orm(\Sigma)$ be a finite set of atomic facts for R , and let Ξ be a finite subset of $\mathcal{F}orm(\Sigma)$. Then Λ is consistent with Ξ iff for all semi-atomic facts A for R , $RT(R, \Lambda) \Rightarrow A$ is provable only if $RT(R, \Lambda), \Xi \Rightarrow \circ A$ is provable. We write $Con(\Xi)$ for the set of all finite sets of atomic facts for R that are consistent with Ξ .

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, let $\Lambda \subseteq \mathcal{F}orm(\Sigma)$ be a finite set of atomic facts for R , let $DB = (R, RT(R, \Lambda), \Xi)$ be a relational database, and let $(x_1, \dots, x_n) \bullet A$ be a query that is applicable to DB . Then a *strongly consistent answer* to $(x_1, \dots, x_n) \bullet A$ with respect to DB is a $(c_1, \dots, c_n) \in Func_0(\Sigma)^n$ such that, for each Λ' that is \leq_{Λ} -minimal in $Con(\Xi)$, $RT(R, \Lambda') \Rightarrow [x_1 := c_1] \dots [x_n := c_n]A$ is provable. The elements of $Con(\Xi)$ that are \leq_{Λ} -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 Arenas et al. (1999). It represents, presumably, the first view on what the repairs of an inconsistent database are. Other views have been taken in e.g. Lopatenko and Bertossi (2006), Greco and Molinaro (2011), ten Cate et al. (2015), Calautti et al. (2018), and Bertossi (2019).

Decidability

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

Theorem 6.1: *Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, let $DB = (R, \Theta, \Xi)$ be a relational database, and let $(x_1, \dots, x_n) \bullet A$ be a query applicable to DB . Then it is decidable whether, for a given $(c_1, \dots, c_n) \in Func_0(\Sigma)^n$:*

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

Proof: Each of these decidability results follows immediately from Theorem 5.1 and the definition of the kind of answer concerned. ■

As a corollary of Theorem 6.1, 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 $LPQ^{\supset, F}$ 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 $\forall x \bullet \neg(P(x) \wedge Q(x))$. 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 but 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 but do not cause the database to be inconsistent are $\{P(a), \neg P(c), [2]Q(a), \neg Q(b), Q(c)\}$ 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 addition 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 ternary 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 $\forall x, y, z, y', z' \cdot (P(x, y, z) \wedge P(x, y', z')) \rightarrow [2]y = y'$. Moreover, consider the query $y \cdot \exists x, z \cdot P(x, y, z)$. Clearly, the set of answers is $\{b, c, f\}$.

The sets of semi-atomic formulas that are logical consequences of the relational theory but 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:

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

Because a, d, e , and g are not 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:

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

Because a, d, e , and g are not 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 computing consistent query answers. 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 atomic facts in a database that causes inconsistency with its integrity constraints suspect and consequently not to use them in computing answers to a query with respect to the database. The plain choice not to use the source or sources of inconsistency in computing consistent query answers 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, 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 Bry (1997). 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 $LPQ^{\supset,F}$. What is missing in Bry (1997) 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 Section 6 is essentially the same as the definition of a consistent answer to a query with respect to a database given in Arenas et al. (1999). 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 all minimally repaired versions 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. Lopatenko and Bertossi (2006), Greco and Molinaro (2011), ten Cate et al. (2015), Calautti et al. (2018), and Bertossi (2019).

The view taken in Arenas et al. (1999) is that only deletions and additions of atomic facts may be made to the original database and that the extent to which a repaired version differs from the original database is the symmetric difference of the set of atomic facts in the repaired version and the set of atomic facts in the original database. Most other views differ from this view by considering other allowed kinds of changes. For

example, the case that only deletions of atomic facts may be made and the case that only additions of atomic facts may be made are considered in ten Cate et al. (2015) and the case that only alterations of atomic facts may be made is considered in Greco and Molinaro (2011). The view taken in Lopatenko and Bertossi (2006) differs from the view taken in Arenas et al. (1999) by taking the cardinality of the above-mentioned symmetric difference as the extent to which a repaired version differs from the original database. It is easy to adapt the definition of a strongly consistent answer given in Section 6 to the above-mentioned views. The operational view of repairs taken in Calautti et al. (2018) seems incommensurable with those views. It is unclear whether the definition of a strongly consistent answer given in Section 6 can be adapted to this operational view without much difficulty.

In Bry (1997), 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 a consistent part of the database. From the viewpoint taken in Arenas et al. (1999), 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 Chomicki (2004). 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 Reiter (1984) on what is a relational database, a query applicable to a relational database, and an answer to a query with respect to a consistent relational database;
- the view of Bry (1997) on what is a consistent answer to a query with respect to an inconsistent relational database;
- the view of Arenas et al. (1999) on what is a consistent answer to a query with respect to an inconsistent relational database.

The view of Reiter has been combined with the view of Bry as well as with the view of Arenas et al. and adapted to the setting of the paraconsistent logic $LPQ^{\supset, F}$. 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 generalised by allowing its basis to be a set of Horn clauses and adapting the completion axioms as sketched in Gallaire et al. (1984). This generalisation gives rise to a generalisation 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 generalisation of the notion of a relational database. Further generalisation of the notion of an indefinite deductive database is a different matter.

Note

1. If we replace the inference rule \neg -R by the inference rule \neg -L in the sequent calculus proof system of $\text{LPQ}^{\supset, F}(\Sigma)$, then we obtain a sound and complete proof system of the paracomplete analogue of $\text{LPQ}^{\supset, F}$. The propositional part of that logic ($\text{K3}^{\supset, F}$) is studied in e.g. Middelburg (2021).

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

C. A. Middelburg  <http://orcid.org/0000-0002-8725-0197>

References

- Arenas, M., Bertossi, L., & Chomicki, J. (1999). Consistent query answers in inconsistent databases. In *PODS 1999* (pp. 68–79). ACM Press. <https://doi.org/10.1145/303976.303983>
- Bagai, R., & Sunderraman, R. (1995). A paraconsistent relational data model. *International Journal of Computer Mathematics*, 55(1–2), 39–55. <https://doi.org/10.1080/00207169508804361>
- Bertossi, L. (2019). Database repairs and consistent query answering: Origins and further developments. In *PODS 2019* (pp. 48–58). ACM Press. <https://doi.org/10.1145/3294052.3322190>
- Bry, F. (1997). Query answering in information systems with integrity constraints. In S. Jajodia, W. List, G. McGregor, & L. Strous (Eds.), *IICIS 1997* (pp. 113–130). Chapman and Hall. https://doi.org/10.1007/978-0-387-35317-3_6
- Calautti, M., Libkin, L., & Pieris, A. (2018). An operational approach to consistent query answering. In *PODS 2018* (pp. 239–251). ACM Press. <https://doi.org/10.1145/3196959.3196966>
- Chomicki, J. (2004). Consistent query answering. In S. Jajodia & L. Strous (Eds.), *IICIS 2003* (Vol. 140, pp. 219–239). Springer-Verlag. https://doi.org/10.1007/1-4020-7901-X_15
- Codd, E. F. (1970). A relational model for large shared data banks. *Communications of the ACM*, 13(6), 377–387. <https://doi.org/10.1145/362384.362685>
- Codd, E. F. (1972). *Relational completeness of data base sublanguage* (Tech. Rep. No. RJ 987). IBM.
- Colacito, A., de Jongh, D., & Vargas, A. L. (2017). Subminimal negation. *Soft Computing*, 21(1), 165–174. <https://doi.org/10.1007/s00500-016-2391-8>
- D’Ottaviano, I. M. L. (1985). The completeness and compactness of a three-valued first-order logic. *Revista Colombiana De Matemáticas*, 19(1–2), 77–94.
- Gallaire, H., Minker, J., & Nicolas, J. M. (1984). Logic and databases: A deductive approach. *Computing Surveys*, 16(2), 153–185. <https://doi.org/10.1145/356924.356929>
- Greco, S., & Molinaro, C. (2011). Consistent query answering over inconsistent databases. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 15(3), 119–129. <https://doi.org/10.3233/KES-2011-0216>
- Lopatenko, A., & Bertossi, L. (2006). Consistent query answering by minimal-size repairs. In *DEXA 2006* (pp. 558–562). IEEE. <https://doi.org/10.1109/DEXA.2006.43>
- Middelburg, C. A. (2020). *A classical-logic view of a paraconsistent logic*. arXiv:2008.07292v6 [cs.LG].
- Middelburg, C. A. (2021). On the strongest three-valued paraconsistent logic contained in classical logic and its dual. *Journal of Logic and Computation*, 31(2), 597–611. <https://doi.org/10.1093/logcom/exaa084>
- Negri, S., & von Plato, J. (2001). *Structural proof theory*. Cambridge University Press.
- Odintsov, S. P. (2007). The lattice of extensions of the minimal logic. *Siberian Advances in Mathematics*, 17(2), 112–143. <https://doi.org/10.3103/S1055134407020034>
- Piccolo, L. (2020). Truth in a logic of formal inconsistency: How classical can it get? *Logic Journal of the IGPL*, 28(5), 771–806. <https://doi.org/10.1093/jigpal/jzy059>

- Priest, G. (1979). The logic of paradox. *Journal of Philosophical Logic*, 8(1), 219–241. <https://doi.org/10.1007/BF00258428>
- Reiter, R. (1984). Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, & J. W. Schmidt (Eds.), *On conceptual modelling* (pp. 191–238). Springer-Verlag. https://doi.org/10.1007/978-1-4612-5196-5_8
- Takeuti, G. (1975). *Proof theory*. North-Holland.
- ten Cate, B., Fontaine, G., & Kolaitis, P. G. (2015). On the data complexity of consistent query answering. *Theory of Computing Systems*, 57(4), 843–891. <https://doi.org/10.1007/s00224-014-9586-0>
- Vardi, M. Y. (1986). Querying logical databases. *Journal of Computer and System Sciences*, 33(2), 142–160. [https://doi.org/10.1016/0022-0000\(86\)90016-4](https://doi.org/10.1016/0022-0000(86)90016-4)