Belnap-Dunn Logic and Query Answering in Inconsistent Databases with Null Values

Middelburg, C.A.

DOI
10.7561/SACS.2023.2.159

Publication date
2023

Document Version
Final published version

Published in
Scientific Annals of Computer Science

Scientific Annals of Computer Science vol. 33 (2), 2023, pp. 159–192
doi: 10.7561/SACS.2023.2.159

Belnap-Dunn Logic and Query Answering in Inconsistent Databases with Null Values

C.A. Middelburg[1]

Abstract

This paper concerns an expansion of first-order Belnap-Dunn logic, called $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$, and an application of this logic in the area of relational database theory. The notion of a relational database, the notion of a query applicable to a relational database, and several notions of an answer to a query with respect to a relational database are considered from the perspective of this logic, taking into account that a database may be an inconsistent database and/or a database with null values. The chosen perspective enables among other things the definition of a notion of a consistent answer to a query with respect to a possibly inconsistent database without resort to database repairs. For each of the notions of an answer considered, being an answer to a query with respect to a database of the kind considered is decidable.

Keywords: relational database, inconsistent database, null value, consistent query answering, Belnap-Dunn logic, indeterminate value.

1 Introduction

In the area of relational database theory, it is quite common since the 1980s to take the view that a database is a set of formulas of first-order classical logic, a query is a formula of first-order classical logic, and query answering amounts to proving that a formula is a logical consequence of a set of formulas

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

in first-order classical logic (see e.g. [11, 26]). The use of first-order classical logic makes it difficult to take into account the possibility that a database is an inconsistent database and the possibility that a database is a database with null values.

In the logical view of relational databases, an inconsistent relational database is an inconsistent set of formulas in the sense that there exists a formula such that both that formula and its negation are logical consequences of the set of formulas. Because in classical logic every formula is a logical consequence of each two formulas of which one is the negation of the other, it becomes difficult to take the possibility that a database is inconsistent properly into account. A logic in which not every formula is a logical consequence of each two formulas of which one is the negation of the other is called a paraconsistent logic.

The view on null values in relational databases taken in this paper can be described as follows: (a) in relational databases with null values, a single dummy value, called the null value, is used for values that are indeterminate, (b) a value that is indeterminate is a value that is either unknown or nonexistent, and (c) independent of whether it is unknown or nonexistent, no meaningful answer can be given to the question whether the null value and whatever value, including the null value itself, are the same. Several variations on this view on null values in relational databases have been studied (see e.g. [10, 14, 30]). However, those variations are less basic and most of them give rise to similar problems in a logical view of relational databases with null values.

In the logical view of relational databases, a relational database with null values is an incomplete set of formulas in the sense that there exists a formula such that neither that formula nor its negation is a logical consequence of the set of formulas. Because in classical logic, for each two formulas of which one is the negation of the other, one or the other is a logical consequence of every set of formulas, it becomes difficult to take the possibility that null values occur in a database properly into account. A logic in which not, for each two formulas of which one is the negation of the other, one or the other is a logical consequence of every set of formulas is called a paracomplete logic.

In [21], the notion of a relational database, the notion of a query applicable to a relational database, and several notions of an answer to a query with respect to a relational database are considered from the perspective of $\text{LPQ}^{\supset,\mathsf{F}}$, an expansion of the first-order version of Priest's logic of paradox known as LPQ [25]. The possibility that a database is an inconsistent database is taken into account, but the possibility that a database is a database with null values is not taken into account. The reason for this is that $\mathrm{LPQ}^{\supset,\mathsf{F}}$ is paraconsistent, but not paracomplete. There are many other paraconsistent logics. The choice of $\mathrm{LPQ}^{\supset,\mathsf{F}}$ stems from the fact that its connectives and quantifiers are all familiar from classical logic and its logical consequence relation is very closely connected to the one of classical logic.

In [22], $\mathrm{BD}^{\supset,\mathsf{F}}$, an expansion of first-order Belnap-Dunn logic [1] with classical connectives, is introduced and studied with the emphasis on (a) the connection between the logical consequence relations of $\mathrm{BD}^{\supset,\mathsf{F}}$ and the version of classical logic with the same connectives and quantifiers and (b) the definability in $\mathrm{BD}^{\supset,\mathsf{F}}$ of interesting non-classical connectives added to Belnap-Dunn logic in its expansions that have been studied earlier. $\mathrm{BD}^{\supset,\mathsf{F}}$ is both paraconsistent and paracomplete. Like for $\mathrm{LPQ}^{\supset,\mathsf{F}}$, it holds for $\mathrm{BD}^{\supset,\mathsf{F}}$ that its connectives and quantifiers are all familiar from classical logic and its logical consequence relation is very closely connected to the one of classical logic.

An appendix of [22] goes briefly into an minor variation of $\mathrm{BD}^{\supset,\mathsf{F}}$ that covers terms with an indeterminate value. This variation, called $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$, is also paraconsistent and paracomplete. Moreover, its treatment of equations corresponds to the above-mentioned thought that no meaningful answer can be given to the question whether the null value and whatever value, including the null value itself, are the same value. All this means that both the possibility that a database is an inconsistent database and the possibility that a database is a database with null values can be properly taken into account when the notions considered in [21] are considered from the perspective of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$.

This is taken up in the current paper. In order to make this paper self-contained, the language and logical consequence relation of $\mathrm{BD}^{\supset,\mathsf{F}}$ as well as a sequent calculus proof system for $\mathrm{BD}^{\supset,\mathsf{F}}$ are presented. $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$ is introduced as a minor variation of $\mathrm{BD}^{\supset,\mathsf{F}}$. The notion of a relational database, the notion of a query applicable to a relational database, and several notions of an answer to a query with respect to a relational database are defined in the setting of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$, taking into account that a database may be an inconsistent database and/or a database with null values. Like in [21], the definitions concerned are based on those given in [26]. Two notions of a consistent answer to a query with respect to a possibly inconsistent relational database are introduced. Like in [21], one of them is reminiscent of the notion of a consistent answer from [6] and the other is reminiscent of the notion of a consistent answer from [2].

The structure of this paper is as follows. In Sections 2, 3, and 4, the language of $\mathrm{BD}^{\supset,\mathsf{F}}$, the logical consequence relation of $\mathrm{BD}^{\supset,\mathsf{F}}$, and a sequent calculus proof system for $\mathrm{BD}^{\supset,\mathsf{F}}$ are presented. In Section 5, $\mathrm{BD}^{\supset,\mathsf{F}}_{\bot}$ is introduced as a minor variation of $\mathrm{BD}^{\supset,\mathsf{F}}$. In Sections 6 and 7, relational databases and query answering with respect to a database that may be an inconsistent database and/or a database with null values are considered from the perspective of $\mathrm{BD}^{\supset,\mathsf{F}}$. In Section 8, some variations of the view on null values in relational databases taken in this paper are briefly discussed. In Section 9, some concluding remarks are made. Parts of the sections in which $\mathrm{BD}^{\supset,\mathsf{F}}$ is introduced overlap with parts of [22].

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

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

Signatures and Alphabets

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

The symbols from $\bigcup\{\mathcal{F}unc_n \mid n \in \mathbb{N}\} \cup \bigcup\{\mathcal{P}red_n \mid n \in \mathbb{N}\}$ are known as *non-logical symbols*. 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 $\bigcup\{\mathcal{F}unc_n \mid n \in \mathbb{N}\} \cup \bigcup\{\mathcal{P}red_n \mid n \in \mathbb{N}\}$. We write $\mathcal{F}unc_n(\Sigma)$ and $\mathcal{P}red_n(\Sigma)$, where Σ is a signature and $n \in \mathbb{N}$, for the sets $\Sigma \cap \mathcal{F}unc_n$ and $\Sigma \cap \mathcal{P}red_n$, respectively.

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

The *alphabet* of the language of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ consists of the following symbols:

- the variables from $\mathcal{V}ar$;
- the non-logical symbols from Σ ;
- the *logical symbols* of $\mathrm{BD}^{\supset,\mathsf{F}}$, to wit:
  - the *equality symbol* $=$;
  - the *falsity connective* F ;
  - the *negation connective* $\neg$;
  - the *conjunction connective* $\wedge$;
  - the *disjunction connective* $\vee$;
  - the *implication connective* $\supset$;
  - the *universal quantifier* $\forall$;
  - the *existential quantifier* $\exists$.

Terms and Formulas

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

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

1. if $x \in \mathcal{V}ar$, then $x \in \mathcal{T}erm(\Sigma)$;
2. if $c \in \mathcal{F}unc_0(\Sigma)$, then $c \in \mathcal{T}erm(\Sigma)$;
3. if $f \in \mathcal{F}unc_{n+1}(\Sigma)$ and $t_1, \ldots, t_{n+1} \in \mathcal{T}erm(\Sigma)$, then $f(t_1, \ldots, t_{n+1}) \in \mathcal{T}erm(\Sigma)$.

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

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

1. if $p \in \mathcal{P}red_0(\Sigma)$, then $p \in \mathcal{F}orm(\Sigma)$;

2. if $P \in \mathcal{P}red_{n+1}(\Sigma)$ and $t_1, \ldots, t_{n+1} \in \mathcal{T}erm(\Sigma)$, then $P(t_1, \ldots, t_{n+1}) \in \mathcal{F}orm(\Sigma)$;
3. if $t_1, t_2 \in \mathcal{T}erm(\Sigma)$, then $t_1 = t_2 \in \mathcal{F}orm(\Sigma)$;
4. $\mathsf{F} \in \mathcal{F}orm(\Sigma)$;
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 $A_1 \land A_2$, $A_1 \lor A_2$, $A_1 \supset A_2 \in \mathcal{F}orm(\Sigma)$;
7. if $x \in \mathcal{V}ar$ and $A \in \mathcal{F}orm(\Sigma)$, then $\forall x \bullet A$, $\exists x \bullet A \in \mathcal{F}orm(\Sigma)$.

The set $\mathcal{A}tom(\Sigma)$ of all *atomic formulas of* $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is the subset of $\mathcal{F}orm(\Sigma)$ can be formed by applying formation rules 1–3 only.

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.

In the coming sections, we will write $\mathrm{CL}^{\supset,\mathsf{F}}$ for the version of classical logic that has the same language as $\mathrm{BD}^{\supset,\mathsf{F}}$.

Notational Conventions

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 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 $\neg$, $\land$, $\lor$, $\supset$. 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 $\forall x_1 \bullet \cdots \forall x_n \bullet A$ and $\exists x_1 \bullet \cdots \exists x_n \bullet A$ are usually written as $\forall x_1, \ldots, x_n \bullet A$ and $\exists x_1, \ldots, x_n \bullet A$, respectively.

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

In this section, the truth value of formulas of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ and the logical consequence relation on sets of formulas of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ will be defined. This will be done using the logical matrix of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$.

First, the logical matrix of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is defined. Next, the truth value of formulas of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ and the logical consequence relation on sets of formulas of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ are defined. The truth value of formulas is defined with respect to (a) a structure consisting of a set of values and an interpretation of each non-logical symbol and the equality symbol and (b) an assignment of values from that set of values to the variables. Structures and assignments are introduced before the definition in question.

Matrix

The interpretation of the logical symbols of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, with the exception of the equality symbol, is given by means of a logical matrix.

In the definition of this matrix, t (*true*), f (*false*), b (*both true and false*), and n (*neither true nor false*) are taken as truth values. Moreover, use is made of the partial order $\leq$ on the set $\{\mathsf{t},\mathsf{f},\mathsf{b},\mathsf{n}\}$ in which f is the least element, t is the greatest element, and b and n are incomparable. We write $\inf V$ and $\sup V$, where $V \subseteq \{\mathsf{t},\mathsf{f},\mathsf{b},\mathsf{n}\}$, for the greatest lower bound and least upper bound, respectively, of V with respect to $\leq$.

The *matrix* of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is the triple $\langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$, where:

- $\mathcal{V} = \{\mathsf{t},\mathsf{f},\mathsf{b},\mathsf{n}\}$;
- $\mathcal{D} = \{\mathsf{t},\mathsf{b}\}$;
- $\mathcal{O}$ consists of the following constant and functions:

$$
\begin{array}{lll}
\widetilde{\mathsf{F}} \in \mathcal{V} \quad \text{defined by} & \widetilde{\mathsf{F}} & = \; \mathsf{f}\,, \\
\widetilde{\neg} : \mathcal{V} \to \mathcal{V} \quad \text{defined by} & \widetilde{\neg}(a) & = \begin{cases} \mathsf{t} & \text{if } a = \mathsf{f} \\ \mathsf{f} & \text{if } a = \mathsf{t} \\ a & \text{otherwise}\,, \end{cases}
\end{array}
$$

$$
\begin{array}{llrcl}
\widetilde{\wedge} : \mathcal{V} \times \mathcal{V} \to \mathcal{V} & \text{defined by} & \widetilde{\wedge}(a_1, a_2) & = & \inf\{a_1, a_2\}\ , \\
\widetilde{\vee} : \mathcal{V} \times \mathcal{V} \to \mathcal{V} & \text{defined by} & \widetilde{\vee}(a_1, a_2) & = & \sup\{a_1, a_2\}\ , \\
\widetilde{\supset} : \mathcal{V} \times \mathcal{V} \to \mathcal{V} & \text{defined by} & \widetilde{\supset}(a_1, a_2) & = & \begin{cases} \mathsf{t} & \text{if } a_1 \notin \{\mathsf{t}, \mathsf{b}\} \\ a_2 & \text{otherwise}\ , \end{cases} \\
\widetilde{\forall} : \mathcal{P}(\mathcal{V}) \setminus \{\emptyset\} \to \mathcal{V} & \text{defined by} & \widetilde{\forall}(V) & = & \inf V\ ,^{2} \\
\widetilde{\exists} : \mathcal{P}(\mathcal{V}) \setminus \{\emptyset\} \to \mathcal{V} & \text{defined by} & \widetilde{\exists}(V) & = & \sup V\ ,
\end{array}
$$

where a, a_1, and a_2 range over all truth values from $\mathcal{V}$ and V ranges over all non-empty subsets of $\mathcal{V}$.

$\mathcal{V}$ is the set of *truth values* of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, $\mathcal{D}$ is the set of *designated truth values* of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, and $\widetilde{\mathsf{F}}$, $\widetilde{\neg}$, $\widetilde{\wedge}$, $\widetilde{\vee}$, $\widetilde{\supset}$, $\widetilde{\forall}$, and $\widetilde{\exists}$ are the *truth functions* that are the interpretations of the logical symbols F, $\neg$, $\wedge$, $\vee$, $\supset$, $\forall$, and $\exists$, respectively. The idea behind the designated truth values is that a formula is valid if its truth value with respect to all structures and assignments in those structures (both defined below) is a designated truth value.

The set of *non-designated truth values* of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, written $\overline{\mathcal{D}}$, is the set $\mathcal{V} \setminus \mathcal{D}$.

Structures

The interpretation of the non-logical symbols of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ and the equality symbol is given by means of structures.

A structure $\mathbf{A}$ of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is a pair $\langle \mathcal{U}^{\mathbf{A}}, \mathcal{I}^{\mathbf{A}} \rangle$, where:

- $\mathcal{U}^{\mathbf{A}}$ is a set, called the *domain of* $\mathbf{A}$, such that $\mathcal{U}^{\mathbf{A}} \neq \emptyset$ and $\mathcal{U}^{\mathbf{A}} \cap \mathcal{V} = \emptyset$;
- $\mathcal{I}^{\mathbf{A}}$ consists of:
  - an element $c^{\mathbf{A}} \in \mathcal{U}^{\mathbf{A}}$ for every $c \in \mathcal{F}unc_0(\Sigma)$;
  - a function $f^{\mathbf{A}} : \overbrace{\mathcal{U}^{\mathbf{A}} \times \cdots \times \mathcal{U}^{\mathbf{A}}}^{n+1 \text{ times}} \to \mathcal{U}^{\mathbf{A}}$ for every $f \in \mathcal{F}unc_{n+1}(\Sigma)$ and $n \in \mathbb{N}$;
  - an element $p^{\mathbf{A}} \in \mathcal{V}$ for every $p \in \mathcal{P}red_0(\Sigma)$;
  - a function $P^{\mathbf{A}} : \overbrace{\mathcal{U}^{\mathbf{A}} \times \cdots \times \mathcal{U}^{\mathbf{A}}}^{n+1 \text{ times}} \to \mathcal{V}$ for every $P \in \mathcal{P}red_{n+1}(\Sigma)$ and $n \in \mathbb{N}$;

[2]We write $\mathcal{P}(S)$, where S is a set, for the powerset of S.

– a function $=^{\mathbf{A}} : \mathcal{U}^{\mathbf{A}} \times \mathcal{U}^{\mathbf{A}} \to \mathcal{V}$ where, for all $d_1, d_2 \in \mathcal{U}^{\mathbf{A}}$, $=^{\mathbf{A}}(d_1, d_2) \in \mathcal{D}$ iff $d_1 = d_2$.

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

The interpretation of the variables of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is given by means of assignments.

Let $\mathbf{A}$ be a structure of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$. Then an *assignment in* $\mathbf{A}$ is a function $\alpha : \mathcal{V}ar \to \mathcal{U}^{\mathbf{A}}$. For every assignment α in $\mathbf{A}$, variable $x \in \mathcal{V}ar$, 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

Let $\mathbf{A}$ be a structure of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, and let α be an assignment in $\mathbf{A}$. Then the *valuation of* $\mathcal{T}erm(\Sigma)$ *in structure* $\mathbf{A}$ *under assignment* α is the function $tval^{\mathbf{A}}_{\alpha} : \mathcal{T}erm(\Sigma) \to \mathcal{U}^{\mathbf{A}}$ that maps each term t to the element of $\mathcal{U}^{\mathbf{A}}$ that is the value of t in $\mathbf{A}$ under assignment α. Similarly, the *valuation of* $\mathcal{F}orm(\Sigma)$ *in structure* $\mathbf{A}$ *under assignment* α is the function $fval^{\mathbf{A}}_{\alpha} : \mathcal{F}orm(\Sigma) \to \mathcal{V}$ that maps each formula A to the element of $\mathcal{V}$ 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 $tval^{\mathbf{A}}_{\alpha}(t)$ and $fval^{\mathbf{A}}_{\alpha}(A)$, respectively.

These valuation functions are inductively defined in Table 1. In this table, x is a meta-variable ranging over all variables from $\mathcal{V}ar$, c is a meta-variable ranging over all function symbols from $\mathcal{F}unc_0(\Sigma)$, f is a meta-variable ranging over all function symbols from $\mathcal{F}unc_{n+1}(\Sigma)$, p is a meta-variable ranging over all predicate symbols from $\mathcal{P}red_0(\Sigma)$, P is a meta-variable ranging over all predicate symbols from $\mathcal{P}red_{n+1}(\Sigma)$, $t_1, \ldots, t_{n+1}$ are meta-variables ranging over all terms from $\mathcal{T}erm(\Sigma)$, and A, A_1, and A_2 are meta-variables ranging over all formulas from $\mathcal{F}orm(\Sigma)$.

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

Proposition 1 *Let* $\mathbf{A}$ *be a structure of* $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ *such that* $\mathcal{U}^{\mathbf{A}}$ *is finite, and let* α *be an assignment in* $\mathbf{A}$*. Then, it is decidable whether, for a formula* $A \in \mathcal{F}orm(\Sigma)$*,* $[\![A]\!]^{\mathbf{A}}_{\alpha} \in \mathcal{D}$*.*

Table 1: Valuations of terms and formulas of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$

$$
\begin{aligned}
[\![x]\!]^{\mathbf{A}}_{\alpha} &= \alpha(x)\ , \\
[\![c]\!]^{\mathbf{A}}_{\alpha} &= c^{\mathbf{A}}\ , \\
[\![f(t_1,\ldots,t_{n+1})]\!]^{\mathbf{A}}_{\alpha} &= f^{\mathbf{A}}([\![t_1]\!]^{\mathbf{A}}_{\alpha},\ldots,[\![t_{n+1}]\!]^{\mathbf{A}}_{\alpha}) \\
[\![p]\!]^{\mathbf{A}}_{\alpha} &= p^{\mathbf{A}}\ , \\
[\![P(t_1,\ldots,t_{n+1})]\!]^{\mathbf{A}}_{\alpha} &= P^{\mathbf{A}}([\![t_1]\!]^{\mathbf{A}}_{\alpha},\ldots,[\![t_{n+1}]\!]^{\mathbf{A}}_{\alpha})\ , \\
[\![t_1 = t_2]\!]^{\mathbf{A}}_{\alpha} &= =^{\mathbf{A}}([\![t_1]\!]^{\mathbf{A}}_{\alpha},[\![t_2]\!]^{\mathbf{A}}_{\alpha})\ , \\
[\![\mathsf{F}]\!]^{\mathbf{A}}_{\alpha} &= \widetilde{\mathsf{F}}\ , \\
[\![\neg A]\!]^{\mathbf{A}}_{\alpha} &= \widetilde{\neg}([\![A]\!]^{\mathbf{A}}_{\alpha})\ , \\
[\![A_1 \wedge A_2]\!]^{\mathbf{A}}_{\alpha} &= \widetilde{\wedge}([\![A_1]\!]^{\mathbf{A}}_{\alpha},[\![A_2]\!]^{\mathbf{A}}_{\alpha})\ , \\
[\![A_1 \vee A_2]\!]^{\mathbf{A}}_{\alpha} &= \widetilde{\vee}([\![A_1]\!]^{\mathbf{A}}_{\alpha},[\![A_2]\!]^{\mathbf{A}}_{\alpha})\ , \\
[\![A_1 \supset A_2]\!]^{\mathbf{A}}_{\alpha} &= \widetilde{\supset}([\![A_1]\!]^{\mathbf{A}}_{\alpha},[\![A_2]\!]^{\mathbf{A}}_{\alpha})\ , \\
[\![\forall x \bullet A]\!]^{\mathbf{A}}_{\alpha} &= \widetilde{\forall}(\{[\![A]\!]^{\mathbf{A}}_{\alpha(x \to d)} \mid d \in \mathcal{U}^{\mathbf{A}}\})\ , \\
[\![\exists x \bullet A]\!]^{\mathbf{A}}_{\alpha} &= \widetilde{\exists}(\{[\![A]\!]^{\mathbf{A}}_{\alpha(x \to d)} \mid d \in \mathcal{U}^{\mathbf{A}}\})\ ,
\end{aligned}
$$

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

Below, the notion of a model of a set of formulas of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is defined in terms of valuations.

Let Γ be a set of formulas from $\mathcal{F}\!orm(\Sigma)$, and let $\mathbf{A}$ be a structure of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$. Then $\mathbf{A}$ *is a model of* Γ iff for all assignments α in $\mathbf{A}$, for all $A \in \Gamma$, $[\![A]\!]^{\mathbf{A}}_{\alpha} \in \mathcal{D}$.

Logical Consequence

Given the valuations of terms and formulas of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, it is easy to make precise when in $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ a set of formulas is a logical consequence of another set of formulas.

Let Γ and Δ be sets of formulas from $\mathcal{F}\!orm(\Sigma)$. Then Δ *is a logical consequence of* Γ, written $\Gamma \vDash \Delta$, iff for all structures $\mathbf{A}$ of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, for all assignments α in $\mathbf{A}$, if $[\![A]\!]^{\mathbf{A}}_{\alpha} \in \mathcal{D}$ for all $A \in \Gamma$, then $[\![A']\!]^{\mathbf{A}}_{\alpha} \in \mathcal{D}$ for some

$A' \in \Delta$. We write $\Gamma \nvDash \Delta$ to indicate that it is not the case that $\Gamma \vDash \Delta$.

The two properties concerning the logical consequence relation of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ mentioned below are proved in [22].

The logical consequence relation $\vDash$ of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is such that

$$\Gamma \vDash \Delta, A_1 \supset A_2 \;\text{ iff }\; A_1, \Gamma \vDash \Delta, A_2$$

for all $\Gamma, \Delta \subseteq \mathcal{F}orm(\Sigma)$ and $A_1, A_2 \in \mathcal{F}orm(\Sigma)$ and moreover there exists a logic $\mathcal{L}$ with the same language as $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ and a logical consequence relation $\vDash'$ such that:

- $\vDash \subseteq \vDash'$;
- the matrix $\langle \mathcal{V}', \mathcal{D}', \mathcal{O}' \rangle$ of $\mathcal{L}$ is such that $\mathcal{V}' = \{\mathsf{t}, \mathsf{f}\}$, $\mathcal{D}' = \{\mathsf{t}\}$, and the interpretation of $\neg$ in $\mathcal{O}'$ is as follows:

$$\tilde{\neg}'(a) = \begin{cases} \mathsf{t} & \text{if } a = \mathsf{f} \\ \mathsf{f} & \text{if } a = \mathsf{t} \end{cases}$$

  for all $a \in \mathcal{V}'$.

This means that $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is $\neg$-*coherent with classical logic* in the sense of [3].

There exist $A, A' \in \mathcal{F}orm(\Sigma)$ such that $A, \neg A \nvDash A'$. Because $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is also $\neg$-coherent with classical logic, this means that $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is *paraconsistent* in the sense of [3].

There exist a $\Gamma \subseteq \mathcal{F}orm(\Sigma)$ and $A, A' \in \mathcal{F}orm(\Sigma)$ such that $\Gamma, A \vDash A'$ and $\Gamma, \neg A \vDash A'$, but $\Gamma \nvDash A'$. Because $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is also $\neg$-coherent with classical logic, this means that $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ is *paracomplete* in the sense of [3].

Abbreviations

In what follows, the following abbreviations will be used:

$$\begin{array}{rll}
t_1 \neq t_2 \text{ stands for } & \neg(t_1 = t_2), & \\
\mathsf{T} \text{ stands for } & \neg\mathsf{F}, & \\
A_1 \rightarrow A_2 \text{ stands for } & (A_1 \supset A_2) \wedge (\neg A_2 \supset \neg A_1), & \\
\Delta A \text{ stands for } & \neg(A \supset \mathsf{F}) & \text{(designatedness)}, \\
\circ A \text{ stands for } & \neg(\Delta(A \wedge \neg A)) & \text{(consistency)}, \\
\star A \text{ stands for } & \Delta(A \vee \neg A) & \text{(determinacy)}.
\end{array}$$

It follows from the definitions concerned that:

$$[\![\Delta A]\!]_\alpha^{\mathbf{A}} = \begin{cases} \mathsf{t} & \text{if } [\![A]\!]_\alpha^{\mathbf{A}} \in \{\mathsf{t}, \mathsf{b}\} \\ \mathsf{f} & \text{otherwise,} \end{cases}$$

$$[\![\circ A]\!]_\alpha^{\mathbf{A}} = \begin{cases} \mathsf{t} & \text{if } [\![A]\!]_\alpha^{\mathbf{A}} \in \{\mathsf{t}, \mathsf{f}, \mathsf{n}\} \\ \mathsf{f} & \text{otherwise,} \end{cases}$$

$$[\![\star A]\!]_\alpha^{\mathbf{A}} = \begin{cases} \mathsf{t} & \text{if } [\![A]\!]_\alpha^{\mathbf{A}} \in \{\mathsf{t}, \mathsf{f}, \mathsf{b}\} \\ \mathsf{f} & \text{otherwise.} \end{cases}$$

This means that the abbreviations ΔA, $\circ A$, and $\star A$ correspond to formulas of studied expansions of Belnap-Dunn logic whose connectives include Δ, $\circ$ or $\star$. The connective Δ is for example found in the expansion of Belnap-Dunn logic known as BD$\varDelta$ [27]. The connectives $\circ$ and $\star$ have for example been studied in the setting of Belnap-Dunn logic in [8]. The connective $\circ$ is also found in the expansion of the first-order version of Priest's logic of paradox [25] known as LP$^\circ$ [24]. The connective $\star$ is the counterpart of $\circ$ in the setting of Kleene's strong three-valued logic [15, Section 64].

Moreover, notice that:

$$[\![\circ A \wedge \star A]\!]_\alpha^{\mathbf{A}} = \begin{cases} \mathsf{t} & \text{if } [\![A]\!]_\alpha^{\mathbf{A}} \in \{\mathsf{t}, \mathsf{f}\} \\ \mathsf{f} & \text{otherwise.} \end{cases}$$

4 A Proof System for BD$^{\supset,\mathsf{F}}(\Sigma)$

In this section, a sequent calculus proof system for BD$^{\supset,\mathsf{F}}(\Sigma)$ 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 BD$^{\supset,\mathsf{F}}(\Sigma)$ 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. Extensions of the proof system of BD$^{\supset,\mathsf{F}}(\Sigma)$ which can serve as proof systems for closely related logics are also described.

Sequents

In the sequent calculus proof system for BD$^{\supset,\mathsf{F}}(\Sigma)$, a *sequent* is an expression of the form $\Gamma \Rightarrow \Delta$, where Γ and Δ are finite sets of formulas from $\mathcal{F}orm(\Sigma)$. We write Γ, Γ' for $\Gamma \cup \Gamma'$ and A, where A is a formula from $\mathcal{F}orm(\Sigma)$, 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 3 holds between Γ and Δ. 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 for $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ consists of the inference rules given in Table 2. In this table, x and y are meta-variables ranging over all variables from $\mathcal{V}ar$, t, t_1, and t_2 are meta-variables ranging over all terms from $\mathcal{T}erm(\Sigma)$, A, A_1, and A_2 are meta-variables ranging over all formulas from $\mathcal{F}orm(\Sigma)$, and Γ and Δ are meta-variables ranging over all finite sets of formulas from $\mathcal{F}orm(\Sigma)$.

Derivations and Proofs

In the sequent calculus proof system for $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, 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 $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ 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 $\mathcal{F}orm(\Sigma)$. 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.

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

Theorem 1 *Let Γ and Δ be sets of formulas from $\mathcal{F}orm(\Sigma)$. Then $\Gamma \vdash \Delta$ iff $\Gamma \vDash \Delta$.*

Proof: See Appendix A of [22]. □

Table 2: A sequent calculus proof system for $\mathrm{BD}^{\supset,\mathsf{F}}$

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

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

Extensions of the Proof System

The languages of $\mathrm{CL}^{\supset,\mathsf{F}}$ and $\mathrm{BD}^{\supset,\mathsf{F}}$ are the same. A sound and complete sequent calculus proof system of $\mathrm{CL}^{\supset,\mathsf{F}}$ can be obtained by adding the following two inference rules to the sequent calculus proof system of $\mathrm{BD}^{\supset,\mathsf{F}}$:

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

If we add only the inference rule ¬-R to the sequent calculus proof system of $\mathrm{BD}^{\supset,\mathsf{F}}$, then we obtain a sound and complete proof system of the paraconsistent (but not paracomplete) logic $\mathrm{LPQ}^{\supset,\mathsf{F}}$ presented in [21]. If we add only the inference rule ¬-L to the sequent calculus proof system of $\mathrm{BD}^{\supset,\mathsf{F}}$, then we obtain a sound and complete proof system of the obvious first-order version of the paracomplete (but not paraconsistent) propositional logic $\mathrm{K3}^{\supset,\mathsf{F}}$ presented in [20].

5 Covering Terms with an Indeterminate Value

This section goes into a minor variation of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, called $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$, that can deal with terms with an indeterminate value. Semantically, this is handled by restricting the domain of structures to sets that contain a special dummy value $\perp$ and assigning this value to terms with an indeterminate value. Thus, $\perp$ is treated as a genuine value. It differs from other values only by conditions imposed on the equality relation of structures (see below). The conditions concerned fit in with the intuition that a term with an indeterminate value has an unknown value or does not have a value.

The language of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ and the language of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$ are the same. The logical consequence relation of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$ is defined as for $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$, but in terms of structures of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$. These structures differ slightly from the structures of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$.

Structures

A *structure* $\mathbf{A}$ of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$ is a pair $\langle \mathcal{U}^{\mathbf{A}}, \mathcal{I}^{\mathbf{A}} \rangle$, where:

- $\mathcal{U}^{\mathbf{A}}$ is a set, called the *domain of* $\mathbf{A}$, such that $\perp \in \mathcal{U}^{\mathbf{A}}$, $\mathcal{U}^{\mathbf{A}} \setminus \{\perp\} \neq \emptyset$, and $\mathcal{U}^{\mathbf{A}} \cap \mathcal{V} = \emptyset$;
- $\mathcal{I}^{\mathbf{A}}$ consists of:

- an element $c^{\mathbf{A}} \in \mathcal{U}^{\mathbf{A}}$ for every $c \in \mathcal{F}unc_0(\Sigma)$;
- a function $f^{\mathbf{A}} : \overbrace{\mathcal{U}^{\mathbf{A}} \times \cdots \times \mathcal{U}^{\mathbf{A}}}^{n+1 \text{ times}} \to \mathcal{U}^{\mathbf{A}}$ for every $f \in \mathcal{F}unc_{n+1}(\Sigma)$ and $n \in \mathbb{N}$;
- an element $p^{\mathbf{A}} \in \mathcal{V}$ for every $p \in \mathcal{P}red_0(\Sigma)$;
- a function $P^{\mathbf{A}} : \overbrace{\mathcal{U}^{\mathbf{A}} \times \cdots \times \mathcal{U}^{\mathbf{A}}}^{n+1 \text{ times}} \to \mathcal{V}$ for every $P \in \mathcal{P}red_{n+1}(\Sigma)$ and $n \in \mathbb{N}$;
- a function $=^{\mathbf{A}} : \mathcal{U}^{\mathbf{A}} \times \mathcal{U}^{\mathbf{A}} \to \mathcal{V}$ where, for all $d_1, d_2 \in \mathcal{U}^{\mathbf{A}}$:

  $=^{\mathbf{A}}(d_1, d_2) \in \mathcal{D}$ iff $d_1, d_2 \in \mathcal{U}^{\mathbf{A}} \setminus \{\bot\}$ and $d_1 = d_2$;
  $=^{\mathbf{A}}(d_1, d_2) = \mathsf{n}$ iff $d_1 = \bot$ or $d_2 = \bot$.

By the first condition imposed on $=^{\mathbf{A}}$ in the definition of a structure $\mathbf{A}$ of $\mathrm{BD}_{\bot}^{\supset,\mathsf{F}}(\Sigma)$, $[\![t = t]\!]_{\alpha}^{\mathbf{A}} \in \mathcal{D}$ iff $[\![t]\!]_{\alpha}^{\mathbf{A}} \neq \bot$. In other words, the truth value of the formula $t = t$ is a designated truth value iff the value of the term t is determinate.

By the second condition imposed on $=^{\mathbf{A}}$ in the definition of a structure $\mathbf{A}$ of $\mathrm{BD}_{\bot}^{\supset,\mathsf{F}}(\Sigma)$, $[\![t_1 = t_2]\!]_{\alpha}^{\mathbf{A}} = \mathsf{n}$ iff $[\![t_1]\!]_{\alpha}^{\mathbf{A}} = \bot$ or $[\![t_2]\!]_{\alpha}^{\mathbf{A}} = \bot$. In other words, the truth value of an equation $t_1 = t_2$ is neither true nor false (n) iff the value of t_1 or t_2 or both is indeterminate. Similar conditions are not imposed on the interpretations of the function and predicate symbols from Σ. However, because of the first condition imposed on $=^{\mathbf{A}}$, the relevant conditions can be expressed in $\mathrm{BD}_{\bot}^{\supset,\mathsf{F}}(\Sigma)$.

Logical Consequence

Let Γ and Δ be sets of formulas from $\mathcal{F}orm(\Sigma)$. Then Δ *is a logical consequence of* Γ, written $\Gamma \vDash_{\bot} \Delta$, iff for all structures $\mathbf{A}$ of $\mathrm{BD}_{\bot}^{\supset,\mathsf{F}}(\Sigma)$, for all assignments α in $\mathbf{A}$, if $[\![A]\!]_{\alpha}^{\mathbf{A}} \in \mathcal{D}$ for all $A \in \Gamma$, then $[\![A']\!]_{\alpha}^{\mathbf{A}} \in \mathcal{D}$ for some $A' \in \Delta$.

By the conditions on $=^{\mathbf{A}}$ in the definition of a structure $\mathbf{A}$ of $\mathrm{BD}_{\bot}^{\supset,\mathsf{F}}(\Sigma)$, we have that, for all $t_1, t_2 \in \mathcal{T}erm(\Sigma)$:

$$t_1 = t_1 \land t_2 = t_2 \vDash_{\bot} t_1 = t_2 \lor \neg(t_1 = t_2)\ ,$$
$$t_1 = t_2 \lor \neg(t_1 = t_2) \vDash_{\bot} t_1 = t_1 \land t_2 = t_2\ .$$

Proof System

A sequent calculus proof system of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$ is obtained by replacing the inference rule =-Refl by the following two inference rules in the sequent calculus proof system of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$:

$$\boxed{\delta\text{-=-L}}\ \frac{t_1 = t_1, t_2 = t_2, \Gamma \Rightarrow \Delta}{t_1 = t_2 \vee t_1 \neq t_2, \Gamma \Rightarrow \Delta}$$

$$\boxed{\delta\text{-=-R}}\ \frac{\Gamma \Rightarrow \Delta, t_1 = t_1 \quad \Gamma \Rightarrow \Delta, t_2 = t_2}{\Gamma \Rightarrow \Delta, t_1 = t_2 \vee t_1 \neq t_2}$$

The resulting proof system is sound and complete. It is easy to see that the resulting proof system is sound. As explained in [22], the completeness proof requires a minor adaptation of the completeness proof for the proof system of $\mathrm{BD}^{\supset,\mathsf{F}}(\Sigma)$ given in that paper.

Abbreviations

In what follows, the following abbreviations will be used:

$$\begin{array}{rll} t{\downarrow} & \text{stands for } \Delta(t = t) & \text{(term determinacy)}, \\ t_1 == t_2 & \text{stands for } t_1 = t_2 \vee \neg(t_1{\downarrow} \vee t_2{\downarrow}) & \text{(strong equality)}. \end{array}$$

It follows from these definitions that:

$$\begin{aligned} [\![t{\downarrow}]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} \text{ if } [\![t]\!]^{\mathbf{A}}_{\alpha} \in \mathcal{U}^{\mathbf{A}} \setminus \{\perp\} \\ \mathsf{f} \text{ otherwise,} \end{cases} \\ [\![t_1 == t_2]\!]^{\mathbf{A}}_{\alpha} &= \begin{cases} \mathsf{t} & \text{if } [\![t_1]\!]^{\mathbf{A}}_{\alpha} = \perp \text{ and } [\![t_2]\!]^{\mathbf{A}}_{\alpha} = \perp \\ =^{\mathbf{A}}([\![t_1]\!]^{\mathbf{A}}_{\alpha}, [\![t_2]\!]^{\mathbf{A}}_{\alpha}) & \text{otherwise.} \end{cases} \end{aligned}$$

Notes

$\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$ looks like a free logic [17, 23] that is paraconsistent and paracomplete. However, $\perp$ is included in the range of variables in $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$, whereas $\perp$ would be excluded from the range of variables in a free logic. This difference reflects that $\perp$ is treated as a genuine value in $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$ and $\perp$ is not treated as a genuine value in free logics.

$\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$ has been devised to deal with logical theories (i.e. sets of formulas) of the kind that arise for example when possibly inconsistent relational databases with possibly null values are viewed as logical theories.

6 Relational Databases Viewed through $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$

In this section, relational databases are considered from the perspective of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$, taking into account that a database may be an inconsistent database (cf. [6, 2]) and/or a database with null values (cf. [30, 31, 14]). The proof theoretic point of view is taken, i.e. a relational database is viewed as a logical theory. 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 to a great extent based on those given in [26]. However, types are ignored for the sake of simplicity (cf. [11, 29]). The view on null values in relational databases taken here is informally described in Section 1.

Relational Languages

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

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

- $\mathcal{F}unc_0(\Sigma)$ is finite, $\mathsf{nil} \in \mathcal{F}unc_0(\Sigma)$, and $\mathcal{F}unc_0(\Sigma) \setminus \{\mathsf{nil}\}$ is non-empty;
- $\bigcup\{\mathcal{F}unc_{n+1}(\Sigma) \mid n \in \mathbb{N}\}$ 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 $P(c_1, \ldots, c_{n+1})$, where $P \in \mathcal{P}red_{n+1}(\Sigma)$ and $c_1, \ldots, c_{n+1} \in \mathcal{F}unc_0(\Sigma)$.

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

$$\neg(\mathsf{nil}{\downarrow})$$

and the *equality semi-normality axiom for R* is the formula

$$\forall x, x' \bullet \circ(x = x') \wedge ((x{\downarrow} \wedge x'{\downarrow}) \rightarrow \star(x = x'))\ .$$

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

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

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 and let $P \in \mathcal{P}red_{n+1}(\Sigma)$ ($n \in \mathbb{N}$). Then the *P-determinacy axiom for R* is the formula

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

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 $P(c_1^1, \ldots, c_{n+1}^1)$, $\ldots$, $P(c_1^m, \ldots, c_{n+1}^m)$ be all formulas from Λ in which P occurs. Then the *P-completion axiom for Λ* is the formula

$$\forall x_1, \ldots, x_{n+1} \bullet P(x_1, \ldots, x_{n+1}) \rightarrow \\ x_1 == c_1^1 \wedge \ldots \wedge x_{n+1} == c_{n+1}^1 \ \vee \ \ldots \ \vee \ x_1 == c_1^m \wedge \ldots \wedge x_{n+1} == c_{n+1}^m\ .$$

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

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

The domain closure, unique name, and P-completion axioms are adopted from [26]. The nil-indeterminacy, equality semi-normality, and P-determinacy axioms are new. The nil-indeterminacy axiom states that the value of nil is indeterminate. The equality semi-normality axiom states that equations are interpreted classically except that their truth value is neither true nor false if terms with an indeterminate value are involved. The P-determinacy axiom states that P never yields the truth value neither true nor false.

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

- A is the nil-indeterminacy axiom for R;
- A is the equality semi-normality axiom for R;
- A is the domain closure axiom for R;
- A is an element of the unique name axiom set for R;
- A is the P-determinacy axiom for R
  for some $P \in \bigcup\{\mathcal{P}red_{n+1}(\Sigma) \mid n \in \mathbb{N}\}$.

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 $RT(R, \Lambda)$, is the set of all formulas $A \in \mathcal{F}orm(\Sigma)$ for which one of the following holds:

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

A set $\Theta \subseteq \mathcal{F}orm(\Sigma)$ is called a *relational theory for R* if $\Theta = RT(R, \Lambda)$ for some finite set $\Lambda \subseteq \mathcal{F}orm(\Sigma)$ 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 2 *Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, and let Γ be a finite subset of $\mathcal{F}orm(\Sigma)$ such that $RSA(R) \subseteq \Gamma$. Then it is decidable whether, for a formula $A \in \mathcal{F}orm(\Sigma)$, $\Gamma \Rightarrow A$ is provable.*

Proof: Since it is known from Theorem 1 that $\Gamma \Rightarrow A$ is provable iff $\Gamma \vDash A$, it is shown instead that it is decidable whether, for a formula $A \in \mathcal{F}orm(\Sigma)$, $\Gamma \vDash A$.

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 occurring in $\Gamma \cup \{A\}$. Because the set of all variable occurring 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 Proposition 1 that it is decidable whether, for a formula $A \in \mathcal{F}orm(\Sigma)$, $\Gamma \vDash A$. □

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 $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$.

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

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

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.

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

Models of Relational Theories

The models of relational theories for a relational language $R = (\Sigma, \mathcal{F}orm(\Sigma))$ are structures of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$ 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 $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$ such that:

- $\mathsf{nil}^{\mathbf{A}} = \perp$;
- for all $d_1, d_2 \in \mathcal{U}^{\mathbf{A}}$:

  $=^{\mathbf{A}}(d_1, d_2) \neq \mathsf{b}$;
  if $d_1 \neq \perp$ and $d_1 \neq \perp$, then $=^{\mathbf{A}}(d_1, d_2) \neq \mathsf{n}$;
- for all $d \in \mathcal{U}^{\mathbf{A}}$:

  if $d \neq \perp$, then there exists a $c \in \mathcal{F}unc_0(\Sigma)$ such that $=^{\mathbf{A}}(d, c^{\mathbf{A}}) = \mathsf{t}$;
  if $d = \perp$, then there exists a $c \in \mathcal{F}unc_0(\Sigma)$ such that $=^{\mathbf{A}}(d, c^{\mathbf{A}}) = \mathsf{n}$;
- for all $c_1, c_2 \in \mathcal{F}unc_0(\Sigma)$:

  if $=^{\mathbf{A}}(c_1{}^{\mathbf{A}}, c_1{}^{\mathbf{A}}) \neq \mathsf{n}$ and $=^{\mathbf{A}}(c_2{}^{\mathbf{A}}, c_2{}^{\mathbf{A}}) \neq \mathsf{n}$ and $=^{\mathbf{A}}(c_1{}^{\mathbf{A}}, c_2{}^{\mathbf{A}}) = \mathsf{t}$
  or
  $=^{\mathbf{A}}(c_1{}^{\mathbf{A}}, c_1{}^{\mathbf{A}}) = \mathsf{n}$ and $=^{\mathbf{A}}(c_2{}^{\mathbf{A}}, c_2{}^{\mathbf{A}}) = \mathsf{n}$,
  then $c_1 \equiv c_2$;
- for all $n \in \mathbb{N}$, for all $P \in \mathcal{P}red_{n+1}(\Sigma)$, for all $d_1, \ldots, d_{n+1} \in \mathcal{U}^{\mathbf{A}}$:

  $P^{\mathbf{A}}(d_1, \ldots, d_{n+1}) \neq \mathsf{n}$.

The following corollary of the definitions of relational structure axioms for R and relational structure for R justifies the term "relational structure axioms".

Corollary 1 *Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, and let $\mathbf{A}$ be a structure of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}(\Sigma)$. Then $\mathbf{A}$ is a relational structure for R iff, for all assignments α in $\mathbf{A}$, for all $A \in RSA(R)$, $[\![A]\!]_{\alpha}^{\mathbf{A}} \in \mathcal{D}$.*

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, and let Θ be a relational theory for R. Then all models of Θ are relational structures for R because $RSA(R) \subseteq \Theta$. 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 $R = (\Sigma, \mathcal{F}orm(\Sigma))$ 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 $c \in \mathcal{F}unc_0(\Sigma)$, $c^{\mathbf{A}'} = c^{\mathbf{A}}$;
- for each $n \in \mathbb{N}$, for each $P \in \mathcal{P}red_{n+1}(\Sigma)$, for each $d_1, \ldots, d_{n+1} \in \mathcal{U}^{\mathbf{A}'}$,
$P^{\mathbf{A}'}(d_1, \ldots, d_{n+1}) = \begin{cases} \mathsf{t} & \text{if } P^{\mathbf{A}}(d_1, \ldots, d_{n+1}) \in \{\mathsf{t}, \mathsf{b}\} \\ \mathsf{f} & \text{otherwise;} \end{cases}$
- for each $d_1, d_2 \in \mathcal{U}^{\mathbf{A}'}$, $=^{\mathbf{A}'}(d_1, d_2) \;=\; =^{\mathbf{A}}(d_1, d_2)$.

Theorem 3 *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.*

Proof: The proof goes in almost the same way as the proof of part 1 of Theorem 3.1 from [26]. The only point of attention is that it may be the case that, for some $P \in \mathcal{P}red_{n+1}(\Sigma)$ and $c_1, \ldots, c_{n+1} \in \mathcal{F}unc_0(\Sigma)$ $(n \in \mathbb{N})$, 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 4 *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: The proof goes in the same way as the proof of part 2 of Theorem 3.1 from [26]. □

7 Query Answering Viewed through $\mathrm{BD}^{\supset,\mathsf{F}}_{\perp}$

In this section, queries applicable to a relational database and their answers are considered from the perspective of $\mathrm{BD}^{\supset,\mathsf{F}}_{\perp}$. As a matter of fact, the queries introduced below are closely related to the relational-calculus-oriented queries originally introduced by [9]. Three kinds of answers to queries are introduced. Two of them take the integrity constraint of the database into account. They differ in their approach to deal with inconsistencies. A discussion of these approaches can be found in [21, Section 8].

Queries

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

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

- $x_1, \ldots, x_n \in \mathcal{V}ar$;
- $A \in \mathcal{F}\!orm(\Sigma)$ 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 $R = (\Sigma, \mathcal{F}\!orm(\Sigma))$ 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 $(c_1, \ldots, c_n) \in \mathcal{F}unc_0(\Sigma)^n$ 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 formula that corresponds to an answer is a logical consequence of some set of atomic facts and negations of atomic facts that are logical consequences of the relational theory of the database;
- 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 $R = (\Sigma, \mathcal{F}\!orm(\Sigma))$ be a relational language. Then a *semi-atomic fact for R* is an $A \in \mathcal{F}\!orm(\Sigma)$ 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.

Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ 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 $(c_1, \ldots, c_n) \in \mathcal{F}unc_0(\Sigma)^n$ for which there exists a set Φ of semi-atomic facts 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 [6]. 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 every minimally repaired version 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 [2], 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 $R = (\Sigma, \mathcal{F}orm(\Sigma))$ be a relational language, let $\Lambda \subseteq \mathcal{F}orm(\Sigma)$ be a finite set of atomic facts for R, and let Ξ is 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 $(x_1, \ldots, x_n) \bullet A$ 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 $(c_1, \ldots, c_n) \in \mathcal{F}unc_0(\Sigma)^n$ 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 [2]. It represents, presumably, the first view on what the repairs of an inconsistent database are. Other views on what the repairs of an inconsistent database are have been taken in e.g. [4, 7, 13, 19, 28].

Null Values in Answers

If $(c_1, \ldots, c_n)$ is an answer, consistent answer or strongly consistent answer to a query with respect to a database with null values, then each of $c_1, \ldots, c_n$ may be syntactically equal to nil. Such answers are called *answers with null values.* In most work on query answering with respect to a database with null values, answers with null values are not considered. Notable exceptions are [14, 18], where the view taken on null values in relational databases is different from the view taken in this paper.

Decidability

The following theorem is a decidability result concerning being an answer to a query with respect to a database.

Theorem 5 *Let $R = (\Sigma, \mathcal{F}orm(\Sigma))$ 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 $(c_1, \ldots, c_n) \in \mathcal{F}unc_0(\Sigma)^n$:*

- *$(c_1, \ldots, c_n)$ 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 2 and the definition of the kind of answer concerned. □

As a corollary of Theorem 5, 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.

Example

The example given below is kept extremely simple so that readers that are not initiated in the sequent calculus proof system of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$ can understand the remarks made about the provability of sequents.

Consider the relational database whose relational language, say R, has constant symbols a, b, c, d, and nil and ternary predicate symbol P, whose relational theory is the relational theory of which $P(a, b, \mathsf{nil})$, $P(a, \mathsf{nil}, c)$, $P(a, \mathsf{nil}, d)$, and $P(b, c, d)$ are the atomic facts, and whose single integrity constraint is $\forall x, y, z, y', z' \bullet (P(x, y, z) \land P(x, y', z')) \to y = y'$. Moreover, consider the query $x \bullet \exists y, z, z' \bullet P(x, y, z) \land P(x, y, z') \land z \neq z'$. Clearly, the set of answers is $\{a\}$.

The sets of semi-atomic formulas that are logical consequences of the relational theory and do not cause the database to be inconsistent include only one of $P(a, b, \mathsf{nil})$, $P(a, \mathsf{nil}, c)$, and $P(a, \mathsf{nil}, d)$. This means that, for each such set, say Φ, there is no constant symbol $k \in \{a, b, c, d, \mathsf{nil}\}$ such that $\Phi, RSA(R) \Rightarrow [y := k](\exists x, z, z' \bullet P(x, y, z) \land P(x, y, z') \land z \neq z')$ is provable. Hence, the set of consistent answers is $\emptyset$.

The repairs of $\{P(a, b, \mathsf{nil}), P(a, \mathsf{nil}, c), P(a, \mathsf{nil}, d), P(b, c, d)\}$ include only one of $P(a, b, \mathsf{nil})$, $P(a, \mathsf{nil}, c)$, and $P(a, \mathsf{nil}, d)$. This means that, for each repair, say Λ, there is no constant symbol $k \in \{a, b, c, d, \mathsf{nil}\}$ such that $RT(R, \Lambda) \Rightarrow [y := k](\exists x, z, z' \bullet P(x, y, z) \land P(x, y, z') \land z \neq z')$ is provable. Hence, the set of strongly consistent answers is also $\emptyset$.

Now, consider the relational database that differs from the one described above only in that in the integrity constraint $y = y'$ is replaced by $y == y'$. It is easy to see that in this case the set of answers and the set of strongly consistent answers to the query described above remain the same, but the set of consistent answers becomes $\{a\}$.

Several examples in which null values play no role can be found in [21].

8 The Views on Null Values in Databases

In this short section, some remarks are made on the different views that exist on null values in relational databases.

Our View

The view taken in this paper on null values in relational databases can be summarized as follows:

- in relational databases with null values, a single dummy value, called the null value, is used for values that are indeterminate;
- a value that is indeterminate is a value that is either unknown or nonexistent;
- independent of whether it is unknown or nonexistent, no meaningful answer can be given to the question whether the null value and whatever value, including the null value itself, are the same.

In the database literature, a null value for all values that are indeterminate, i.e. for all values that are either unknown or nonexistent, is usually called a *no information* null value. Moreover, the term *inapplicable* is often used instead of nonexistent.

Other Views

Firstly, the view taken in this paper means that there are no separate null values for values that are unknown and values that are nonexistent. Separate null values for values that are unknown and values that are nonexistent have for example been studied in [12, 30]. In those papers, query answering is based on a four-valued logic devised to deal with the two different null values. The logics concerned differ from each other and both differ a lot from $\mathrm{BD}_{\bot}^{\supset,\mathsf{F}}$.

Secondly, the view taken in this paper means that the values for which the null value is a dummy value is not restricted to values that are unknown. A single null value, for values that are unknown only, has for example been studied in [5, 10, 14]. In [14] and later publications in which this kind of null value is considered, query answering with respect to a database with null values is usually approached in a way that is based on the idea that a database with null values represents a set of possible databases without null values. This approach is reminiscent of the database repair approach to query answering with respect to an inconsistent database from [2].

Thirdly, the view taken in this paper means that there are no multiple null values. Multiple null values, each for values that are unknown only, has for example been studied in [14, 18]. In those papers, the same approach to query answering is applied as in [14] to the case of a single null value.

A single null value for both values that are unknown and values that are nonexistent has also been studied before in [16, 31]. In [31], query answering is based on a three-valued logic that is closely related to $\mathrm{BD}_{\bot}^{\supset,\mathsf{F}}$. In [16], an attempt is made to apply the approach to query answering from [14] to the case of a single null value for both values that are unknown and values that are nonexistent.

The fact that so many views on null values in relational databases have been studied indicates that time and again the views already studied turned out to be unsatisfactory from one angle or another. The view taken in this paper is primarily based on the consideration that it should be relatively simple from a logical perspective, taking into account that a database is anyhow no more than a representation of an approximation of a piece of reality.

The Equality Issue

The view taken in this paper is essentially the same as the view taken in [31] and is based on similar considerations. There is one aspect of this view that is often considered undesirable in work on null values in relational databases, namely the point that no meaningful answer can be given to the question whether the null value equals itself. Made precise in the setting of this paper, it is the following that is often considered undesirable:

$$[\![t_1 = t_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{n} \quad \text{if} \quad [\![t_1]\!]^{\mathbf{A}}_{\alpha} = [\![\mathsf{nil}]\!]^{\mathbf{A}}_{\alpha} \text{ and } [\![t_2]\!]^{\mathbf{A}}_{\alpha} = [\![\mathsf{nil}]\!]^{\mathbf{A}}_{\alpha} \; .$$

The main reason that is given to consider this undesirable is that it may lead to counterintuitive answers to queries. However, this is only so in the case where it is (implicitly) assumed that the null value is a dummy value for values that are unknown only. In that case, the following is considered more appropriate:

$$[\![t_1 = t_2]\!]^{\mathbf{A}}_{\alpha} = \mathsf{t} \quad \text{if} \quad [\![t_1]\!]^{\mathbf{A}}_{\alpha} = [\![\mathsf{nil}]\!]^{\mathbf{A}}_{\alpha} \text{ and } [\![t_2]\!]^{\mathbf{A}}_{\alpha} = [\![\mathsf{nil}]\!]^{\mathbf{A}}_{\alpha}$$

(see e.g. [18]).

It is worth mentioning that the adapted equality predicate can be expressed in the setting of this paper. Consider the abbreviation $t_1 == t_2$ introduced earlier. As mentioned before, it follows from the definition of this abbreviation that:

$$[\![t_1 == t_2]\!]^{\mathbf{A}}_{\alpha} = \begin{cases} \mathsf{t} & \text{if } [\![t_1]\!]^{\mathbf{A}}_{\alpha} = \bot \text{ and } [\![t_2]\!]^{\mathbf{A}}_{\alpha} = \bot \\ =^{\mathbf{A}}([\![t_1]\!]^{\mathbf{A}}_{\alpha}, [\![t_2]\!]^{\mathbf{A}}_{\alpha}) & \text{otherwise.} \end{cases}$$

Clearly, $==$ corresponds to the adapted equality predicate. This means that, although the definitions in Sections 6 and 7 are based on the view that the null value is a dummy value for both values that are unknown and values that are nonexistent, $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$ allows for treating the null value as a dummy value for values that are unknown only.

9 Concluding Remarks

In this paper, a coherent logical view on relational databases and query answering is presented that takes into account the possibility that a database is a database with null values and the possibility that a database is an inconsistent database. The presented view combines:

- the proof-theoretic logical view of Reiter [26] 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 without null values;
- the view of Zaniolo [31] on null values in relational databases;
- the view of Bry [6] as well as the view of Arenas et al [2] on what is a consistent answer to a query with respect to an inconsistent relational database.

The view is expressed in the setting of the paraconsistent and paracomplete logic $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$. In a logic that is not paracomplete, it would have been difficult to take properly into account the possibility that a database is a database with null values, and in a logic that is not paraconsistent, it would have been difficult to take properly into account the possibility that a database is an inconsistent database.

The main views on what is a consistent answer to a query with respect to an inconsistent relational database are the view of Bry [6] and the view of Arenas et al [2]. The latter view is based on a notion of a minimal repair of an inconsistent database. Most other views on consistent query answering, for example the views presented in [4, 7, 13, 19, 28], are based on a notion of a minimal repair of an inconsistent database that differs from the one from [2] by different choices concerning, among other things, the kinds of changes (deletions, additions, alterations) that may be made to the original database to obtain a repair and what is taken as the extent to which two databases differ.

There are four main views on null values in relational databases, namely the view of Zaniolo [31], two views of Imieliński and Lipski [14], and the view of Vassilion [30]. Other views are mostly variations on one of these views. The *Codd-table* view from [14], in which there are no multiple null values, can be expressed in the setting of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$ because the single null value can be treated as in that view by using == instead of =. In the *V-table* view from [14], there may be multiple null values (called marked nulls). A variation of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$ is needed to express this view. In the view from [30], there are separate null values for values that are unknown and values that are nonexistent. It is questionable whether $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$ is suitable to express this view on null values in relational databases, in particular if the possibility that a databases is inconsistent has to be taken into account as well.

It is shown in this paper that, for each of the three kinds of answer introduced, being an answer to a query with respect to a relational database is decidable. However, it is to be expected that query answering for these kinds of answers is intractable without severe restrictions on the integrity constraints and/or the use of null values. A study of the computational complexity of query answering for these kinds of answers in the setting of $\mathrm{BD}_{\perp}^{\supset,\mathsf{F}}$ is an interesting option for future work.

Acknowledgement

I thank an anonymous referee for carefully reading a preliminary version of this paper, for suggesting improvements of the presentation of the paper, and for drawing attention to several typing errors.

References

[1] Alan Anderson and Nuel Belnap. First degree entailments. *Mathematische Annalen*, 149(4):302–319, 1963. doi:10.1007/BF01471125.

[2] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In Victor Vianu and Christos H. Papadimitriou, editors, *18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1999*, pages 68–79. ACM Press, 1999. doi:10.1145/303976.303983.

[3] Ofer Arieli and Arnon Avron. Four-valued paradefinite logics. *Studia Logica*, 105(6):1087–1122, 2017. doi:10.1007/s11225-017-9721-4.

[4] Leopoldo E. Bertossi. Database repairs and consistent query answering: Origins and further developments. In Dan Suciu, Sebastian Skritek,

and Christoph Koch, editors, *38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019*, pages 48–58. ACM Press, 2019. doi:10.1145/3294052.3322190.

[5] Joachim Biskup. A foundation of Codd's relational maybe-operations. *ACM Transactions on Database Systems*, 8(4):608–636, 1983. doi:10.1145/319996.320014.

[6] François Bry. Query answering in information systems with integrity constraints. In Sushil Jajodia, William List, Graeme W. McGregor, and Leon Strous, editors, *1st Working Conference on Integrity and Internal Control in Information Systems, IICIS 1997*, volume 109 of *IFIP Conference Proceedings*, pages 113–130. Chapman and Hall, 1997. doi:10.1007/978-0-387-35317-3_6.

[7] Marco Calautti, Leonid Libkin, and Andreas Pieris. An operational approach to consistent query answering. In Jan Van den Bussche and Marcelo Arenas, editors, *37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2018*, pages 239–251. ACM Press, 2018. doi:10.1145/3196959.3196966.

[8] Roberto Ciuni and Massimiliano Carrara. Normality operators and classical recapture in many-valued logic. *Logic Journal of the IGPL*, 28(5):657–683, 2020. doi:10.1093/jigpal/jzy055.

[9] Edgar F. Codd. Relational completeness of data base sublanguage. Technical Report RJ 987, IBM, 1972.

[10] Edgar F. Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, 4(4):397–434, 1979. doi:10.1145/320107.320109.

[11] Hervé Gallaire, Jack Minker, and Jean-Marie Nicolas. Logic and databases: A deductive approach. *Computing Surveys*, 16(2):153–185, 1984. doi:10.1145/356924.356929.

[12] G. H. Gessert. Four valued logic for relational database systems. *ACM SIGMOD Record*, 19(3):29–35, 1990. doi:10.1145/382274.382401.

[13] Sergio Greco and Cristian Molinaro. Consistent query answering over inconsistent databases. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 15(3):119–129, 2011. doi:10.3233/KES-2011-0216.

[14] Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984. doi:10.1145/1634.1886.

[15] Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, 1952.

[16] Hans-Joachim Klein. Null values in relational databases and sure information answers. In Leopoldo E. Bertossi, Gyula O. H. Katona, Klaus-Dieter Schewe, and Bernhard Thalheim, editors, *2nd International Workshop on Semantics in Databases*, volume 2582 of *Lecture Notes in Computer Science*, pages 119–138. Springer-Verlag, 2001. doi:10.1007/3-540-36596-6_7.

[17] Karel Lambert. Free logic and the concept of existence. *Notre Dame Journal of Formal Logic*, 8(1–2):133–144, 1967. doi:10.1305/ndjfl/1093956251.

[18] Leonid Libkin. SQL's three-valued logic and certain answers. *ACM Transactions on Database Systems*, 41(1):Article 1, 2016. doi:10.1145/2877206.

[19] Andrei Lopatenko and Leopoldo E. Bertossi. Consistent query answering by minimal-size repairs. In *17th International Workshop on Database and Expert Systems Applications, DEXA 2006*, pages 558–562. IEEE, 2006. doi:10.1109/DEXA.2006.43.

[20] Cornelis A. Middelburg. On the strongest three-valued paraconsistent logic contained in classical logic and its dual. *Journal of Logic and Computation*, 31(2):597–611, 2021. doi:10.1093/logcom/exaa084.

[21] Cornelis A. Middelburg. Paraconsistent logic and query answering in inconsistent databases. *CoRR*, abs/2208.12976, 2022. arXiv:2208.12976.

[22] Cornelis A. Middelburg. A conventional expansion of first-order Belnap-Dunn logic. *CoRR*, abs/2301.10555, 2023. arXiv:2301.10555.

[23] John Nolt. Free logics. In Dale Jacquette, editor, *Philosophy of Logic*, Handbook of the Philosophy of Science, pages 1023–1060. Elsevier, 2007. doi:10.1016/B978-044451541-4/50027-0.

[24] Lavinia María Picollo. Truth in a logic of formal inconsistency: How classical can it get? *Logic Journal of the IGPL*, 28(5):771–806, 2020. doi:10.1093/jigpal/jzy059.

[25] Graham Priest. The logic of paradox. *Journal of Philosophical Logic*, 8(1):219–241, 1979. doi:10.1007/BF00258428.

[26] Raymond Reiter. Towards a logical reconstruction of relational database theory. In Michael L. Brodie, John Mylopoulos, and Joachim W. Schmidt, editors, *On Conceptual Modelling*, Topics in information systems, pages 191–238. Springer-Verlag, 1984. doi:10.1007/978-1-4612-5196-5_8.

[27] Katsuhiko Sano and Hitoshi Omori. An expansion of first-order Belnap-Dunn logic. *Logic Journal of the IGPL*, 22(3):458–481, 2014. doi:10.1093/jigpal/jzt044.

[28] Balder ten Cate, Gaëlle Fontaine, and Phokion G. Kolaitis. On the data complexity of consistent query answering. *Theory of Computing Systems*, 57(4):843–891, 2015. doi:10.1007/s00224-014-9586-0.

[29] Moshe Y. Vardi. Querying logical databases. *Journal of Computer and System Sciences*, 33(2):142–160, 1986. doi:10.1016/0022-0000(86)90016-4.

[30] Yannis Vassiliou. Null values in data base management: A denotational semantics approach. In Philip A. Bernstein, editor, *1979 ACM SIGMOD International Conference on Management of Data*, pages 162–169. ACM Press, 1979. doi:10.1145/582095.582123.

[31] Carlo Zaniolo. Database relations with null values. *Journal of Computer and System Sciences*, 28(1):142–166, 1984. doi:10.1016/0022-0000(84)90080-1.