TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Artificial Intelligence Knowledge Representation and Reasoning

Expressing View-Based Query Processing and Related Approaches with Second-Order Operators

Christoph Wernhard

KRR Report 14-02

Mail to
Technische Universität Dresden
01062 Dresden

Bulk mail to
Technische Universität Dresden
Helmholtzstr. 10
01069 Dresden

Office
Room 2006
Nöthnitzer Straße 46
01187 Dresden

Internet
http://www.wv.inf.tu-dresden.de

Expressing View-Based Query Processing and Related Approaches with Second-Order Operators

Christoph Wernhard
Technische Universität Dresden

Table of Contents

Chapter 1

Introduction

An "intensional" answer [Mot94] obtained from a database system or a knowledge-based system characterizes the requested information not just in terms of explicitly listed predicate extensions, but in a way that is more convenient for the client, in terms of concepts that can be understood or efficiently processed in some sense by the client, abstracting from concrete extensions. In distributed settings with multiple agents and knowledge sources, such answers are particularly useful as intermediates that are passed as queries to agents with specialized knowledge bases and processing capabilities. View-based query processing [Hal01,CGLV07,Mar07,NSV10] is an approach from database research that involves such an intermediate layer and has numerous applications in query optimization, database design, information integration and distributed knowledge processing. A further approach, to be called here *split rewriting*, has been investigated in [BdBF$^+$10,FKN12,FKN13]. It is related to view-based query processing, but also differs from it in important aspects. Both approaches are actually closely related to the investigation of definability in a general logic setting, with second-order operators naturally suggesting themselves as means of expression [Tar35,Mar07].

The objective of this work is to develop a unifying formal framework that captures different forms of answers to queries semantically, including those involved in view-based query processing and split rewriting. The technical basis is classical first-order logic, extended with second-order operators, in particular for projection, a generalization of predicate quantification, and for circumscription. The second-order operators play a twofold role: First, they add the expressivity required to express the envisaged forms of answers in a natural way. Second, a form of computational processing is associated with them: Second-order operator elimination, that is, to compute for a given formula with second-order operators, an equivalent formula without them. Further second-order operators that correspond to recurring application patterns can be defined in terms of the primitive operator for projection. Here, the most important of these patterns is the *globally weakest sufficient condition* [Wer12], which is closely related to *weakest sufficient condition* [Lin01,DLS01] and provides the basis to specify notions of *definition* and *definability*. An answer can then be characterized generically as an alternate definition of the query with respect to the background knowledge base, meeting constraints about the allowed vocabulary and satisfying further application dependent properties, similarly to generic notions of *abductive explanation* [KKT98]. With this second-order framework, we model various forms of answers that occur in view-based query processing as made precise in [CGLV07] and in split rewriting as investigated in [BdBF$^+$10,FKN12,FKN13].

The reconstructions formally relate specific concepts of view-based query processing and split rewriting to fundamental general concepts of logic, indicating ways to generalize notions from databases to different types of knowledge bases and showing parallels with other areas, suggesting ways to transfer and combine techniques. Particularly relevant are abductive reasoning in logic programming [KKT98,DŁS01,Wer13], applications of uniform interpolation in description logics [GLW06,KWW09,LW11,CM12,KS13], and the investigation of dedicated methods for second-order quantifier elimination [GSS08].

The impatient reader might now skip to Sect. 25, where a tabular summary of the reconstructions of the concepts involved in view-based query processing is given. For split rewriting, the characterization and essential properties are given with Def. 74, Prop. 76 and Prop. 77.

The rest of the paper is organized as follows: In Chapter 2, syntax and semantics of the background framework is introduced, that is, first-order logic extended with a second-order operator for projection. Properties of projection that will be relevant in subsequent chapters are noted. The globally weakest sufficient condition and related application patterns of projection are specified in Chapter 3. Properties of them are stated formally and further concepts are defined in terms of them: *Definition* and *definability* as well as the related *unique definability*, *scope definability*, and the *conservative* property. In Chapter 4, a generic model of queries and answers is specified on the basis of the concept of *definition*. It is then shown how the conventional basic types of answers and slight extensions of these are rendered in this model. This includes the distinction between consistency- and entailment-based extensional answer, extensional answers that are characterized by formulas with equality constraints, answers with respect to datalog formulas, as well as a simple form of intensional answers. In Chapter 5, the presented framework is applied to reconstruct the concepts involved in *view-based query processing* as specified in [CGLV07]. Chapter 6 provides reconstructions of the concepts of split rewriting investigated in [BdBF$^+$10,FKN12,FKN13]. In the conclusion, Chapter 7, issues for future research that are opened up by this report are summarized.

Chapter 2

Notation and Preliminaries

1 First-Order Logic without Function Symbols

We assume a fixed first-order *signature* $\Sigma = \langle \mathsf{CONST}, \mathsf{PRED}, \mathsf{VAR} \rangle$, that is, a triple of a nonempty finite or countable infinite set CONST of *constants*, a finite set PRED of *predicates*, each with an associated arity larger than or equal to 0, and a finite set VAR of *variables*. The letters x, y, z, also with subscripts, denote variables. We basically consider formulas that are constructed from first-order *atoms* over Σ, truth value constants $\top, \bot$, the unary connective $\neg$, binary connectives $\wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow$ and first-order quantifiers $\forall, \exists$. In addition, we have $\doteq$ as logic operator for syntactic equality of terms. As meta-level notation we use n-ary versions of $\wedge$ and $\vee$. To save parentheses, we assume that the syntactic scope of quantifiers reaches as far to the right as possible. A *sentence* is a formula without free variables. We us $\boldsymbol{x}$ as a shorthand for the sequence $x_1, \ldots, x_n$ of variables. We write a formula F whose free variables are exactly $\boldsymbol{x}$ also as $F(\boldsymbol{x})$. If $\mathbf{c} = c_1, \ldots, c_n$ is a sequence of constants, then, in a context where $F(\boldsymbol{x})$ is specified, $F(\mathbf{c})$ denotes the sentence obtained from F by substituting each free occurrence of x_i by c_i, for $i \in \{1, \ldots, n\}$. A *universal first-order formula* is a formula of the form $\forall \boldsymbol{x} F$, where F is a first-order formulas without occurrences of quantifiers. Later on we will extend the notion of *formula* by allowing certain second-order operators.

If CONST is infinite, the restriction of first-order logic by disallowing function symbols with exception of constants does not essentially constrain expressivity. It simplifies expressing certain concepts and properties and is straightforwardly compatible with established formalizations of databases.

2 Literals and Scopes

A *literal* is a pair of an atom and a sign, where we write the positive (negative) literal with atom A as $+A$ ($-A$). The complement of a literal L is denoted by $\overline{L}$. If S is a set of literals, then $\overline{S}$ denotes the set of the complements of the members of S. An atom or literal without variables is called *ground*. Notice that so far we use literals "by themselves", as representatives of an atom and a polarity, in contrast to formula constituents. We call a formula that is an atom or a negated atom a *literal formula*, and only if no ambiguity arises, also briefly a *literal*.

A *scope* is a set of ground literals. The sets of all ground literals, all positive ground literals, and all negative ground literals over Σ are denoted by ALL, POS, NEG, respectively. An *atom scope* S is a scope such that $S = \overline{S}$. Since a

literal is a member of an atom scope if and only if its complement is a member, as a shorthand, we write an atom scope also just as the *set of atoms* of its members. A *predicate scope* is a scope whose members are all the ground literals whose predicate is in a given set of predicates. We use this *set of predicates* as a shorthand for a predicate scope. As an example, consider the scope $\{+\mathsf{p}(\mathsf{a}), -\mathsf{p}(\mathsf{a}), +\mathsf{p}(\mathsf{b}), -\mathsf{p}(\mathsf{q})\}$. Since it is an atom scope, it can be written as the set of atoms $\{\mathsf{p}(\mathsf{a}), \mathsf{p}(\mathsf{b})\}$. If $\mathsf{CONST} = \{\mathsf{a}, \mathsf{b}\}$, then it is a predicate scope and can also be written as $\{\mathsf{p}\}$.

3 Semantic Framework

We use a notational variant of the framework of Herbrand interpretations: An *interpretation* is a pair $\langle I, \beta\rangle$, where I is a *structure*, that is, a set of ground literals that contains for all ground atoms A over Σ exactly one of $+A$ or $-A$, and β is a *variable assignment*, that is, a mapping of the set of variables VAR into the set of constants CONST. As explained in [Wer12], structures in this sense represent Herbrand structures in the usual sense considered in model theory. The representation as sets of ground literals facilitates to express the semantics of certain second-order operators discussed later on.

The formula F with all free variables replaced by their image in β is denoted by $F\beta$; the variable assignment that maps x to the constant c and all other variables to the same values as β is denoted by $\beta\frac{c}{x}$.

The satisfaction relation between interpretations and a formulas is defined as shown for a choice of operators in Def. 1 below. The semantics of the remaining well-known operators can be specified analogously.

Definition 1 (Satisfaction Relation for First-Order Formulas). For interpretations $\langle I, \beta\rangle$, atoms A, terms t, u and formulas F, G, the *satisfaction* relation $\models$ is defined as follows"

$\langle I, \beta\rangle \models A$	iff_{def}	$+A\beta \in I$.
$\langle I, \beta\rangle \models t \doteq u$	iff_{def}	$\beta(t) = \beta(u)$.
$\langle I, \beta\rangle \models \top$.		
$\langle I, \beta\rangle \not\models \bot$.		
$\langle I, \beta\rangle \models \neg F$	iff_{def}	$\langle I, \beta\rangle \not\models F$.
$\langle I, \beta\rangle \models F \wedge G$	iff_{def}	$\langle I, \beta\rangle \models F$ land $\langle I, \beta\rangle \models G$.
$\langle I, \beta\rangle \models F \vee G$	iff_{def}	$\langle I, \beta\rangle \models F$ or $\langle I, \beta\rangle \models G$.
$\langle I, \beta\rangle \models \forall x\, F$	iff_{def}	for all $c \in \mathsf{CONST}$ it holds that $\langle I, \beta\frac{c}{x}\rangle \models F$.
$\langle I, \beta\rangle \models \exists x\, F$	iff_{def}	there is a $c \in \mathsf{CONST}$ s.th. $\langle I, \beta\frac{c}{x}\rangle \models F$.

If F is a sentence, then the β component of an interpretation $\langle I, \beta\rangle$ is irrelevant for the meaning of $\langle I, \beta\rangle \models F$. In this case, we sometimes let just the structure component I take the place of the interpretation, that is, we write $I \models F$ instead of $\langle I, \beta\rangle \models F$.

A formula F is called *satisfiable* if and only if there exists an interpretation $\langle I, \beta\rangle$ such that $\langle I, \beta\rangle \models F$. Entailment, equivalence, satisfiability and validity are straightforwardly defined in terms of the satisfaction relation. Entailment: $F \models G$ holds if and only if for all $\langle I, \beta\rangle$ such that $\langle I, \beta\rangle \models F$ it holds

that $\langle I, \beta\rangle \models G$. Equivalence: $F \equiv G$ if and only if $F \models G$ and $G \models F$. A formula F is satisfiable if and only if there exists an interpretation $\langle I, \beta\rangle$ such that $\langle I, \beta\rangle \models F$. A formula F is valid if and only if for all interpretations $\langle I, \beta\rangle$ it holds that $\langle I, \beta\rangle \models F$.

4 A Second-Order Operator for Projection

We extend first-order logic by certain second-order operators. One of these is for *projection* [Wer09,Wer08,LLM03], a generalization of second-order predicate quantification. Each of the standard operators for first-order logic has been semantically defined by a clause in Def. 1. The semantic definition of projection provides such a clause for the project operator:

Definition 2 (Projection). The *projection* of formula F onto scope S, in symbols $\mathsf{project}_S(F)$, is a formula whose semantics is defined as follows: For all interpretations $\langle I, \beta\rangle$ it holds that

$$\langle I, \beta\rangle \models \mathsf{project}_S(F) \quad \text{iff}_{\text{def}} \quad \begin{array}{l}\text{there exists a structure } J \text{ such that} \\ \langle J, \beta\rangle \models F \text{ and } J \cap S \subseteq I.\end{array}$$

Forgetting is a notational variant of projection, where the scope is considered complementary. We define it here not as a primitive but in terms of projection:

Definition 3 (Forgetting). The *forgetting* in formula F about scope S is defined as

$$\mathsf{forget}_S(F) \quad \stackrel{\text{def}}{=} \quad \mathsf{project}_{\mathsf{ALL}\setminus S}(F).$$

Combined with propositional logic, projection generalizes Boolean quantification, combined with first-order logic second-order quantification: The second-order formula $\exists p\, F$, where p is a predicate, can be expressed as projection of F onto the set of all ground literals with a predicate other than p, or equivalently, as the forgetting about the set of all ground literals with predicate p. Intuitively, the projection of a formula F onto scope S is a formula that expresses about literals in S the same as F, but expresses nothing about other literals.

Recall that a propositional formula is in negation normal form if $\wedge$ and $\vee$ are the only allowed binary connectives, and negation $\neg$ is only allowed in front of atoms. We say that a literal $+A$ does occur in such a formula if and only if there is an unnegated occurrence of A, and analogously, that $-A$ does occur in it if and only if there is a negated occurrence. A projection of a propositional formula is equivalent to a propositional formula in negation normal form such that all literals occurring in the formula are members of the projection scope. Such a formula is a *uniform interpolant* of the original formula with respect to the scope. A naive way to construct such an interpolant – or, in other words, to eliminate the projection operator – is indicated by the following equivalences, which hold for propositional formulas F and atoms A, where $F[A\backslash\top]$ ($F[A\backslash\bot]$,

resp.) denotes F with all occurrences of atom A replaced by $\top$ ($\bot$, resp.):

$$\mathsf{forget}_{\{A\}}(F) \equiv F[A\backslash\top] \vee F[A\backslash\bot]. \tag{E1}$$

$$\mathsf{forget}_{\{+A\}}(F) \equiv F[A\backslash\top] \vee (\neg A \wedge F[A\backslash\bot]). \tag{E2}$$

$$\mathsf{forget}_{\{-A\}}(F) \equiv (A \wedge F[A\backslash\top]) \vee F[A\backslash\bot]. \tag{E3}$$

The particular variants of projection and forgetting specified in Def. 2 and 3 are also called *literal projection* and *literal forgetting* [Wer08,LLM03], since they allow, so-to-speak, to express quantification upon just atom occurrences with positive or negative polarity in a formula. This can be contrasted with *atom projection* and *atom forgetting*, where the polarity is not taken into account, and which can be expressed by literal projection and literal forgetting, respectively, onto atom scopes.

The following proposition gives an overview on basic properties of projection. Most of them follow straightforwardly from the semantic definition of $\mathsf{project}$. Proofs, as well as more thorough material on projection can be found in [Wer08,Wer09]. In the subsequent sections we will show further properties of projection and define additional logic operators in terms of projection.

Proposition 4 (Basic Properties of Projection). *For all formulas F, G and scopes S, S_1, S_2 the following properties hold:*

(i) $F \models \mathsf{project}_S(F)$.
(ii) *If* $F \models G$, *then* $\mathsf{project}_S(F) \models \mathsf{project}_S(G)$.
(iii) *If* $F \equiv G$, *then* $\mathsf{project}_S(F) \equiv \mathsf{project}_S(G)$.
(iv) *If* $S_1 \supseteq S_2$, *then* $\mathsf{project}_{S_1}(F) \models \mathsf{project}_{S_2}(F)$.
(v) $\mathsf{project}_{S_2}(\mathsf{project}_{S_1}(F)) \equiv \mathsf{project}_{S_1 \cap S_2}(F)$.
(vi) $F \models \mathsf{project}_S(G)$ *iff* $\mathsf{project}_S(F) \models \mathsf{project}_S(G)$.
(vii) $\mathsf{project}_{\mathsf{ALL}}(F) \equiv F$.
(viii) F *is satisfiable iff* $\mathsf{project}_S(F)$ *is satisfiable.*
(ix) *If no instance of* L *is in* S, *then* $\mathsf{project}_S(L) \equiv \top$.
(x) *If all instances of* L *are in* S, *then* $\mathsf{project}_S(L) \equiv L$.
(xi) $\mathsf{project}_S(\top) \equiv \top$.
(xii) $\mathsf{project}_S(\bot) \equiv \bot$.
(xiii) $\mathsf{project}_S(F \vee G) \equiv \mathsf{project}_S(F) \vee \mathsf{project}_S(G)$.
(xiv) $\mathsf{project}_S(F \wedge G) \models \mathsf{project}_S(F) \wedge \mathsf{project}_S(G)$.
(xv) $\mathsf{project}_S(\exists x F) \equiv \exists x\, \mathsf{project}_S(F)$.
(xvi) $\mathsf{project}_S(\forall x F) \models \forall x\, \mathsf{project}_S(F)$.
(xvii) $\mathsf{project}_{\overline{S}}(\neg\mathsf{project}_S(F)) \equiv \neg\mathsf{project}_S(F)$.

5 Projection onto the Empty Set

Projection onto the empty set has a special relationship to satisfiability, validity and can be applied to specify of sets tuples of constants that represent the variable assignments that satisfy a formula. We define the operator sat for projection onto the empty set, together with the dual operator valid:

Definition 5 (Sat, Valid). For formulas F define:

(i) $\mathsf{sat}(F) \stackrel{\text{def}}{=} \mathsf{project}_{\emptyset}(F)$.
(ii) $\mathsf{valid}(F) \stackrel{\text{def}}{=} \neg\mathsf{project}_{\emptyset}(\neg F)$.

For all sentences F it holds that F is satisfiable if and only if $\mathsf{sat}(F)$ is valid if and only if $\mathsf{sat}(F)$ is satisfiable. Analogously, for all sentences F it holds that F is valid if and only if $\mathsf{valid}(F)$ is satisfiable if and only if $\mathsf{valid}(F)$ is valid.

The operators sat and valid can also be applied to formulas with free variables. Let $F(\boldsymbol{x})$ be a formula. Then

$$\{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models \mathsf{sat}(F(\mathbf{c}))\} \tag{E4}$$

is the set of all tuples $\mathbf{c}$ of constants such that $F(\mathbf{c})$ is satisfiable. Concerning the notation used here, recall that we have defined $\boldsymbol{x}$ as shorthand for $x_1, \ldots, x_n$, thus n is the arity of $\boldsymbol{x}$. In addition, we use for tuples of terms the same notation that we use for sequences of terms. The set of all tuples $\mathbf{c}$ such that $F(\mathbf{c})$ is valid can then be written, analogously to (E4), as:

$$\{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models \mathsf{valid}(F(\boldsymbol{x})).\} \tag{E5}$$

6 Symbolic Notation in Proofs

Most of the material developed in the subsequent sections is accompanied by detailed proofs, where we use the following additional symbolic shorthands and abbreviations:

$\dot{\forall} F \Subset S:$	For all formulas F such that $F \Subset S$ it holds that
$\dot{\exists} F \Subset S:$	There exists a formula F such that $F \Subset S$ and
$\dot{\wedge}$	and
$\Rightarrow$	implies
$\Leftrightarrow$	if and only if
exp.	expanding the definition of
con.	contracting the definition of

7 Aboutness of a Formula

The following notation provides a semantic account for expressing that a formula is "in" a scope, or, in other words, just "about" literals in a scope:

Definition 6 ($\Subset$). For formulas F and scopes S define:

$$F \Subset S \;\text{ iff}_{\text{def}}\; F \equiv \mathsf{project}_S(F).$$

We use the symbol $\Subset$ also when introducing variables, e.g., "let $F \Subset S$ be a formula" for "let F be a formula such that $F \Subset S$". When used in this way, it applies only to the single formula that directly precedes it, for example, "for all $F, G \Subset S$" stands for "for all F and for all $G \Subset S$." If F is a propositional formula,

then $F \Subset S$ holds if and only if F is equivalent to some formula in negation normal form in which only literals from scope S do occur. We express $F \Subset S$ verbally as *F is in scope S*. The following two propositions show properties that are useful in proofs and involve $\Subset$.

Proposition 7 (Scope Closure and Negation). *If F is a formula and S is a scope, then*

$$\neg F \Subset \overline{S} \text{ if and only if } F \Subset S.$$

Proof (Proposition 7). Consider the following tables. We show that the right side of the stated equivalence follows from the assumption of the left side, and vice versa. Left-to-right:

(1)	$\neg F \Subset \overline{S}$	assumption
(2)	$\neg F \equiv \mathsf{project}_{\overline{S}}(\neg F)$.	by (1), exp. $\Subset$
(3)	$F \equiv \neg\mathsf{project}_{\overline{S}}(\neg F)$.	by (2)
(4)	$F \equiv \mathsf{project}_{S}(\neg\mathsf{project}_{\overline{S}}(\neg F))$.	by (3), Prop. 4.xvii
(5)	$F \equiv \mathsf{project}_{S}(F)$.	by (4), (3)
(6)	$F \Subset S$.	by (5), con. $\Subset$

The right-to-left direction is analogous:

(7)	$F \Subset S$.	assumption
(8)	$F \equiv \mathsf{project}_{S}(F)$.	by (7), exp. $\Subset$
(9)	$\neg F \equiv \neg\mathsf{project}_{S}(F)$.	by (8)
(10)	$\neg F \equiv \mathsf{project}_{\overline{S}}(\neg\mathsf{project}_{S}(F))$.	by (9), Prop. 4.xvii
(11)	$\neg F \equiv \mathsf{project}_{\overline{S}}(\neg F)$.	by (10), (9)
(12)	$\neg F \Subset \overline{S}$.	by (11), con. $\Subset$

□

Proposition 8 (Modifying Models Outside the Formula Scope). *Let S be a scope, let F be a formula, let I, J be structures and let β be an assignment. If $F \Subset S$, $\langle I, \beta\rangle \models F$, and $I \cap S \subseteq J$, then $\langle J, \beta\rangle \models F$.*

Proof (Proposition 8). Steps to derive the conclusion of the proposition from assuming its preconditions are shown in the following table:

(1)	$F \Subset S$.	assumption
(2)	$\langle I, \beta\rangle \models F$.	assumption
(3)	$I \cap S \subseteq J$.	assumption
(4)	$\langle I, \beta\rangle \models \mathsf{project}_{S}(F)$.	by (2), (1)
(5)	There exists a structure K such that $\langle K, \beta\rangle \models F$ and $K \cap S \subseteq I$.	by (4)
(6)	There exists a structure K such that $\langle K, \beta\rangle \models F$ and $K \cap S \subseteq J$.	by (5), (3)
(7)	$\langle J, \beta\rangle \models \mathsf{project}_{S}(F)$.	by (6)
(8)	$\langle J, \beta\rangle \models F$.	by (7), (1)

□

8 Further Properties of Projection

In this section we show further properties of projection that are useful for the material developed in later sections. We first turn to the interplay of projection

and conjunction. As we have seen with Prop. 4.xiii, projection distributes over disjunction. From the projection of a conjunction, the projection of the individual conjuncts does follow (Prop. 4.xiv), however, the converse does not hold in general. Proposition 9 shows semantic conditions that allow conclusions in both directions.

A different semantic characterization of the interplay of projection with conjunction shown in [Wer08] involves the additional concept of *essential literal base* and applies in full only to formulas satisfying a certain compactness property, called $\mathcal{E}$-formulas in [Wer08]. For these reasons, we prefer to handle the semantics of the interplay of projection and conjunction with Prop. 9, which suffices for the applications in this report.

Proposition 9 (Projection over Contained Conjunct). *Let F be a formula and let S be a scope. It then holds that*

(i) $\mathsf{project}_{\overline{S}}(F \wedge \mathsf{project}_S(G)) \equiv \mathsf{project}_{\overline{S}}(\mathsf{project}_{\overline{S}}(F) \wedge \mathsf{project}_S(G))$.

(ii) *If S is an atom scope and a $F \Subset S$, then*

$$F \wedge \mathsf{project}_S(G) \equiv \mathsf{project}_S(F \wedge G).$$

Proof (Proposition 9).

(9.i) The left-to-right direction follows from Prop. 4.i and 4.ii. The right-to-left direction can be shown as follows: Consider the table below. Let $\langle I, \beta\rangle$ be a model of the right side, that is, an interpretation such that (1) holds.

(1)	$\langle I, \beta\rangle \models \mathsf{project}_{\overline{S}}(\mathsf{project}_{\overline{S}}(F) \wedge \mathsf{project}_S(G))$.	assumption
(2)	There exist structures J, K, K' such that:	by (1), exp. project
(3)	$\langle K, \beta\rangle \models F$,	
(4)	$K \cap \overline{S} \subseteq J$,	
(5)	$\langle K', \beta\rangle \models G$,	
(6)	$K' \cap S \subseteq J$,	
(7)	$J \cap \overline{S} \subseteq I$.	
(8)	$J \cap S \subseteq K$.	by (4), properties of structures
(9)	$K' \cap S \subseteq K$.	by (8), (6)
(10)	$K \cap \overline{S} \Subset I$.	by (4), (7)
(11)	$\langle I, \beta\rangle \models \mathsf{project}_{\overline{S}}(F \wedge \mathsf{project}_S(G))$.	by (3), (5), (9), (10), con. project

(9.ii) The equivalence can be shown in the following steps, proceeding from the right to the left side:

(1)	$\mathsf{project}_S(F \wedge G)$	
(2)	$\equiv \mathsf{project}_S(\mathsf{project}_S(F) \wedge G)$	
(3)	$\equiv \mathsf{project}_S(\mathsf{project}_S(F) \wedge \mathsf{project}_S(G)$	by Prop. 9.i
(4)	$\equiv \mathsf{project}_S(F) \wedge \mathsf{project}_S(G)$,	by Prop. 4.ii, 4.v, 4.i
(5)	$\equiv F \wedge \mathsf{project}_S(G)$.	

□

In Prop. 10 below we give a variant of the basic property Prop. 4.vi with flipped antecedent and consequent, which is useful in proofs.

Proposition 10 (Consequences of a Projection). *If F, G are formulas and S is a scope, then*

$$\mathsf{project}_S(F) \models G \text{ if and only if } F \models \neg\mathsf{project}_{\overline{S}}(\neg G).$$

Proof (Proposition 10). Consider the following equivalences:

(1)	$\mathsf{project}_S(F) \models G$	
(2)	iff $\neg G \models \neg\mathsf{project}_S(F)$	
(3)	iff $\neg G \models \mathsf{project}_{\overline{S}}(\neg\mathsf{project}_S(F))$	by Prop. 4.xvii
(4)	iff $\mathsf{project}_{\overline{S}}(\neg G) \models \mathsf{project}_{\overline{S}}(\neg\mathsf{project}_S(F))$	by Prop. 4.vi
(5)	iff $\mathsf{project}_{\overline{S}}(\neg G) \models \neg\mathsf{project}_S(F)$	by Prop. 4.xvii
(6)	iff $\mathsf{project}_S(F) \models \neg\mathsf{project}_{\overline{S}}(\neg G)$	
(7)	iff $\mathsf{project}_S(F) \models \mathsf{project}_S(\neg\mathsf{project}_{\overline{S}}(\neg G))$	by Prop. 4.xvii
(8)	iff $F \models \mathsf{project}_S(\neg\mathsf{project}_{\overline{S}}(\neg G))$	by Prop. 4.vi
(9)	iff $F \models \neg\mathsf{project}_{\overline{S}}(\neg G)$.	by Prop. 4.xvii

□

Proposition 11 below gives alternate characterizations of entailment and equivalence of formulas after projection, along with versions that in addition involve negation.

Proposition 11 (Entailment of Projections and Projection). *If F, G are formulas and S is a scope, then*

(i) *The following statements are equivalent:*

1. *For all formulas $H \Subset S$ it holds that if $F \models H$, then $G \models H$.*
2. *$G \models \mathsf{project}_S(F)$.*

(ii) *The following statements are equivalent:*

1. *For all formulas $H \Subset S$ it holds that $F \models H$ if and only if $G \models H$.*
2. *$\mathsf{project}_S(F) \equiv \mathsf{project}_S(G)$.*

(iii) *The following statements are equivalent:*

1. *For all formulas $H \Subset S$ it holds that if $H \models F$, then $H \models G$.*
2. *$\neg\mathsf{project}_{\overline{S}}(\neg F) \models G$.*

(iv) *The following statements are equivalent:*

1. *For all formulas $H \Subset S$ it holds that $H \models F$ if and only if $H \models G$.*
2. *$\mathsf{project}_{\overline{S}}(\neg F) \equiv \mathsf{project}_{\overline{S}}(\neg G)$.*

Proof (Proposition 11).

(11.i) Assume the left side of the proposition, that is, for all formulas $H \Subset S$ it holds that if $F \models H$ then $G \models H$. Since $\mathsf{project}_S(F) \equiv \mathsf{project}_S(\mathsf{project}_S(F))$ (by Prop. 4.v) and $F \models \mathsf{project}_S(F)$ (by Prop. 4.i) it follows that $G \models \mathsf{project}_S(F)$.

Right-to-left: Assume the right side of the proposition, that is, $G \models \mathsf{project}_S(F)$. Let $H \Subset S$ be a formula such that $F \models H$. From Prop. 4.vi it then follows that $\mathsf{project}_S(F) \models H$. From the assumption it then follows that $G \models H$.

(11.ii) Follows from Prop. 11.i and 4.iii.

(11.iii) For all formulas $H \Subset S$, the following statements are equivalent:

(1) $H \models F$, then $H \models G$
(2) iff If $\mathsf{project}_S(H) \models F$, then $H \models G$
(3) iff If $H \models \neg\mathsf{project}_{\overline{S}}(\neg F)$, then $H \models G$ by Prop. 10
(4) iff If $\mathsf{project}_{\overline{S}}(\neg F) \models \neg H$, then $\neg G \models \neg H$.

Thus, also the following statements are equivalent:

(5) For all formulas H s.th. $H \Subset S$:
If $H \models F$, then $H \models G$
(6) iff For all formulas H s.th. $H \equiv \mathsf{project}_{\overline{S}}(H)$:
If $\mathsf{project}_{\overline{S}}(\neg F) \models H$, then $\neg G \models H$ by equiv. of (1) to (3), Prop. 7
(7) iff $\neg G \models \mathsf{project}_{\overline{S}}(\neg F)$. by Prop. 11.i and 4.v
(8) iff $\neg\mathsf{project}_{\overline{S}}(\neg F) \models G$.

(11.iv) Follows from Prop. 11.iii and 4.iii.

□

Chapter 3

Definability and Related Concepts

Our basic tools for approaching queries and answers are the concepts of *definition* and *definability*. Both of them can be characterized in terms of two particular application patterns of projection, the *globally strongest necessary condition* and *globally weakest sufficient condition*. In this chapter we develop this "intermediate layer" between the general projection operator and the analysis of queries and answers.

9 Globally Strongest Necessary and Weakest Sufficient Condition

The dual concepts *globally strongest necessary condition (GSNC)* and *globally weakest sufficient condition (GWSC)* [Wer12] are application patterns of projection that arise in application such as non-monotonic reasoning [Wer12], characterizing abductive explanations [Wer13] and characterizing definability and its facets, which is the focus here. We define GSNC and GWSC as second-order operators that expand into projection. They are closely related to *strongest necessary conditions* and *weakest sufficient conditions*, devised in [Lin01] for propositional logic and adapted to first-order logic in [DŁS01]. As shown in [Wer12], aside of the consideration of polarity, the main difference to the variants introduced in [Lin01] is that for a given formula and scope only the "global" variants are unique up to equivalence. This justifies to speak of *the* GSNC and *the* GWSC.

Definition 12 (GSNC/GWSC). For scopes S and formulas F define:

(i) $\mathsf{gsnc}_S(F, G) \stackrel{\text{def}}{=} \mathsf{project}_S(F \wedge G)$.

(ii) $\mathsf{gwsc}_S(F, G) \stackrel{\text{def}}{=} \neg\mathsf{project}_{\overline{S}}(F \wedge \neg G)$.

The following proposition gathers properties of the GSNC and GWSC. They are straightforward to prove from the definitions of GSNC and GWSC and properties of projection.

Proposition 13 (Properties of GSNC/GWSC). *For all scopes S and formulas F, G it holds that*

(i) $\mathsf{gsnc}_S(F, G) \equiv \neg\mathsf{gwsc}_{\overline{S}}(F, \neg G)$.

(ii) $\mathsf{gwsc}_S(F, G) \equiv \neg\mathsf{gsnc}_{\overline{S}}(F, \neg G)$.

(iii) *If* $F_1 \equiv F_2$ *and* $G_1 \equiv G_2$, *then* $\mathsf{gsnc}_S(F_1, G_1) \equiv \mathsf{gsnc}_S(F_2, G_2)$.

(iv) *If* $F_1 \equiv F_2$ *and* $G_1 \equiv G_2$, *then* $\mathsf{gwsc}_S(F_1, G_1) \equiv \mathsf{gwsc}_S(F_2, G_2)$.

(v) $F \models G \to \mathsf{gsnc}_S(F, G)$.

(vi) $F \models \mathsf{gwsc}_S(F, G) \to G$.

(vii) $H \equiv \mathsf{gsnc}_S(F, G)$ *if and only if:*
1. $H \Subset S$.
2. $F \models G \rightarrow H$.
3. *For all formulas* $H' \Subset S$ *such that* $F \models G \rightarrow H'$ *it holds that* $H \models H'$.

(viii) $H \equiv \mathsf{gwsc}_S(F, G)$ *if and only if:*
1. $H \Subset S$.
2. $F \models H \rightarrow G$.
3. *For all formulas* $H' \Subset S$ *such that* $F \models H' \rightarrow G$ *it holds that* $H' \models H$.

(ix) $\mathsf{gsnc}_S(F, G) \Subset S$.

(x) $\mathsf{gwsc}_S(F, G) \Subset S$.

(xi) *If* $F \models H$, *then* $\mathsf{gsnc}_S(F, G) \models \mathsf{gsnc}_S(H, G)$.

(xii) *If* $H \models F$, *then* $\mathsf{gwsc}_S(F, G) \models \mathsf{gwsc}_S(H, G)$.

(xiii) *If* $G \models H$, *then* $\mathsf{gsnc}_S(F, G) \models \mathsf{gsnc}_S(F, H)$.

(xiv) *If* $G \models H$, *then* $\mathsf{gwsc}_S(F, G) \models \mathsf{gwsc}_S(F, H)$.

(xv) $F \wedge \mathsf{gwsc}_S(F, G) \models \mathsf{gsnc}_{\overline{S}}(F, G)$.

GSNC and GWSC are inter-definable (Prop. 13.i and 13.ii). Each of them is a "semantic" operator, that is, for equivalent arguments, the values are equivalent (Prop. 13.iii and 13.iv). The GSNC is a "necessary condition" of formulas F and G, and analogously, the GWSC is a "sufficient condition" of F and G, (Prop. 13.v and 13.vi). The GSNC can be characterized as the strongest "necessary condition" of formulas F and G with respect to scope S, that is, the strongest formula $H \Subset S$ such that $F \models G \rightarrow H$ (Prop. 13.vii). Analogously, the GWSC can be characterized as the weakest "sufficient condition" of F and G with respect to S, that is, the weakest formula $H \Subset S$ such that $F \models H \rightarrow G$ (Prop. 13.viii). The GSNC as well as the GWSC with respect to scope S are in scope S (Prop.13.ix and 13.x). In the first argument, the GSNC is monotonic, while the GWSC is antimonotonic (Prop. 13.xi and 13.xii). Both operators are monotonic in their second argument (Prop. 13.xiii and 13.xiv). When combined with the base formula, the GWSC entails the GSNC after flipping the polarity of the literals in the scope (Prop. 13.xv).

The following property relates the entailment of GWSCs to the conditional entailment of their argument formulas.

Proposition 14 (Entailment of GWSCs and of Arguments). *For all formulas* F_1, F_2, G_1, G_2 *and scopes* S *it holds that if* $\mathsf{gwsc}_S(F_1, G_1) \models \mathsf{gwsc}_S(F_2, G_2)$ *and* $F_1 \models G_1$, *then* $F_2 \models G_2$.

Proof. Consider the following table. Assume the preconditions of the proposition, steps (1) and (2). With step (7) we derive the conclusion.

(1)	$\mathsf{gwsc}_S(F_1, G_1) \models \mathsf{gwsc}_S(F_2, G_2)$.	assumption
(2)	$F_1 \models G_1$.	assumption
(3)	$\neg\mathsf{project}_S(F_1 \wedge \neg G_1) \models \neg\mathsf{project}_S(F_2 \wedge \neg G_2)$.	by (1), exp. gwsc
(4)	$F_1 \wedge G_1 \equiv \bot$.	by (2)
(5)	$\models \neg\mathsf{project}_S(F_2 \wedge \neg G_2)$.	by (4), (3), Prop. 4.xii
(6)	$\models \neg(F_2 \wedge \neg G_2)$.	by (5), Prop. 4.i
(7)	$F_2 \models \neg G_2$.	by (6)

□

Concluding conversely to Prop. 14 from implication of entailments to entailment of the GWSCs is not in general possible: Since $\mathsf{p}(\mathsf{a}) \not\models \mathsf{p}(x)$ it holds that if $\mathsf{p}(\mathsf{a}) \models \mathsf{p}(x)$, then $\mathsf{p}(\mathsf{b}) \models \mathsf{p}(x)$. Any interpretation $\langle I, \beta\rangle$ such that $\beta(x) = \mathsf{a}$ is then a model of $\mathsf{gwsc}_{\emptyset}(\mathsf{p}(\mathsf{a}), \mathsf{p}(x))$, which is equivalent to $x \doteq a$, but not of $\mathsf{gwsc}_{\emptyset}(\mathsf{p}(\mathsf{b}), \mathsf{p}(x))$, which is equivalent to $x \doteq b$.

Proposition 15 below relates projection to entailment of GWSCs, analogously as Prop. 11.iii and 11.iv relate projection to entailment from formulas that are in a given scope.

Proposition 15 (Entailment of GWSCs and Projection). *If F, G are formulas and S, T are scopes such that $T \subseteq S$, then*

(i) *The following statements are equivalent:*

1. *For all $H \Subset S$ it holds that $\mathsf{gwsc}_T(H, F) \models \mathsf{gwsc}_T(H, G)$.*
2. *$\neg\mathsf{project}_{\overline{S}}(\neg F) \models G$.*

(ii) *The following statements are equivalent:*

1. *For all $H \Subset S$ it holds that $\mathsf{gwsc}_T(H, F) \equiv \mathsf{gwsc}_T(H, G)$.*
2. *$\mathsf{project}_{\overline{S}}(\neg F) \equiv \mathsf{project}_{\overline{S}}(\neg G)$.*

Proof. (15.i) Left-to-right: Assume statement (1.). From Prop. 14 if follows that for all $H \Subset S$ it holds that if $H \models F$, then $H \models G$. By Prop. 11.iii it follows that $\neg\mathsf{project}_{\overline{S}}(\neg F) \models G$, that is, statement (2.).

Right-to-left. Consider the following table. Assume steps (1)–(4), that is, assume the precondition of the propositions, statement (2.), let $H \Subset S$ be a formula, and let $\langle I, \beta\rangle$ be a model of the left side of the statement (1.).

(1)	$T \subseteq S$.	assumption
(2)	$\neg\mathsf{project}_{\overline{S}}(\neg F) \models G$.	assumption
(3)	$H \Subset S$.	assumption
(4)	$\langle I, \beta\rangle \models \mathsf{gwsc}_T(H, F)$.	assumption
(5)	$\langle I, \beta\rangle \models \neg\mathsf{project}_{\overline{T}}(H \wedge \neg F)$.	by (4), exp. gwsc
(6)	$\langle I, \beta\rangle \models \neg\mathsf{project}_{\overline{T}}(\mathsf{project}_{\overline{S}}(H \wedge \neg F))$.	by (5), (1), Prop. 4.v
(7)	$\langle I, \beta\rangle \models \neg\mathsf{project}_{\overline{T}}(\mathsf{project}_{\overline{S}}(H \wedge \mathsf{project}_{\overline{S}}(\neg F)))$.	by (6), (3), Prop. 9.i
(8)	$\langle I, \beta\rangle \models \neg\mathsf{project}_{\overline{T}}(H \wedge \mathsf{project}_{\overline{S}}(\neg F))$.	by (7), (1), Prop. 4.v
(9)	$\langle I, \beta\rangle \models \neg\mathsf{project}_{\overline{T}}(H \wedge \neg G)$.	by (8), (2), Prop. 4.ii
(10)	$\langle I, \beta\rangle \models \mathsf{gwsc}_T(H, G)$.	by (9), con. gwsc

(15.ii) Follows from Prop. 15.i with Prop. 4.vi.

□

Further properties of GSNC and GWSC that are related to the concept of *definition* are shown as Prop. 17 in next section.

10 Characterization of Definitions and Definability

We specify the concept of *definition of a formula in terms of a scope within a formula* semantically as follows:

Definition 16 (Definition). Let S be a scope and let F, G be formulas. A formula H is a called a *definition* of G in terms of S within F if and only if

1. $H \Subset S$,
2. $\mathsf{gsnc}_S(F, G) \models H$, and
3. $H \models \mathsf{gwsc}_S(F, G)$.

In Prop. 18 below we show that Def. 16 indeed captures the intuitive concept of *definition*. The proof of that proposition resides on the following properties of GSNC/GWSC:

Proposition 17 (Sufficient and Necessary Conditions). *For all scopes S and formulas F, G and $H \Subset S$ it holds that*

(i) $F \models G \rightarrow H$ *if and only if* $\mathsf{gsnc}_S(F, G) \models H$.
(ii) $F \models H \rightarrow G$ *if and only if* $H \models \mathsf{gwsc}_S(F, G)$.
(iii) $F \models H \leftrightarrow G$ *if and only if* $\mathsf{gsnc}_S(F, G) \models H$ *and* $H \models \mathsf{gwsc}_S(F, G)$.

Proof (Proposition 17).

(17.i) Consider the following equivalences:

(1) $\mathsf{gsnc}_S(F, G) \models H$
(2) iff $\mathsf{project}_S(F \wedge G) \models H$
(3) iff $F \wedge G \models H$ by the precondition $H \Subset S$ and Prop. 4.vi
(4) iff $F \models G \rightarrow H$.

(17.ii) Consider the following equivalences:

(1) $H \models \mathsf{gwsc}_S(F, G)$
(2) iff $H \models \neg\mathsf{project}_{\overline{S}}(F \wedge \neg G)$
(3) iff $\mathsf{project}_{\overline{S}}(F \wedge \neg G) \models \neg H$
(4) iff $F \wedge \neg G \models \neg H$ by the precondition $H \Subset S$ and Prop. 4.vi, 4.xvii
(5) iff $F \models H \rightarrow G$.

(17.iii) Immediate from Prop. 17.ii and 17.i. □

Proposition 18 (Characteristics of a Definition). *Let S be a scope and let F, G be formulas. A formula H is a* definition *of G in terms of S within F if and only if*

1. $H \Subset S$, *and*
2. $F \models H \leftrightarrow G$.

Proof (Proposition 18). Immediate from Prop. 17.iii. □

Notice that, although condition (2.) in Prop. 18 has the shape of a biconditional, the characterization in that proposition, like the equivalent Def. 16, corresponds to implicit rather than explicit definability.

From Prop. 18 it is easy to see that definitions of predicates with arguments are covered as special cases by our characterizations: If $G(\boldsymbol{x})$ and $H(\boldsymbol{x})$ are formulas with free variables $\boldsymbol{x}$ and F does not contain free occurrences of these variables, then

$$F \models \forall \boldsymbol{x}\, H(\boldsymbol{x}) \leftrightarrow G(\boldsymbol{x}) \;\text{ iff }\; F \models H(\boldsymbol{x}) \leftrightarrow G(\boldsymbol{x}). \tag{E6}$$

The concept of *definability* can be specified analogously to *definition* in terms of GSNC/GWSC:

Definition 19 (Definable). Let S be a scope and let F, G be formulas. Then G is called *definable* in terms of S within F if and only if

$$\mathsf{gsnc}_S(F, G) \models \mathsf{gwsc}_S(F, G).$$

The intuitive characterization of *definable* as existence of a definition is shown by the following proposition:

Proposition 20 (Characteristics of Definability). *Let S be a scope and let F, G be formulas. Then G is definable in terms of S within F if and only if there exists a definition of G in terms of S within F.*

Proof (Proposition 19). Left-to-right. Assume definability, that is, $\mathsf{gsnc}_S(F, G) \models \mathsf{gwsc}_S(F, G)$. Now $\mathsf{gsnc}_S(F, G)$ is a definition, since in the role of H it satisfies the three conditions stated in Def. 16:

1. $\mathsf{gsnc}_S(F, G) \Subset S$, by Prop. 13.ix.
2. $\mathsf{gsnc}_S(F, G) \models \mathsf{gsnc}_S(F, G)$, holds trivially.
3. $\mathsf{gsnc}_S(F, G) \models \mathsf{gwsc}_S(F, G)$, as assumed.

Right-to-left. Immediate from Def. 16. □

The following proposition states further ways of characterizing *definable*, which involve only either GSNC or GWSC.

Proposition 21 (Definable: Further Characterizations). *Let S be a scope and let F, G be formulas. Then G is definable in terms of S within F if and only if*

(i) $F \wedge \mathsf{gsnc}_S(F, G) \models G$.
(ii) $F \wedge G \models \mathsf{gwsc}_S(F, G)$.

Proof (Proposition 21). Consider the equivalences shown in the tables below.

(21.i)

(1) $\mathsf{gsnc}_S(F, G) \models \mathsf{gwsc}_S(F, G)$
(2) iff $\mathsf{project}_S(F \wedge G) \models \neg\mathsf{project}_{\overline{S}}(F \wedge \neg G)$
(3) iff $\mathsf{project}_{\overline{S}}(F \wedge \neg G) \models \neg\mathsf{project}_S(F \wedge G)$
(4) iff $F \wedge \neg G \models \neg\mathsf{project}_S(F \wedge G)$ by Prop. 4.xvii, 4.vi
(5) iff $F \wedge \mathsf{project}_S(F \wedge G) \models G$
(6) iff $F \wedge \mathsf{gsnc}_S(F, G) \models G$.

(21.ii)

(1) $\mathsf{gsnc}_S(F, G) \models \mathsf{gwsc}_S(F, G)$
(2) iff $\mathsf{project}_S(F \wedge G) \models \mathsf{gwsc}_S(F, G)$
(3) iff $F \wedge G \models \mathsf{gwsc}_S(F, G)$. by Prop. 13.x, 4.vi

□

If definability holds, then the GSNC and the GWSC themselves are both definitions:

Proposition 22 (GSNC/GWSC as Definitions). *Let S be a scope and let F, G be formulas such that G is definable in terms of S within F. Then the following formulas are definitions of G in terms of S within F:*

(i) $\mathsf{gsnc}_S(F, G)$.
(ii) $\mathsf{gwsc}_S(F, G)$.

Proof (Proposition 22). Assume that G is definable as stated in the precondition of the proposition, that is, $\mathsf{gsnc}_S(F, G) \models \mathsf{gwsc}_S(F, G)$. By Prop. 13.ix and 13.x it holds that $\mathsf{gsnc}_S(F, G) \Subset S$ and $\mathsf{gwsc}_S(F, G) \Subset S$. The conditions of Def. 16 can then be easily verified for formulas $\mathsf{gsnc}_S(F, G)$ and $\mathsf{gwsc}_S(F, G)$ in the role of H. □

11 Unique Definability

We have seen that in case of definability all formulas in scope S that are, with respect to entailment, between the GSNC and the GWSC, including GSNC and GWSC themselves, are definitions (Def. 16 and Prop. 22). So far, it is well possible that a formula has different definitions that are not semantically equivalent. The following definition of *uniquely definable* characterizes the case where a formula has exactly one definition, modulo equivalence.

Definition 23 (Uniquely Definable). Let S be a scope and let F, G be formulas. Then G is called *uniquely definable* in terms of S within F if and only if

$$\mathsf{gsnc}_S(F, G) \equiv \mathsf{gwsc}_S(F, G).$$

The characteristic property of unique definability is as follows:

Proposition 24 (Characteristics of Unique Definability). *Let S be a scope and let F, G be formulas. Then G is uniquely definable in terms of S within F if and only if there exists a definition H of G in terms of S within F such that for all definitions H' of G in terms of S within F it holds that $H \equiv H'$.*

Proof (Proposition 24). We understand *definable* and *definition* here implicitly with respect to the parameters S and F.

Left-to-right. Assume the left side of the proposition, that is, $\mathsf{gsnc}_S(F, G) \equiv \mathsf{gwsc}_S(F, G)$. By Def. 19 then G is definable, thus, by Prop. 20 there exists a definition H of G. Let H'' be an arbitrary definition of G. It then holds that

$\mathsf{gsnc}_S(F,G) \models H''$ and $H'' \models \mathsf{gwsc}_S(F,G)$. With our assumption $\mathsf{gsnc}_S(F,G) \equiv \mathsf{gwsc}_S(F,G)$ this implies $\mathsf{gwsc}_S(F,G) \equiv H''$. Since this applies to arbitrary definitions H'' of G, we can conclude that for all definitions H' of G it holds that $H \equiv H'$.

Right-to-left. Assume that there exists a definition H of G such that for all definitions H' of G it holds that $H \equiv H'$. The formula G is then definable, and from Prop. 22 follows that the formulas $\mathsf{gsnc}_S(F,G)$ and $\mathsf{gwsc}_S(F,G)$ are both definitions of G. Since then each of these two formulas must be equivalent to H, it follows that $\mathsf{gsnc}_S(F,G) \equiv \mathsf{gwsc}_S(F,G)$. □

The following example shows a case where definability holds, but *unique* definability fails.

Example 25 (Uniquely Definable: A Counterexample). Let $S = \{\mathsf{r},\mathsf{s}\}$ and let

$$F = ((\mathsf{p} \leftrightarrow \mathsf{r} \wedge \mathsf{s}) \wedge (\mathsf{r} \rightarrow \mathsf{s})).$$

Then $F \models \mathsf{p} \leftrightarrow (\mathsf{r} \wedge \mathsf{s})$ and $F \models \mathsf{p} \leftrightarrow \mathsf{r}$. Thus $(\mathsf{r} \wedge \mathsf{s})$ and r are both definitions of p in terms of S within F. Since $(\mathsf{r} \wedge \mathsf{s}) \not\equiv \mathsf{r}$, it follows that p is not *uniquely* definable in terms of S within F. Let us now consider the relevant characterizations in terms of GSNC and GWSC. It holds that:

$$\mathsf{gsnc}_S(F) \equiv \mathsf{project}_{\{\mathsf{r},\mathsf{s}\}}((\mathsf{p} \leftrightarrow \mathsf{r} \wedge \mathsf{s}) \wedge (\mathsf{r} \rightarrow \mathsf{s}) \wedge \mathsf{p}) \equiv (\mathsf{r} \wedge \mathsf{s}), \text{ and}$$
$$\mathsf{gwsc}_S(F) \equiv \neg\mathsf{project}_{\{\mathsf{r},\mathsf{s}\}}((\mathsf{p} \leftrightarrow \mathsf{r} \wedge \mathsf{s}) \wedge (\mathsf{r} \rightarrow \mathsf{s}) \wedge \neg\mathsf{p}) \equiv \mathsf{r}.$$

Since $(\mathsf{r} \wedge \mathsf{s}) \models \mathsf{r}$, it follows that p is definable in terms of S within F. From Prop. 22 it follows that $(\mathsf{r} \wedge \mathsf{s})$ and r are both definitions in terms of S within F. Since $\mathsf{r} \not\models (\mathsf{r} \wedge \mathsf{s})$ it follows that p is not uniquely definable in terms of S within F.

If definability is given, then for *atom* scopes unique definability follows by a condition that just depends on the scope and the base formula, independently of the particular formula whose unique definability is under consideration:

Proposition 26 (Uniquely Definable: A Further Characterization for Atom Scopes). *Let S be an atom scope and let F, G be formulas. Assume that G is definable in terms of S within F. Then G is uniquely definable in terms of S within F if and only if*

$$\models \mathsf{project}_S(F).$$

Proof (Proposition 26). The formula G is uniquely definable in terms of S within F if and only if

$$\mathsf{gsnc}_S(F,G) \equiv \mathsf{gwsc}_S(F,G). \tag{E7}$$

We assumed definability as precondition, which implies the left-to-right direction of (E7). It thus suffices to show equivalence of the right-to-left direction of (E7) to the statement $\models \mathsf{project}_S(F)$:

(1) $\models \mathsf{project}_S(F)$
(2) iff $\models \mathsf{project}_S((F \wedge G) \vee (F \wedge \neg G))$
(3) iff $\models \mathsf{project}_S(F \wedge G) \vee \mathsf{project}_S(F \wedge \neg G)$ by Prop. 4.xiii
(4) iff $\neg\mathsf{project}_S(F \wedge \neg G) \models \mathsf{project}_S(F \wedge G)$
(5) iff $\mathsf{gwsc}_{\overline{S}}(F, G) \models \mathsf{gsnc}_S(F, G)$
(6) iff $\mathsf{gwsc}_S(F, G) \models \mathsf{gsnc}_S(F, G)$. since S is an atom scope

□

In the special case of definability with respect to the empty scope, the requirement of Prop. 26 for unique definability amounts to satisfiability of the base formula:

Proposition 27 (Uniquely Definable: Characterization for the Empty Scope). *Let F be a sentence and let G be a formula. Assume that G is definable in terms of $\emptyset$ within F. Then G is uniquely definable in terms of $\emptyset$ within F if and only if F is satisfiable.*

Proof (Proposition 27). This follows as the special case of Prop. 26 where $S = \emptyset$ and F is a sentence. As explained in Sect. 5, it holds that $\models \mathsf{project}_\emptyset(F)$ iff $\models \mathsf{sat}(F)$ iff F is satisfiable. □

12 Scope Definability

A typical database can be considered as a formula that provides definitions of a set of predicates, the "database predicates", where these definitions are extensional, that is, in terms of the empty scope. Thus all formulas which are in the scope corresponding to the database predicates are definable within the database in terms of the empty scope. The notion of *scope defining formula* expresses this property of formulas considered as databases, generalized such that the definientia must not necessarily be in the empty scope.

Definition 28 (Scope Defining Formula). A formula F is said to *define* a scope T in terms of a second scope S if and only if for all formulas $G \Subset T$ it holds that G is definable in terms of S within F.

To ensure scope definability, that is, definability of *all* formulas in a given scope it suffices to show just the definability of *all ground literals* in that scope, as shown by the following proposition.

Proposition 29 (From Literal Definability to Scope Definability). *Let S, T be scopes and let F be a formulas such that for all ground literals $L \in T$ it holds that L is definable in terms of S within F. Then F defines T in terms of S within F.*

Proof (Proposition 29). Consider the following table. Assume the precondition of the proposition, that is, step (1). Let $G \Subset T$ be a formula, as stated in step (2). We prove the proposition by showing that G is definable in terms of S within F. Assume that, to the contrary, G is not definable in terms of S within F. Then

there must exist an interpretation $\langle I, \beta \rangle$ such that steps (3) and (4) hold. We derive a contradiction from these assumptions.

(1)	For all $L \in T$: L is definable i.t.o. S w. F.	assumption
(2)	$G \Subset T$.	assumption
(3)	$\langle I, \beta \rangle \models \mathsf{gsnc}_S(F, G)$.	assumption
(4)	$\langle I, \beta \rangle \not\models \mathsf{gwsc}_S(F, G)$.	assumption
(5)	$\langle I, \beta \rangle \models \mathsf{project}_S(F \wedge G)$.	by (3)
(6)	There exists a structure J s.th.:	
(7)	$\quad \langle J, \beta \rangle \models F$,	
(8)	$\quad \langle J, \beta \rangle \models G$,	
(9)	$\quad J \cap S \subseteq I$.	by (5)
(10)	$\langle I, \beta \rangle \models \mathsf{project}_{\overline{S}}(F \wedge \neg G)$.	by (4)
(11)	There exists a structure K s.th.:	
(12)	$\quad \langle K, \beta \rangle \models F$,	
(13)	$\quad \langle K, \beta \rangle \models \neg G$.	by (9)
(14)	$\quad K \cap \overline{S} \subseteq I$.	by (10)
(15)	$J \cap T \not\subseteq K$.	by (13), (8), (2), Prop. 8
(16)	There is a ground literal M s.th.:	
(17)	$\quad M \in J$,	
(18)	$\quad M \in T$,	
(19)	$\quad \overline{M} \in K$.	by (15)
(20)	$\langle J, \beta \rangle \models F \wedge M$.	by (17), (7)
(21)	$\langle I, \beta \rangle \models \mathsf{project}_S(F \wedge M)$.	by (20), (9)
(22)	$\langle I, \beta \rangle \models \mathsf{gsnc}_S(F, M)$.	by (21)
(23)	$\langle K, \beta \rangle \models F \wedge \neg M$.	by (19), (12)
(24)	$\langle I, \beta \rangle \models \mathsf{project}_{\overline{S}}(F \wedge \neg M)$.	by (23), (14)
(25)	$\langle I, \beta \rangle \models \neg\mathsf{gwsc}_S(F, M)$.	by (24)
(26)	$\mathsf{gsnc}_S(F, M) \models \mathsf{gwsc}_S(F, M)$.	by (18), (1)
(27)	contradiction.	by (26), (25), (22)

□

The following concept of *uniquely scope defining formula* strengthens the property specified in Def. 28 to require *unique* definability instead of just definability of all formulas in some scope.

Definition 30 (Uniquely Scope Defining Formula). A formula F is said to *uniquely define* a scope T in terms of a second scope S if and only if for all formulas $G \Subset T$ it holds that G is uniquely definable in terms of S within F.

Analogously to Prop. 29, we would like to conclude *unique* definability of *all* formulas in a given scope from unique definability of *all ground literals* in that scope. However, as Prop. 31 below shows, this applies only under the precondition that the scope of the definientia is an *atom* scope. Example 32 below gives a counterexample for the case where this precondition fails. A different condition for concluding unique definability from definability, which also applies to scopes that are not atom scopes, is given in Prop. 35 further down below.

Proposition 31 (From Unique Literal Definability to Unique Scope Definability). *Let S be an atom scope and let T be a scope that is non-empty. Let F be a formula such that for all ground literals $L \in T$ it holds that L is uniquely definable in terms of S within F. Then F uniquely defines T in terms of S.*

Proof (Proposition 31). Assume the preconditions of the proposition. Let $G \Subset T$ be a formula. Since F is a formula such that for all ground literals $L \in T$ it holds that L is *uniquely* definable in terms of S within F, it also holds for all ground literals $L \in T$ that L is *definable* in terms of S within F. From Prop. 29 it then follows that G is definable in terms of S within F. Given that $T \neq \emptyset$, there must be a ground literal $L \in T$ that is, by the assumed preconditions, uniquely definable in terms of S within F. Since S is an atom scope, it follows from Prop. 26 that $\models \mathsf{project}_S(F)$. Because G is definable in terms of S within F, we can apply Prop. 26 again to conclude that G is uniquely definable in terms of S within F. □

Example 32 (From Unique Literal Definability to Unique Scope Definability: Counterexample for a Proper Literal Scope). Let $U = \{+p, +q\}$, let $B = \{+r, +s\}$, and let

$$V = (p \leftrightarrow r) \wedge (q \leftrightarrow s) \wedge (p \rightarrow \neg q).$$

Then each literal in the scope U is uniquely definable in terms of B within V, which follows since $\mathsf{gsnc}_B(V, p) \equiv \mathsf{gwsc}_B(V, p) \equiv r$ and $\mathsf{gsnc}_B(V, q) \equiv \mathsf{gwsc}_B(V, q) \equiv s$. Let $F = (p \wedge q)$. Clearly $F \Subset U$. However F is not uniquely definable in terms of B within V, which follows since $\mathsf{gwsc}_B(V, F) \equiv (r \wedge s) \not\models \bot \equiv \mathsf{gsnc}_B(V, F)$. Indeed, both $(r \wedge s)$ and $\bot$ are definitions of F in terms of B within V. For the *atom* scope $B' = \{r, s\}$, not all literals in scope U are uniquely definable, since $\mathsf{gwsc}_{B'}(V, p) \equiv r \not\models (r \wedge \neg s) \equiv \mathsf{gsnc}_{B'}(V, p)$.

13 Conservative Formulas

The property *conservative*, specified in the following definition, is closely related to the concept of *conservative extension*, which is used as basis of knowledge base modularization in description logics [GLW06].

Definition 33 (Conservative). Let S be a scope and let F, G be formulas. Then G is called *conservative* for S within F if and only if

$$F \models \mathsf{gsnc}_S(G, F).$$

A semantic analog of the notion of conservative extension can be characterized with *conservative* as follows: A *conservative extension* of F for S is a formula $(F \wedge G)$ such that G is conservative for S with respect to F. "*Semantic* analog" refers here to the semantic characterization involving scopes instead of the usual syntactic characterization in terms of formula signature.

For base formulas that are within the scope considered for conservativeness, and in the particular case that this is an atom scope, conservativeness can be characterized in further ways:

Proposition 34 (Conservativeness within Scope Closed Formulas). *Let S be a scope, and let $F \Subset S$, G be formulas. Then*

(i) *G is conservative for S within F if and only if*

$$\mathsf{gsnc}_S(G, F) \equiv F.$$

(ii) *If S is an atom scope, then G is conservative for S within F if and only if*

$$F \models \mathsf{project}_S(G).$$

Proof (Proposition 34).

(34.i) The left-to-right direction follows, since under the assumption $F \Subset S$ it holds that $\mathsf{gsnc}_S(G, F) \equiv \mathsf{project}_S(G \wedge F) \models \mathsf{project}_S(F) \equiv F$. The right-to-left direction is immediate from Def. 33.

(34.i) Under the assumptions that $F \Subset S$ and that S is an atom scope, by Prop. 9.ii it follows that $\mathsf{gsnc}_S(G, F) \equiv \mathsf{project}_S(G \wedge F) \equiv (\mathsf{project}_S(G) \wedge \mathsf{project}_S(F)) \equiv (\mathsf{project}_S(G) \wedge F)$. Thus $F \models \mathsf{gsnc}_S(G, F)$ holds if and only if $F \models \mathsf{project}_S(G) \wedge F$ if and only if $F \models \mathsf{project}_S(G)$. □

The following proposition shows that conservativeness with respect to all formulas in a scope and definability in terms of that scope together imply unique definability.

Proposition 35 (Conservativeness and Unique Definability). *Let S be a scope and let F, G be formulas such that*

1. *For all formulas $H \Subset S$ it holds that F is conservative for S within H, and*
2. *G is definable in terms of S within F.*

Then G is uniquely definable in terms of S within F.

Proof (Proposition 35). Assume the preconditions of the proposition, that is, (1.) for all formulas $H \Subset S$ it holds that F is conservative for S within H, and (2.) that G is definable in terms of S within F. Assume further that the conclusion does not hold, that is, (3.) that G is not uniquely definable in terms of S within F. From (2.) and (3.) follows that there exist formulas $H_1 \Subset S$ and $H_2 \Subset S$ such that (4.) $H_1 \not\equiv H_2$, $F \models (G \leftrightarrow H_1)$, and $F \models (G \leftrightarrow H_2)$. Hence $F \models (H_1 \leftrightarrow H_2)$, hence $F \wedge H_1 \leftrightarrow F \wedge H_2$, hence by Prop. 4.iii if follows (5.) that $\mathsf{project}_S(F \wedge H_1) \equiv \mathsf{project}_S(F \wedge H_2)$. With (1.) and Prop. 34.i it follows from (5.) that $H_1 \equiv \mathsf{gsnc}_S(F, H_1) \equiv \mathsf{project}_S(F \wedge H_1) \equiv \mathsf{project}_S(F \wedge H_2) \equiv H_2$, which contradicts with step (4.), that is, $H_1 \not\equiv H_2$. □

14 Definability in First-Order Logic as Validity

As shown in [Tar35], within classical first-order logic checking definability in terms of a predicate scope of predicates can be expressed as checking validity. In our framework, this can be shown as follows. Let S be a predicate scope, let F, G be first-order formulas and let P be the predicate scope that contains exactly (all ground literals with) the predicates occurring in F or in G. Let $U = P \setminus S = \{u_1, \ldots, u_n\}$ and let $V = \{v_1, \ldots, v_n\}$ be a predicate scope that is disjoint with $S \cup P$. Now, for a formula E, let E^V denote the formula obtained from E by systematically replacing each predicate u_i with v_i. It then clearly holds that

$$\mathsf{forget}_U(E) \equiv \mathsf{forget}_V(E^V). \tag{E8}$$

From (E8) and properties of projection, we conclude that the following formulas are equivalent:

(1) $\mathsf{gsnc}_S(F, G) \rightarrow \mathsf{gwsc}_S(F, G)$
(2) $\mathsf{project}_S(F \wedge G) \rightarrow \neg\mathsf{project}_S(F \wedge \neg G)$
(3) $\neg(\mathsf{project}_S(F \wedge G) \wedge \mathsf{project}_S(F \wedge \neg G))$
(4) $\equiv \neg(\mathsf{forget}_U(F \wedge G) \wedge \mathsf{forget}_U(F \wedge \neg G))$
(5) $\equiv \neg(\mathsf{forget}_U(F \wedge G) \wedge \mathsf{forget}_V(F^V \wedge \neg G^V))$
(6) $\equiv \neg(\mathsf{forget}_{U\cup V}(F \wedge G \wedge F^V \wedge \neg G^V))$.

Now we can apply equivalence of (6) to (1), as well as properties of projection to conclude that G is definable in terms of S within F if and only if the first-order formula $(F \wedge G \wedge F^V \wedge \neg G^V)$ is valid:

(7) $\mathsf{gsnc}_S(F, G) \models \mathsf{gwsc}_S(F, G)$
(8) iff $\models \mathsf{gsnc}_S(F, G) \rightarrow \mathsf{gwsc}_S(F, G)$
(9) iff $\models \neg\mathsf{forget}_{U\cup V}(F \wedge G \wedge F^V \wedge \neg G^V)$ by equivalence of (6) to (1)
(19) iff $\models \neg\mathsf{forget}_{\mathsf{ALL}}(\mathsf{forget}_{U\cup V}(F \wedge G \wedge F^V \wedge \neg G^V))$
(11) iff $\models \neg\mathsf{forget}_{\mathsf{ALL}}(F \wedge G \wedge F^V \wedge \neg G^V)$
(12) iff $\models \neg(F \wedge G \wedge F^V \wedge \neg G^V)$.

Clearly $\models \neg(F \wedge G \wedge F^V \wedge \neg G^V)$ holds if and only if $F \wedge G \models F^V \rightarrow G^V$. By Craig's interpolation theorem [Cra57], if $F \wedge G \models F^V \rightarrow G^V$, then there exists a first-order formula H whose predicates are in S such that $F \wedge G \models H$ and $H \models F \rightarrow G$. By Prop. 4.vi and 10 we can conclude that $\mathsf{gsnc}_S(F, G) \models H$ and $H \models \mathsf{gwsc}_S(F, G)$, that is, H is a definition of G in terms of S within F. This justifies that if definability holds, then any first-order proving method that allows to extract interpolants from proofs can be successfully applied to compute a definition H.

Chapter 4

A Generic Model of Query Answering

15 Answers as Alternate Definitions of Queries

We basically consider an answer as a reformulation of a query expression that is obtained by taking a knowledge base into account. The reformulation must satisfy certain requirements, for example, involving only expressions that can be understood or efficiently processed by the clients who will receive the answer. The following definition provides a formal frame for this approach.

Definition 36 (Answer). Let B be a sentence and let Q be a formula. An *answer* in terms of *answer scope* S to *query* Q with respect to *knowledge base* B is a formula A such that

1. A is a definition of Q in terms of S within B.
2. A satisfies certain syntactic and further semantic properties.

With the phrase *certain syntactic and further semantic properties*, Def. 36 provides a hook for instantiating it to model particular applications and kinds of answers. Examples for instantiating *syntactic conditions* would be that predicates not in scope S do actually not occur in A, that first-order quantifiers do not occur in A, and that A has some specific syntactic shape such as disjunctive normal form.

For classical semantics, in contrast to non-monotonic logic programming semantics, Def. 36 is quite similar to the generic characterization of *abductive explanation* [KKT98], with the essential difference, that in case of abduction just a *sufficient condition* in place of a definition is required. More precisely, by Prop. 18 we can express condition (1.) of Def. 36 equivalently as $A \Subset S$ *and* $F \models A \leftrightarrow Q$. If we replace condition (1.) with the weaker condition $A \Subset S$ *and* $A \models \mathsf{gwsc}_S(B, Q)$ or, equivalently, with $A \Subset S$ *and* $F \models A \rightarrow Q$, then we obtain a characterization of *abductive explanation* A for *observation* Q with respect to the *theory presentation* B. A detailed comparison with [KKT98] shows that specializations on our characterization of abductive explanation as well as that of [KKT98] would be needed to obtain a precise matching: The condition $A \Subset S$ included inherently in our condition (1.) of Def. 36 must in the specification according to [KKT98] be stated as *additional criteria*, and the requirement of [KKT98] that $(B \wedge A)$ must be consistent has to be assumed as always present in our condition (2.).

16 Extensional Answers

For data- and knowledge-base systems, typically "extensional" answers are considered, that is, representations of the set of all assignments of the free variables in the query for which the corresponding instance of the query is *consistent* with the knowledge base or *entailed* by the knowledge base. For some important classes of restricted knowledge bases, such as relational databases, the consistency-based and the entailment-based notion coincide. Both types answers are defined in terms of GSNC or GWSC, respectively:

Definition 37 (Extensional Answers). Let B and Q be formulas. We define:

(i) The *consistency-based extensional answer* to Q with respect to B is

$$\mathsf{gsnc}_{\emptyset}(B, Q).$$

(ii) The *entailment-based extensional answer* to Q with respect to B is

$$\mathsf{gwsc}_{\emptyset}(B, Q).$$

If Q is definable in terms of $\emptyset$ within B, then, by Prop. 22, both, the consistency- as well as the entailment-based answer, provide definitions of Q in terms of $\emptyset$ within B. Both these extensional answers are thus answers in the sense of Def. 36 if no additional conditions are required in its condition (2.). The following proposition relates Def. 37 to the common view of extensional answers as sets of tuples representing variable bindings under which the query is consistent with or entailed by the knowledge base.

Proposition 38 (Characteristics of Extensional Answers). *Let B be a sentence and let $Q(\boldsymbol{x})$ be a formula. For all formulas $A(\boldsymbol{x}) \Subset \emptyset$ it holds that*

(i) *$A(\boldsymbol{x})$ is the consistency-based extensional answer to Q with respect to B if and only if*

$$\{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models A(\mathbf{c})\} \;=\; \{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models \mathsf{sat}(B \wedge Q(\mathbf{c}))\}.$$

(ii) *$A(\boldsymbol{x})$ is the entailment-based extensional answer to Q with respect to B if and only if*

$$\{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models A(\mathbf{c})\} \;=\; \{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models \mathsf{valid}(B \rightarrow Q(\mathbf{c}))\}.$$

Proof (Proposition 38). The following equivalences relate the formulas in the proposition to the GSNC and GWSC, where the scope is empty:

$$\mathsf{sat}(B \wedge Q(\boldsymbol{x})) \equiv \mathsf{project}_{\emptyset}(B \wedge Q(\boldsymbol{x})) \equiv \mathsf{gsnc}_{\emptyset}(B, Q(\boldsymbol{x})).$$
$$\mathsf{valid}(B \rightarrow Q(\boldsymbol{x})) \equiv \neg\mathsf{project}_{\emptyset}(B \wedge \neg Q(\boldsymbol{x})) \equiv \mathsf{gwsc}_{\emptyset}(B, Q(\boldsymbol{x})).$$

The proposition then follows since all formulas $F(\boldsymbol{x}), G(\boldsymbol{x})$ such that $F(\boldsymbol{x}) \Subset \emptyset$ and $G(\boldsymbol{x}) \Subset \emptyset$ it holds that

$$\{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models F(\mathbf{c})\} = \{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models G(\mathbf{c})\}$$
$$\text{iff } \; F(\boldsymbol{x}) \equiv G(\boldsymbol{x}).$$

□

Notice that $\models \mathsf{valid}(B \to Q(\mathbf{c}))$ is equivalent to $B \models Q(\mathbf{c})$, allowing to express the equality of sets of tuples in Prop. 38.ii also as

$$\{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models A(\mathbf{c})\} \;=\; \{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } B \models Q(\mathbf{c})\}. \quad \text{(E9)}$$

As stated in the following proposition, in case of unique definability of the query formula, consistency-based and entailment-based extensional answers coincide. Notice that by Prop. 27, *unique* definability in terms of $\emptyset$ within B follows for satisfiable sentences B already from definability.

Proposition 39 (Uniquely Definable Extensional Answers). *Let B, Q be formulas such that Q is uniquely definable in terms of $\emptyset$ within B. The consistency-based and entailment-based extensional answers to Q with respect to B are the same (up to equivalence).*

Proof (Proposition 39). Immediate from Def. 37 and Def. 23 □

17 Answers to Relational Database Queries

We do not adhere here to the often used identification of a relational database with an interpretation, but consider a relational database as a *formula*. This allows smooth passing from relational databases to knowledge bases expressed in richer languages. Those properties of the data- or knowledge bases on which a proven property actually depends can be more clearly exhibited. With second-order operators, even properties that are not first-order expressible, such as finiteness, could be expressed at the formula level, typically with the idea that the second-order operators will be eliminated when answers are computed. A further rationale for the representation of databases as formulas is that this matches with *constraint query languages* [KKR95], an approach that overcomes issues of safety by generalizing databases with finite relations to databases that are *finite representations* of relations, in particular, finite disjunctions of finite conjunctions of constraints.

An answer to a relational database query can be represented as instance of an answer in the sense Def. 36, where the scope S is $\emptyset$ and condition (2.) is set to the following syntactic properties:

$$\begin{array}{ll} \text{(X1)} & A \text{ contains no predicate symbols.} \\ \text{(X2)} & A \text{ does not contain second-order operators.} \\ \text{(X3)} & A \text{ does not contain first-order quantifiers.} \end{array} \quad \text{(E10)}$$

Answers meeting (X1)–(X3) are quantifier-free first-order formulas with syntactic equality statements, but without predicates, such as, for example, $(x \doteq \mathsf{a} \wedge y \doteq \mathsf{b}) \vee (x \doteq \mathsf{a} \wedge y \doteq \mathsf{c}) \vee (x \doteq \mathsf{b} \wedge y \not\doteq \mathsf{a})$, corresponding to the set of tuples $\{\langle \mathsf{a}, \mathsf{b}\rangle, \langle \mathsf{a}, \mathsf{c}\rangle\} \cup \{\langle \mathsf{b}, y\rangle \mid y \in \mathsf{CONST} \setminus \{\mathsf{a}\}\}$. If Q is definable in terms of $\emptyset$ within B, then a formula A' meets the semantic requirements on answers, that is condition (1.) of Def. 18 if and only if $A' \Subset \emptyset$ and $\mathsf{gsnc}_\emptyset(B, Q) \models A' \models \mathsf{gwsc}_\emptyset(B, Q)$. This includes the two extremes $A' \equiv \mathsf{gsnc}_\emptyset(B, Q)$ and $A' \equiv \mathsf{gwsc}_\emptyset(B, Q)$, that is,

the consistency- and the entailment-based extensional answer. In case of unique definability, both extremes would be equivalent. Given such a formula A', an equivalent formula A that meets also the syntactic conditions (X1)–(X2) can be obtained by *eliminating* in A' the second-order operators and first-order quantifiers. We flesh out this view on conventional answers to relational database queries by indicating steps in which an answer can be determined for sentences B of a particular form that can be considered as a relational database:

$$\bigwedge_{i=1}^{n} \forall \boldsymbol{x}_i \, (d_i(\boldsymbol{x}_i) \leftrightarrow G_i(\boldsymbol{x}_i)),$$

where $n \geq 0$, each d_i is a predicate, $\boldsymbol{x} = x_1, \ldots, x_{\mathsf{arity}(d_i)}$, and each $G_i(\boldsymbol{x}_i)$ is a quantifier-free first-order formula without predicates. (E11)

Clearly, such a sentence B provides for each "database predicate" d_i a definition $G_i \Subset \emptyset$ that specifies the extension of d_i by equality constraints. Let D be the predicate scope $\{d_1, \ldots, d_n\}$. Then B defines scope D in terms of $\emptyset$. Moreover, since B is a satisfiable sentence, by Prop. 27 it follows that B defines scope D *uniquely* in terms of $\emptyset$. Any query formula $Q \Subset D$ is thus uniquely definable in terms of $\emptyset$ within B and the semantics of an answer A to Q with respect to $\emptyset$, corresponding to condition (1.) of Def. 36, is given by $A \equiv \mathsf{gsnc}_\emptyset(B, Q) \equiv \mathsf{gwsc}_\emptyset(B, Q)$. If Q is a first-order formula in which only predicates from D do occur, then a formula A that also meets the syntactic conditions (X1)–(X2) can be obtained by eliminating the GSNC or GWSC operator, respectively, in $\mathsf{gsnc}_\emptyset(B, Q)$ or $\mathsf{gwsc}_\emptyset(B, Q)$. This can actually be performed simply by starting with Q and replacing all predicates with their definitions according to B. Finally, (X3) can be ensured by performing first-order quantifier elimination on the result obtained in the previous step.

18 The Datalog Perspective on Extensional Answers

Datalog formulas are first-order formulas that meet certain syntactic restrictions and can be considered under a special semantics, allowing to understand them as recursive specifications of named database relations. While there are many variant classes of datalog formulas investigated in the literature (e.g. [Ull89]), they usually have in common that their members must be universal first-order formulas without functions except for constants. We specify the semantics associated with datalog formulas here in terms of predicate circumscription, expressed by the second-order operator circ for *scope-determined circumscription* [Wer12], which has the same argument types as $\mathsf{project}$ and is defined semantically, analogously to projection:

Definition 40 (Scope-Determined Circumscription). The *scope-determined circumscription* of formula F onto scope S, in symbols $\mathsf{circ}_S(F)$, is a formula whose semantics is defined as follows: For all interpretations $\langle I, \beta \rangle$ it holds that

$$\langle I, \beta \rangle \models \mathsf{circ}_S(F) \quad \text{iff}_{\text{def}} \quad \begin{array}{l} \langle I, \beta \rangle \models F \text{ and} \\ \text{there does not exists a structure } J \text{ such that} \\ \langle J, \beta \rangle \models F \text{ and } J \cap S \subset I \cap S. \end{array}$$

The circ operator allows to express variants of parallel predicate circumscription where the effects on each atom are controlled by a scope argument. Atoms that occur just in a *positive* literal in the scope are minimized, atoms that occur just in a *negative* literal are maximized, atoms that occur in *both polarities* are fixed and atoms that do *not at all* occur in the scope are varying. Thus, if F is a formula whose atoms are in disjoint sets P, Q and Z, then the *parallel predicate circumscription of P in F with fixed Q and varied Z*, traditionally written as $\mathrm{CIRC}[F; P; Z]$, would be expressed as $\mathsf{circ}_{(P \cap \mathsf{POS}) \cup Q}(F)$.

The following proposition shows a property of scope-determined circumscription that underlies the correspondence of the circumscription semantics and classical semantics for datalog, shown in Prop. 43 below: A universal first-order formula "in the circumscription scope" is entailed by a formula if and only if it is entailed by its circumscription. Proofs and further variants of this property are shown in [Wer12].

Proposition 41 (Consequences of Scope-Determined Circumscription). *Let F be a universal first-order formula, let S be a scope and let $G \Subset S$ be a further formula. It then holds that $\mathsf{circ}_S(F) \models G$ if and only if $F \models G$.*

We consider here exemplarily the variant of datalog where a knowledge base is represented by a conjunction of range restricted Horn clauses, defined as follows:

Definition 42 (Range Restricted Horn Clause). A *range restricted Horn clause* is a sentence of the form

$$\forall \boldsymbol{x}\, A_0 \leftarrow A_1 \wedge \ldots \wedge A_k \tag{E12}$$

where $k \geq 0$, the A_i are atoms, and all variables in A_0 occur in $(A_1 \wedge \ldots \wedge A_k)$.

We assume a finite Herbrand domain. If B is a conjunction of range restricted Horn clauses , then B has a unique minimal model in which all predicates have a finite extent. It is the unique model of $\mathsf{circ}_{\mathsf{POS}}(B)$, which is equivalent to a sentence of the form (E11). This implies implying unique definability of all involved predicates in terms of $\emptyset$ within $\mathsf{circ}_{\mathsf{POS}}(B)$. The following proposition gives further insight into the special role of circumscription for datalog and shows a possibility to incorporate goal related projection into datalog knowledge bases.

Proposition 43 (Coincidence Properties for Datalog). *Let B be a conjunction of range restricted Horn clauses and let $q(\boldsymbol{x})$ be an atom with predicate q. Then the following formulas are equivalent:*

1. *The consistency-based extensional answer to $q(\boldsymbol{x})$ with respect to $\mathsf{circ}_{\mathsf{POS}}(B)$.*
2. *The entailment-based extensional answer to $q(\boldsymbol{x})$ with respect to $\mathsf{circ}_{\mathsf{POS}}(B)$.*
3. *The entailment-based extensional answer to $q(\boldsymbol{x})$ with respect to B.*
4. *The entailment-based extensional answer to $q(\boldsymbol{x})$ with respect to $\mathsf{project}_{\{q\}}(B)$.*

Proof (Proposition 43). By Prop. 39, the unique definability of $q(\boldsymbol{x})$ in terms of $\emptyset$ within $\mathsf{circ}_{\mathsf{POS}}(B)$ implies equivalence of (2.) to (1.). By Prop. 38.ii, a formula $A(x) \Subset \emptyset$ is equivalent to (2.) if and only if

$$\begin{aligned} &\{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \models A(\mathbf{c})\} \\ = &\{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } \mathsf{circ}_{\mathsf{POS}}(B) \models q(\mathbf{c})\}. \end{aligned} \tag{E13}$$

Since $q(\boldsymbol{x}) \Subset \mathsf{POS}$, by Prop. 41 it follows that $\mathsf{circ}_{\mathsf{POS}}(B) \models q(\boldsymbol{x})$ if and only if $B \models q(\boldsymbol{x})$. Thus, equality (E13) does not alter its meaning if we replace $\mathsf{circ}_{\mathsf{POS}}(B) \models q(\boldsymbol{x})$ by $B \models q(\boldsymbol{x})$. By applying Prop. 38.ii again, we can conclude equivalence of (3.) to (2.). Equivalence of (4.) to (3.) follows in a similar way from Prop. 4.vi, which justifies that $B \models q(\boldsymbol{x})$ if and only if $\mathsf{project}_{\{q\}}(B) \models q(\boldsymbol{x})$. □

The equivalence of (1.) and (2.) to (3.) in Prop. 43 shows that, while circumscription is essential to capture the meaning of datalog formulas as providing actual definitions of the involved predicates, circumscription is not necessary to evaluate queries, that is, to determine the extent of predicates. Equivalence to (4.) shows that the extent of a predicate can be determined just from the projection of the datalog formula onto the respective predicate.

19 Answers with Allowed Predicates

By Def. 36, an answer A is a special case of a definition of query formula Q in terms of answer scope S within knowledge base B. For extensional answers, the answer scope is empty. If S is not empty, that is, if certain predicates are allowed to occur in answers, they correspond to certain forms of "intensional" answers or "view rewritings". For B of the form (E11), the GSNC and GWSC operators in $\mathsf{gsnc}_S(B, Q)$ or $\mathsf{gwsc}_S(B, Q)$, respectively, can then be eliminated by expanding just predicates that are not in S with their definitions according to B. We generalize (E11) by specifying a class of formulas where predicates can be defined as "views", that is, predicates and first-order quantifiers are permitted on the right side of definitions:

$$\bigwedge_{i=1}^{n} \forall \boldsymbol{x}_i \, (d_i(\boldsymbol{x}_i) \leftrightarrow G_i(\boldsymbol{x}_i)),$$

where $n \geq 0$, each d_i is a predicate, $\boldsymbol{x} = x_1, \ldots, x_{\mathsf{arity}(d_i)}$, and each $G_i(\boldsymbol{x}_i)$ is a first-order formula with predicates just from $d_1, \ldots, d_n$. (E14)

To obtain the form (E14), void definitions $\forall \boldsymbol{x}_i \; d_i(\boldsymbol{x}_i) \leftrightarrow d_i(\boldsymbol{x}_i)$ can be supplied for all predicates d_i that have no other definition. Since definitions in a formula B of form (E14) might be recursive, simple definition expansion does no longer suffice in general as elimination technique to compute $\mathsf{gsnc}_S(B, Q)$ or $\mathsf{gwsc}_S(B, Q)$.

If Q is definable in terms of S within B, then both $\mathsf{gsnc}_S(B, Q)$ and $\mathsf{gwsc}_S(B, Q)$ provide answers. The following is an argument in favor or $\mathsf{gsnc}_S(B, Q)$: If E is a formula that is not implied by B, but relevant for determining the answer, for example the a schema mapping that provides definitions of certain predicates occurring in the query in terms of concepts used in B, then E can be added equivalently to the knowledge base B or the query Q. That is,

$$\mathsf{gsnc}_S(B \wedge E, Q) \;\equiv\; \mathsf{project}_S(B \wedge E \wedge Q) \;\equiv\; \mathsf{gsnc}_S(B, Q \wedge E). \qquad \text{(E15)}$$

A further subtlety concerning answers with allowed predicates is that, since we have specified syntactic equality $\doteq$ as a "built-in" logic operator in Def. 1 and

not as a predicate, "extensional" answers, that is, answers that express results in terms of $\doteq$ and logic operators, are considered as special cases of answers which are restricted to contain only specific allowed predicates. The following example show the interplay of an "extensionally" and an "intensionally defined" predicate for answers with allowed predicates.

Example 44 (Interplay with Extensional Definitions in Answer with Allowed Predicates). Let $B = (\forall x\ \mathsf{q}(x) \leftrightarrow \mathsf{p}(x)) \wedge (\forall x\ \mathsf{p}(x) \leftrightarrow x = \mathsf{a})$, let $Q(x) = \mathsf{q}(x)$ and let $S = \{\mathsf{p}\}$. It then holds that

$$\mathsf{gsnc}_S(B, Q(x)) \equiv (\forall x\, p(x) \leftrightarrow x \doteq a) \wedge x \doteq a \equiv (\forall x\ p(x) \leftrightarrow x \doteq a) \wedge p(x), \text{ and}$$
$$\mathsf{gwsc}_S(B, Q(x)) \equiv (\forall x\, p(x) \leftrightarrow x \doteq a) \rightarrow x \doteq a \equiv (\forall x\ p(x) \leftrightarrow x \doteq a) \rightarrow p(x).$$

Since $Q(x)$ is definable in terms of S within B (i.e., $\mathsf{gsnc}_S(B, Q) \models \mathsf{gwsc}_S(B, Q)$), all the shown equivalents to $\mathsf{gsnc}_S(B, Q)$ and $\mathsf{gwsc}_S(B, Q)$, respectively, provide answers to $Q(\boldsymbol{x})$ in terms of S with respect to B.

Chapter 5

View-Based Query Processing

20 Overview on View-Based Query Processing

In *view-based query processing* [Hal01,CGLV00,CGLV07,Mar07,NSV10], the scenario basically includes two knowledge bases: First, the *relation definitions* or *database* B of the form (E11), such that predicates $D = \{d_1, \ldots, d_n\}$ are uniquely definable in terms of $\emptyset$ within B. Second, the *view definitions* V, that is, a sentence of the form

$$\bigwedge_{i=1}^{m} \forall \boldsymbol{x}_i \, (u_i(\boldsymbol{x}_i) \leftrightarrow H_i(\boldsymbol{x}_i)),$$

where $m \geq 0$, each u_i is a predicate, $\boldsymbol{x} = x_1, \ldots, x_{\mathsf{arity}(u_i)}$, and each $H_i(\boldsymbol{x}_i)$ is a first-order formula with predicates just from D. (E16)

The predicates $U = \{u_1, \ldots, u_m\}$ are then definable in terms of D within V. The general objective of view-based query answering is to answer a query Q over the "database predicates" D by using only answers to the views specified in V. In the context of query optimization and database design, the idea is that answers to these views can be computed in a particular efficient way since they are precomputed or specially optimized. Notice that view-based query processing in this sense corresponds to a proceeding in the direction from definiens to definiendum: A view definition V provides definitions of formulas over view predicates U in terms of the database predicates D, whereas view-based query processing involves to determine for a given query Q over database predicates D an alternate definition that is over the view predicates U.

Data integration is a further important application area of view-based query processing, where the idea is that views are used to describe data sources in terms of the mediated schema, that is, predicates D of a virtual global database. The view definitions are then used to translate a user query over the mediated schema into queries over the data sources, that is, view predicates U.

Query rewriting is a two-step approach to view-based query processing: First, for a given query Q over D and given view definitions V a *rewritten query* R over U is computed, with the characteristics that for all databases B the answer to Q with respect to B is identical to the answer to R with respect to $(B \wedge V)$. As second step, the rewritten query R is evaluated against the view extensions, that is, the answer to R in terms of $\emptyset$ with respect to knowledge base $(B \wedge V)$, for a particular given B, is determined. As discussed in [CGLV00,CGLV07], this approach can be distinguished from *query answering*, where the two phases are not strictly separated and the rewritten query R needs not necessarily to be explicitly constructed.

The concepts involved in view-based query processing have been made precise and investigated for regular-path queries in [CGLV07]. In this chapter, we approach these concepts by semantic characterizations in terms of GSNC/GWSC, *definition* and *definability*, applying to formulas in general as queries, and show their relation to the characteristic properties specified in [CGLV07]. As properties that characterize formula classes of queries and rewritings, we consider here just restrictions of the scope, that is conditions of the form $F \Subset S$ for some scope S. This suffices to express in a semantic way restrictions of the allowed predicates and the polarity in which they do occur. However, it is a limitation compared to [CGLV07], where syntactically constrained classes, such as conjunctive queries, regular-path queries, and conjunctive regular-path queries, are taken into account. Since the essential difference of query answering to query rewriting can be attributed to the consideration of the existence of rewritten queries R in particular such classes, this is not modeled within our semantic framework.

On the other hand, for many of the concepts in view-based query processing we use generalizations that correspond to knowledge processing in first-order logic. In particular, the roles of "database" and "view extension" can be played by arbitrary formulas instead of interpretations, such that for example also disjunctive databases are covered. All involved concepts are characterized just by semantic properties. For instance, a view specification is just a formula that is semantically required to provide definability for all literals in the view scope and to meet a further semantic conservativeness condition.

In this chapter, we basically consider view-based query processing with respect to *entailment-based* extensional answers, that is, we consider as semantics of the answer to query Q with respect to database B the formula $\mathsf{gwsc}_{\emptyset}(B, Q)$, which, if $Q = Q(\boldsymbol{x})$, by Prop. 38.ii and equality (E9), can be regarded as representation of the set of tuples $\{\mathbf{c} \mid \mathbf{c} \in \mathsf{CONST}^n \text{ and } B \models Q(\mathbf{c})\}$.

21 View Definition

The concept of *view definition*, defined as follows, gathers the formula that actually provides definitions with the scopes of definienda and definientia, and ensures definability and conservativeness constraints between them.

Definition 45 (View Definition). Let D, U be scopes and let V be a formula. The triple $\mathcal{V} = \langle V, U, D\rangle$ is called a *view definition of U in terms of D with specification V* if and only if

1. For all formulas $R \Subset U$ it holds that R is definable in terms of D within V.
2. For all formulas $B \Subset D$ it holds that V is conservative for D within B.

Before we discuss the intuition behind that definition in more detail, we note that if its condition (1.) is strengthened by requiring *unique* definability in place of just definability, this leads to an equivalent characterization of *view definition*, as stated in the following proposition, justified by Prop. 35, which allows to conclude unique definability from definability and conservativeness.

Proposition 46 (View Definitions Provide Unique Definability). *Let D, U be scopes and let V be a formula. The triple $\mathcal{V} = \langle V, U, D\rangle$ is a view definition of U in terms of D with specification V if and only if*

1. *For all formulas $R \Subset U$ it holds that R is uniquely definable in terms of D within V.*
2. *For all formulas $B \Subset D$ it holds that V is conservative for D within B.*

Proof (Proposition 46). The right-to-left direction, that is, concluding from *unique* definability just definability, is trivial. The left-to-right direction follows from Prop. 35. □

If V is sentence of the form (E16), then $\mathcal{V} = \langle V, U, D\rangle$, with $U = \{u_1, \ldots, u_m\}$ is a prototypical example of a view definition. The intuition behind Def. 45 is that a view specification is a formula V that provides definitions of all literals in the "view vocabulary" U in terms of the "database vocabulary" D (condition (1.)) and that, if combined with an arbitrary database B, then V does not mess up with the knowledge about the "database vocabulary" that is provided by B (condition (2.)). The semantic characterization of view definition in Def. 45 does not restrict the scopes U and D any further. They may, for example, overlap.

By Proposition 34.i, the condition (2.) in Def. 45 is equivalent to: For all formulas $B \Subset D$ it holds that $\mathsf{gsnc}_D(V, B) \equiv B$. The latter equivalence can also be expressed directly in terms of projection as $\mathsf{project}_D(V \wedge B) \equiv B$. In case the "database vocabulary" D is an atom scope, by Proposition 34.ii the condition (2.) is equivalent to the weaker condition $\mathsf{project}_D(V) \equiv \top$, as stated in the following Proposition 47.

Proposition 47 (View Definition for Atom Scopes). *Let D be an atom scope, let U a scope and let V be a formula. The triple $\mathcal{V} = \langle V, U, D\rangle$ is a view definition of U in terms of D with specification V if and only if*

1. *For all formulas $R \Subset U$ it holds that R is definable in terms of D within V.*
2. *$\mathsf{project}_D(V) \equiv \top$.*

Proof (Proposition 47). Left-to-right: Assume the left side of the proposition, that is, for all formulas $B \Subset D$ it holds that V is conservative for D within B. Since $\top \Subset D$ it then holds that V is conservative for D within $\top$; thus, by Prop. 34.ii, $\top \models \mathsf{project}_D(V)$; thus $\mathsf{project}_D(V) \equiv \top$. Right-to-left: Assume the right side of the proposition, that is, $\mathsf{project}_D(V) \equiv \top$ and let B be a formula such that $B \Subset D$. Then $\top \models \mathsf{project}_D(V)$ and thus $B \models \mathsf{project}_D(V)$. Thus, by Prop. 34.ii, it holds that V is conservative for D within B. □

As the following counterexample shows, the characterization of view definitions in Prop. 47 indeed fails of scopes D that are not atom scopes. The example gives a formula V, a scope U and a scope D that is not an atom scope such that condition (2.) of Prop. 47 is met, but the stronger condition (2.) of Def. 45 fails, establishing that $\mathcal{V} = \langle V, U, D\rangle$ is not a view definition:

Example 48 (Variants of Condition (2.) of View Definition). Let $V = (\neg\mathsf{p} \wedge \neg\mathsf{q})$, $U = \{+\mathsf{p}, -\mathsf{p}\}$ and $D = \{+\mathsf{q}\}$. It then holds that all formulas in scope U are definable in terms of D within V: That p is definable follows from Prop. 19, since $\mathsf{gwsc}_{+\mathsf{q}}(\neg\mathsf{p} \wedge \neg\mathsf{q}, \mathsf{p}) \equiv \mathsf{project}_{+\mathsf{q}}(\neg\mathsf{p} \wedge \neg\mathsf{q} \wedge \mathsf{p}) \equiv \bot$. That $\neg\mathsf{p}$ is definable follows from that proposition, since $\neg\mathsf{gsnc}_{+\mathsf{q}}(\neg\mathsf{p} \wedge \neg\mathsf{q}, \neg\mathsf{p}) \equiv \neg\mathsf{project}_{-\mathsf{q}}(\neg\mathsf{p} \wedge \neg\mathsf{q} \wedge \mathsf{p}) \equiv \top$. Condition (2.) of Prop. 47 holds: $\mathsf{project}_{+\mathsf{q}}(\neg\mathsf{p} \wedge \neg\mathsf{q}) \equiv \top$. Condition (2.) of Def. 45 does not hold. We show this by giving a formula $B \Subset +\mathsf{q}$ such that $\mathsf{project}_D(V \wedge B) \not\equiv B$, which by Prop. 34.i implies failure of the *conservative* property required in the condition. Let $B = \mathsf{q}$. Then $\mathsf{project}_D(V \wedge B) = \mathsf{project}_{\neg\mathsf{q}}(\neg\mathsf{p} \wedge \neg\mathsf{q} \wedge \mathsf{q}) \equiv \bot \not\equiv \mathsf{q}$.

22 View Extension

What we call here *exact view extensions* corresponds to the symbolically written $\mathcal{V}^{\Sigma}(\mathcal{B})$ in [CGLV07]. Intuitively, the *exact view extension* of a view definition $\mathcal{V} = \langle V, U, D\rangle$ for a database $B \Subset D$ is another database $E \Subset U$ that defines the view scope U in terms of $\emptyset$ and is obtained from the view specification V by applying the definitions of the members of the database scope D according to B. For all queries $Q \Subset D$ and $R \Subset U$ such that Q is a definition of R in terms of D within V it must hold that the answer to R with respect to E is the same as the answer to Q with respect to B. Definition 49 below specifies a semantic characterization of such formulas E. It is followed by Prop. 50, which states the characteristic properties of an exact view extension, and certifies that Def. 49 specifies the unique formula with these properties. The condition (2.) of Def. 45, conservativeness, is applied in the proof of Prop. 50 to infer that $\mathsf{gsnc}_D(V, B) \equiv B$. Examples for exact view extensions are then provided with Examp. 51 and 52.

Definition 49 (Exact View Extension). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $B \Subset D$ be a formula. Then the *exact view extension of $\mathcal{V}$ for B* is the formula

$$\mathsf{gsnc}_U(V, B).$$

Proposition 50 (Characteristics of Exact View Extension). *Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $B \Subset D$ be a formula. Then, for all formulas E the following statements are equivalent:*

1. *E is the exact view extension of $\mathcal{V}$ for B.*
2. *$E \Subset U$, and for all formulas $Q \Subset D$ and $R \Subset U$ such that Q is a definition of R in terms of D within V the entailment-based extensional answer to R with respect to E and the entailment-based extensional answer to Q with respect to B are equivalent.*

Proof (Proposition 50). We first note that by Def. 49 statement (1.) is equivalent to

$$E \equiv \mathsf{gsnc}_U(V, B),$$

and, by Def. 37.ii, the equivalence in statement (2.) is equivalent to

$$\mathsf{gwsc}_{\emptyset}(E, R) \equiv \mathsf{gwsc}_{\emptyset}(B, Q).$$

Assume the preconditions of the proposition:

(1)	$\mathcal{V} = \langle V, U, D\rangle$ is a view definition.	assumption
(2)	$B \Subset D$.	assumption

Left-to-right. We show that statement (2.) is true under the stated preconditions if E is replaced by $\mathsf{gsnc}_U(V, B)$. By Prop. 13.ix it holds that $\mathsf{gsnc}_U(V, B) \Subset U$, the first conjunct of statement (2.). We now show the second conjunct Consider the following table. Assume as further preconditions steps (3)–(5). Steps (6)–(8) follow.

(3)	$Q \Subset D$.	assumption
(4)	$R \Subset U$.	assumption
(5)	Q is a definition of R in terms of D within V.	assumption
(6)	$\dot{\forall} B \Subset D : V$ is conservative for D within B.	by (1), cond. (2.) of Def. 45
(7)	$\mathsf{project}_D(V, B) \equiv B$.	by (6), (2), Prop. 34.i, exp. gwsc
(8)	$V \wedge \neg R \equiv V \wedge \neg Q$.	by (5)

Now $\mathsf{gwsc}_{\emptyset}(\mathsf{gsnc}_U(V, B), R) \equiv \mathsf{gwsc}_{\emptyset}(B, Q)$ can be derived as follows:

(9)	$\mathsf{gwsc}_{\emptyset}(\mathsf{gsnc}_U(V, B), R)$	
(10)	$\equiv \neg\mathsf{project}_{\emptyset}(\mathsf{project}_U(V \wedge B) \wedge \neg R)$	by exp. gwsc, gsnc
(11)	$\equiv \neg\mathsf{project}_{\emptyset}(\mathsf{project}_U(\mathsf{project}_U(V \wedge B) \wedge \neg R))$	by Prop. 4.v
(12)	$\equiv \neg\mathsf{project}_{\emptyset}(\mathsf{project}_U(V \wedge B \wedge \neg R))$	by (4), Prop. 4.xvii, 9.i
(13)	$\equiv \neg\mathsf{project}_{\emptyset} V \wedge B \wedge \neg R$	by Prop. 4.v
(14)	$\equiv \neg\mathsf{project}_{\emptyset}(V \wedge B \wedge \neg Q)$	by (8)
(15)	$\equiv \neg\mathsf{project}_{\emptyset}(\mathsf{project}_D(V \wedge B) \wedge \neg Q)$	by (3), Prop. 4.v, Prop. 4.xvii, 9.i
(16)	$\equiv \neg\mathsf{project}_{\emptyset}(B \wedge \neg Q)$	by (7)
(17)	$\equiv \mathsf{gwsc}_{\emptyset}(B, Q)$.	by con. gwsc

Right-to-left. We have to show that, under the preconditions (1) and (2), whenever E satisfies statement (2.), then $E \equiv \mathsf{gsnc}_U(V, B)$. Let $\mathsf{DEF}(V, D, R, Q)$ be a shorthand for *Q is a definition of R in terms of D within V*. Consider the following table. Let E be a formula that satisfies (2.), that is, assume steps (18) and (19). To derive step (20), we assume the left-to-right direction of this proposition, which we have just shown. With step (26) we derive statement (1.).

(18) $E \Subset U$. assumption
(19) $\dot\forall R \Subset U$:
$\dot\forall Q \Subset D : \mathsf{DEF}(V, D, R, Q) \Rightarrow$
$\mathsf{gwsc}_\emptyset(E, R) \equiv \mathsf{gwsc}_\emptyset(B, Q)$. assumption
(20) $\dot\forall R \Subset U$:
$\dot\forall Q \Subset D : \mathsf{DEF}(V, D, R, Q) \Rightarrow$
$\mathsf{gwsc}_\emptyset(E, R) \equiv \mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V, B), R)$. by (19), l-to-r dir. of this prop.
(21) $\dot\forall R \Subset U$:
$(\dot\exists Q \Subset D \mathbin{\dot\wedge} \mathsf{DEF}(V, D, R, Q)) \Rightarrow$
$\mathsf{gwsc}_\emptyset(E, R) \equiv \mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V, B), R)$. equiv. to (20)
(22) $\dot\forall R \Subset U$:
$\dot\exists Q \Subset D : \mathsf{DEF}(V, D, R, Q)$. by (1), cond. (1.) of Def. 45
(23) $\dot\forall R \Subset U$:
$\mathsf{gwsc}_\emptyset(E, R) \equiv \mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V, B), R)$. by (22), (21)
(24) $\dot\forall R \Subset U$:
$(E \models R \Leftrightarrow \mathsf{gsnc}_U(V, B) \models R)$. by (23), Prop. 14
(25) $\mathsf{project}_U(E) \equiv \mathsf{project}_U(\mathsf{gsnc}_U(V, B))$. by (24), Prop. 11.ii
(26) $E \equiv \mathsf{gsnc}_U(V, B)$. by (25), (18), Prop. 13.ix

□

Example 51 (Exact View Extension). Let $\mathsf{CONST} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $D = \{\mathsf{r}/1, \mathsf{s}/1\}$, $U = \{\mathsf{p}/1, \mathsf{q}/1\}$, and let V, B be as follows:

$$V = (\forall x\, \mathsf{p}(x) \leftrightarrow \mathsf{r}(x)) \wedge (\forall x\, \mathsf{q}(x) \leftrightarrow \mathsf{r}(x) \wedge \mathsf{s}(x)),$$
$$B = (\forall x\, \mathsf{r}(x) \leftrightarrow x \doteq \mathsf{a} \vee x \doteq \mathsf{b}) \wedge (\forall x\, \mathsf{s}(x) \leftrightarrow x \doteq \mathsf{b} \vee x \doteq \mathsf{c}).$$

Then the exact view extension E of $\mathcal{V} = \langle V, U, D\rangle$ for B is $E = \mathsf{gsnc}_U(V, B) \equiv$

$$\mathsf{project}_U(V \wedge B) \equiv ((\forall x\, \mathsf{p}(x) \leftrightarrow x \doteq \mathsf{a} \vee x \doteq \mathsf{b}) \wedge (\forall x\, \mathsf{q}(x) \leftrightarrow x \doteq \mathsf{b})).$$

As a query in U consider for example $R = \mathsf{q}(x)$. Then $Q = (\mathsf{r}(x) \wedge \mathsf{s}(x))$ is a definition of R in terms of D within V. The answer to R with respect to E is the same as the answer to B with respect to Q: $\mathsf{gwsc}_\emptyset(E, R) \equiv \mathsf{gwsc}_\emptyset(B, Q) \equiv x \doteq \mathsf{b}$.

The characteristic property of exact view extensions (Prop. 50) applies to "database formulas" B that are in the "database scope" D. In Examp. 51 we considered a formula B where all predicates in D are defined in terms of $\emptyset$, for example, the predicate $\mathsf{r}/1$ as $\mathsf{r}(x) \leftrightarrow x \doteq a$. This is, however, no requirement for the concept of *exact view extension*, as the following example demonstrates, where we consider a formula B in which the predicates in D are not fully defined.

Example 52 (Exact View Extension with an "Incomplete" Database). Let $\mathsf{CONST} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, let $D = \{\mathsf{r}/1\}$, let $U = \{\mathsf{p}/1\}$, let $V = (\forall x\, \mathsf{p}(x) \leftrightarrow \mathsf{r}(x))$ and let $B = \neg\mathsf{r}(\mathsf{a})$. Then the exact view extension E of $\mathcal{V} = \langle V, U, D\rangle$ for B is $E = \mathsf{gsnc}_U(V, B) \equiv \mathsf{project}_U(V \wedge B) \equiv$

$$\mathsf{project}_U((\forall x\, \mathsf{p}(x) \leftrightarrow \mathsf{r}(x)) \wedge \neg\mathsf{r}(\mathsf{a})) \equiv \neg\mathsf{p}(\mathsf{a}).$$

For $\neg\mathsf{p}(x)$ as query in the view scope U, we then obtain as answer $\mathsf{gwsc}_\emptyset(E, \neg\mathsf{p}(x)) \equiv \mathsf{gsnc}_\emptyset(B, \neg\mathsf{r}(x)) \equiv x \doteq \mathsf{a}$. For $\mathsf{p}(x)$ as query in the view scope U, we obtain as answer $\mathsf{gwsc}_\emptyset(E, \mathsf{p}(x)) \equiv \mathsf{gsnc}_\emptyset(B, \mathsf{r}(x)) \equiv \top$.

A *sound view extension* E of a view definition $\mathcal{V} = \langle V, U, D\rangle$ for a database $B \Subset D$ generalizes *exact view extension* by requiring just that for all queries $Q \Subset D$ and $R \Subset U$ such that Q is a definition of R in terms of D within V it must hold that the answer to R with respect to E *entails* the answer to Q respect to B. If answers are considered as sets of tuples, this means that the set of answer tuples to R are a subset of the set of answer tuples to Q. In this sense the answer tuples to R are considered as "sound". Definition 53 below specifies a semantic characterization of sound view extensions. The subsequent Prop. 54 shows its characteristic properties.

Definition 53 (Sound View Extension). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $B \Subset D$ be a formula. Then a formula E is called a *sound view extension of* $\mathcal{V}$ *for* B if and only if

1. $E \Subset U$.
2. $\mathsf{gsnc}_U(V, B) \models E$.

Proposition 54 (Characteristics of Sound View Extension). *Let* $\mathcal{V} = \langle V, U, D\rangle$ *be a view definition and let* $B \Subset D$ *be a formula. Then for all formulas* E *the following statements are equivalent:*

1. *E is a sound view extension of $\mathcal{V}$ for B.*
2. *$E \Subset U$, and for all formulas $Q \Subset D$ and $R \Subset U$ such that Q is a definition of R in terms of D within V it holds that the entailment-based extensional answer to R with respect to E entails the entailment-based extensional answer to Q with respect to B.*

Proof (Proposition 54). We first note that by Def. 53 statement (1.) is equivalent to

$$E \Subset U \text{ and } \mathsf{gsnc}_U(V, B) \models E,$$

and, by Def. 37.ii, the entailment in statement (2.) is equivalent to

$$\mathsf{gwsc}_\emptyset(E, R) \models \mathsf{gwsc}_\emptyset(B, Q).$$

Left-to-right. Let E be a formula that satisfies statement (1.). The first conjunct of statement (2.), that is, $E \Subset U$, then follows immediately. Let R, Q be formulas that satisfy the precondition of the second conjunct of statement (2.). From Prop. 49 it can be concluded that $\mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V, B), R) \models \mathsf{gwsc}_\emptyset(B, Q)$. From our assumption $\mathsf{gsnc}_U(V, B) \models E$ if then follows by Prop. 13.xii that $\mathsf{gwsc}_\emptyset(E, R) \models \mathsf{gwsc}_\emptyset(B, Q)$.

Right-to-left. We proceed similarly than in the proof of Prop. 50. Let $\mathsf{DEF}(V, D, R, Q)$ be a shorthand for *Q is a definition of R in terms of D within V*. Consider the following table. Assume steps (1) and (2) which are preconditions of the proposition and let E be a formula that satisfies statement (2.), that is, assume steps (3) and (4). The first conjunct of statement (1.) is immediate from (3). With step (11) we derive the second conjunct of statement (1.), where, to derive step (5) we apply Prop. 50.

(1) $\mathcal{V} = \langle V, U, D\rangle$ is a view definition. assumption
(2) $B \Subset D$. assumption
(3) $E \Subset U$. assumption
(4) $\dot\forall R \Subset U$:
$\quad \dot\forall Q \Subset D : \mathsf{DEF}(V, D, R, Q) \Rightarrow$
$\quad\quad \mathsf{gwsc}_\emptyset(E, R) \models \mathsf{gwsc}_\emptyset(B, Q)$. assumption
(5) $\dot\forall R \Subset U$:
$\quad \dot\forall Q \Subset D : \mathsf{DEF}(V, D, R, Q) \Rightarrow$
$\quad\quad \mathsf{gwsc}_\emptyset(E, R) \models \mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V, B), R)$. by (4), Prop. 50
(6) $\dot\forall R \Subset U$:
$\quad (\dot\exists Q \Subset D \mathbin{\dot\wedge} \mathsf{DEF}(V, D, R, Q)) \Rightarrow$
$\quad\quad \mathsf{gwsc}_\emptyset(E, R) \models \mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V, B), R)$. equiv. to (5)
(7) $\dot\forall R \Subset U$:
$\quad \dot\exists Q \Subset D : \mathsf{DEF}(V, D, R, Q)$. by (1), cond. (1.) of Def. 45
(8) $\dot\forall R \Subset U$:
$\quad \mathsf{gwsc}_\emptyset(E, R) \models \mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V, B), R)$. by (7), (6)
(9) $\dot\forall R \Subset U$:
$\quad (E \models R \Rightarrow \mathsf{gsnc}_U(V, B) \models R)$. by (8), Prop. 14
(10) $\mathsf{project}_U(\mathsf{gsnc}_U(V, B)) \models E$. by (9), Prop. 11.i
(11) $\mathsf{gsnc}_U(V, B) \models E$. by (10), Prop. 13.ix

□

23 View-Based Query Answering and Rewriting

The query *answering* approach to view-based query processing, to be contrasted with query *rewriting* [CGLV00,CGLV07], is based on the notion of *certain answer* [AD98], which applies to answer tuples. We use here the term *certain answer tuple* instead, for consistency with our use of *answer* to denote full answers instead of components like tuples. The idea is as follows: For a given view definition $\mathcal{V} = \langle V, U, D\rangle$ and formula $E \in U$ there might be several databases, that is, formulas $B \Subset D$, such that $\mathcal{V}$ is a sound view extension for B. Given a query, an answer tuple that is in the answer obtained with respect to each of these databases is called a *certain answer tuple* to the query. We consider here certain answer tuples always *under sound views* [CGLV07]. Definition 55 below characterizes certain answer tuples in terms of the GWSC. The subsequent Prop. 56 shows their characteristic properties, matching the informal explanation above, and Examp. 57 illustrates the concept with examples.

Definition 55 (Certain Answer Tuple). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q(\boldsymbol{x}) \Subset D$ and $E \Subset U$ be formulas. Then an n-tuple $\mathbf{c}$ of constants is a *certain answer tuple to* $Q(\boldsymbol{x})$ *with respect to* $\mathcal{V}$ *and* E if and only if

$$\mathsf{gwsc}_D(V, E) \models Q(\mathbf{c}).$$

Proposition 56 (Characteristics of Certain Answer Tuples). *Let* $\mathcal{V} = \langle V, U, D\rangle$ *be a view definition, let* $E \Subset U$ *and* $Q(\boldsymbol{x}) \Subset D$ *be formulas and let* $\mathbf{c}$ *be an* n*-tuple of constants. Then the following statements are equivalent:*

1. *The tuple* $\mathbf{c}$ *is a certain answer tuple to* $Q(\boldsymbol{x})$ *with respect to* $\mathcal{V}$ *and* E.
2. *For all formulas* $B \Subset D$ *such that* E *is a sound view extension of* $\mathcal{V}$ *for* B *it holds that*
$$B \models Q(\mathbf{c}).$$

Proof (Proposition 56). Assume the following precondition of the proposition:

(1) $E \Subset U$. assumption

A derivation of the equivalence of statements (1.) to (2.) is shown in the following table, where statement (2.) is step (2) and statement (1.) is step (9):

(2)	$\dot{\forall} B \Subset D$: E is a sound view extension of $\mathcal{V}$ for $B \;\Rightarrow\; B \models Q(\mathbf{c})$	
(3)	iff $\dot{\forall} B \Subset D : \mathsf{gsnc}_U(V, B) \models E \;\Rightarrow\; B \models Q(\mathbf{c})$	by (1), Def 53
(4)	iff $\dot{\forall} B \Subset D : \mathsf{project}_U(V \wedge B) \models E \;\Rightarrow\; B \models Q(\mathbf{c})$	by exp. gsnc
(5)	iff $\dot{\forall} B \Subset D : V \wedge B \models E \;\Rightarrow\; B \models Q(\mathbf{c})$	by (1), Prop. 4.vi
(6)	iff $\dot{\forall} B \Subset D : B \models \neg V \vee E \;\Rightarrow\; B \models Q(\mathbf{c})$	
(7)	iff $\neg\mathsf{project}_{\overline{D}}(V \wedge \neg E) \models Q(\mathbf{c})$	by Prop. 11.iii
(8)	iff $\mathsf{gwsc}_D(V, E) \models Q(\mathbf{c})$	by con. gwsc
(9)	iff $\mathbf{c}$ is a certain answer to $Q(\boldsymbol{x})$ w.r.t. $\mathcal{V}$ and E.	by Def. 55

□

Example 57 (Certain Answer Tuple). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition, where $U = \{\mathsf{p}, \mathsf{q}\}$, $D = \{\mathsf{r}, \mathsf{s}\}$, and

$$V \equiv (\mathsf{p} \leftrightarrow \mathsf{r} \vee \mathsf{s}) \wedge (\mathsf{q} \leftrightarrow \mathsf{r}).$$

Let $E = \mathsf{p}$. According to Def. 55, the empty tuple $\langle\rangle$ is a certain answer tuple to a query $Q \Subset D$ with respect to $\mathcal{V}$ and E if and only if $\mathsf{gwsc}_D(V, E) \models Q$. Let us determine the GWSC:

$$\begin{aligned}
&\mathsf{gwsc}_D(V, E)\\
\equiv\ &\neg\mathsf{project}_{\overline{D}}(V \wedge \neg E)\\
\equiv\ &\neg\mathsf{project}_{\{\mathsf{r},\mathsf{s}\}}((\mathsf{p} \leftrightarrow \mathsf{r} \vee \mathsf{s}) \wedge (\mathsf{q} \leftrightarrow \mathsf{r}) \wedge \neg\mathsf{p})\\
\equiv\ &\mathsf{r} \vee \mathsf{s}.
\end{aligned}$$

If $Q = (\mathsf{r} \vee \mathsf{s})$, then, since clearly $(\mathsf{r} \vee \mathsf{s}) \models (\mathsf{r} \vee \mathsf{s})$, the empty tuple $\langle\rangle$ is a certain answer tuple to Q. If, however, we let $Q = \mathsf{r}$, then, since $(\mathsf{r} \vee \mathsf{s}) \not\models \mathsf{r}$, there is no certain answer tuple to Q. Let us now consider examples of formulas $B \Subset D$ such that E is a sound view extension of $\mathcal{V}$ for B, that is, according to Def. 53, such that it holds that $\mathsf{gsnc}_U(V, B) \models E$. If $B = \mathsf{s}$ or $B = (\mathsf{r} \vee \mathsf{s})$ it holds that $\mathsf{gsnc}_U(V, B) \equiv \mathsf{project}_U(V \wedge B) \equiv \mathsf{p} = E$. If $B = \mathsf{r}$, then $\mathsf{gsnc}_U(V, B) \equiv (\mathsf{p} \wedge \mathsf{q}) \models \mathsf{p} = E$. If $Q = (\mathsf{r} \vee \mathsf{s})$, then in all of the three considered cases for B it holds that $B \models Q$, in accordance with statement (2.) of Prop. 56. However, also in accordance with this statement, if $Q = \mathsf{r}$ and $B = \mathsf{s}$, then $B \not\models Q$. As a fourth case for B consider $B = \neg\mathsf{r}$. Then $\mathsf{gsnc}_U(V, B) \equiv \neg\mathsf{q} \not\models \mathsf{p} = E$. Thus, in this case E is not a sound view extension of $\mathcal{V}$ for B and, again in accordance with statement (2.) of Prop. 56, it is of no harm that $B \not\models (\mathsf{r} \vee \mathsf{s})$.

Following [CGLV07], the decision problem of *view-based query answering* is defined on the basis of *certain answer tuple*:

Definition 58 (View-Based Query Answering). *View-based query answering* consists in deciding whether $\mathbf{c}$ is a certain answer tuple to $Q(\boldsymbol{x})$ with respect to V and D, for given $\boldsymbol{x}, \mathbf{c}, V, E, Q(\boldsymbol{x}), D$ be as specified in Def. 55.

It is immediate from the definition of *certain answer tuple*, Def. 55, that view-based query answering according to Def. 58 amounts to deciding

$$\mathsf{gwsc}_D(V, E) \models Q(\mathbf{c}),$$

for the respective given $\boldsymbol{x}, \mathbf{c}, V, E, Q(\boldsymbol{x}), D$. The query that yields for every view extension E the set of certain answer tuples to query Q with respect to view definition $\mathcal{V}$ and E is denoted symbolically by $cert_{Q,\mathcal{V}}$ in [CGLV07]. We call it here *certain query*. It is defined in the following Def. 59 and its characteristics are stated as Prop. 60.

Definition 59 (Certain Query). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. Then the *certain query for Q and $\mathcal{V}$* is defined as

$$\mathsf{gwsc}_U(V, Q).$$

Proposition 60 (Characteristics of the Certain Query). *Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q(\boldsymbol{x}) \Subset D$ be a formula. Then the following statements are equivalent*

1. *$R(\boldsymbol{x})$ is the certain query for $Q(\boldsymbol{x})$ and $\mathcal{V}$.*
2. *$R(\boldsymbol{x}) \Subset U$, and for all n-tuples of constants and formulas $E \Subset U$ it holds that $E \models R(\mathbf{c})$ if and only if $\mathbf{c}$ is a certain answer tuple to $Q(\boldsymbol{x})$ with respect to $\mathcal{V}$ and E.*

Proof (Proposition 60). We start by showing that statement (2.) is equivalent to the following statement (E17), which is like (2.) except that in the right side of the *if and only if* is replaced by $E \models \mathsf{gwsc}_U(V, Q(\mathbf{c}))$:

$$\begin{array}{l} R(\boldsymbol{x}) \Subset U, \text{and for all } n\text{-tuples } \mathbf{c} \text{ and formulas } E \Subset U \text{ it holds that} \\ E \models R(\mathbf{c}) \text{ if and only if } E \models \mathsf{gwsc}_U(V, Q(\mathbf{c})). \end{array} \tag{E17}$$

Consider the table below. Assume (1) and (2) which are preconditions of the proposition, and step (3) which holds in the context of the substatement to be replaced.

(1)	$\mathcal{V} = \langle V, U, D\rangle$ is a view definition.	assumption
(2)	$Q(\boldsymbol{x}) \Subset D$.	assumption
(3)	$E \Subset U$.	assumption
(4)	$\mathsf{gwsc}_D(V, E) \equiv \mathsf{gsnc}_D(V, E)$.	by (3), (1) Prop. 46

Let $\mathbf{c}$ be an n-tuple of constants. We can establish the equivalence of statement (2.) and (E17) in the following steps:

(5) $\mathbf{c}$ is a certain answer tuple to $Q(\mathbf{c})$ w.r.t. V and D
(6) iff $\mathsf{gwsc}_D(V, E) \models Q(\mathbf{c})$ by Def. 55
(7) iff $\mathsf{gsnc}_D(V, E) \models Q(\mathbf{c})$ by (4)
(9) iff $V \wedge E \models Q(\mathbf{c})$ by (2), Prop. 17.i
(12) iff $E \models \mathsf{gwsc}_U(V, Q(\mathbf{c}))$. by (3), Prop. 17.ii

We now conclude the proof by showing the equivalence of (E17) to statement (1.). Consider the following table of equivalences. Statement (E17) is listed there as the first step and statement (1.) as the last step.

(13) $R(\boldsymbol{x}) \Subset U \;\dot\wedge$
$\dot\forall \mathbf{c} \in \mathsf{CONST}^n : \dot\forall E \Subset U :$
$E \models R(\mathbf{c}) \;\Leftrightarrow\; E \models \mathsf{gwsc}_U(V, Q(\mathbf{c}))$
(14) iff $R(\boldsymbol{x}) \Subset U \;\dot\wedge$
$\dot\forall E \Subset U : E \models R(\boldsymbol{x}) \;\Leftrightarrow\; E \models \mathsf{gwsc}_U(V, Q(\boldsymbol{x}))$
(15) iff $R(\boldsymbol{x}) \Subset U \;\dot\wedge$
$\mathsf{project}_{\overline{U}}(\neg R(\boldsymbol{x})) \equiv \mathsf{project}_{\overline{U}}(\neg \mathsf{gwsc}_U(V, Q(\boldsymbol{x})))$ by Prop. 11.iv
(16) iff $R(\boldsymbol{x}) \Subset U \;\dot\wedge\; R(\boldsymbol{x}) \equiv \mathsf{gwsc}_U(V, Q(\boldsymbol{x}))$ by Prop. 4.xvii, 13.x, con. gwsc
(17) iff $R(\boldsymbol{x}) \equiv \mathsf{gwsc}_U(V, Q(\boldsymbol{x}))$ by Prop. 13.x
(18) iff $R(\boldsymbol{x})$ is the certain query for. $Q(\boldsymbol{x})$ and $\mathcal{V}$. by Def. 59

□

In view-based query *rewriting* [CGLV00,CGLV07], a query over the database language is processed by first reformulating it in terms of the view language, and then evaluated with respect to the view extensions. As we will see below, the difference to view-based query answering is in essence only relevant if the allowed reformulations are constrained to be in a certain formula class such as conjunctive queries, regular-path queries, or conjunctive regular-path queries, which can not be straightforwardly specified with our semantic second-order operators. Like view-based answering, we consider view-based rewriting here always under sound views. Definition 61 below specifies the semantic conditions on the involved query reformulations.

Definition 61 (Rewriting). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. A formula R is called a *rewriting of Q with respect to $\mathcal{V}$* if and only if

1. $R \Subset U$.
2. $R \models \mathsf{gwsc}_U(V, Q)$.

In presence of condition (1.) of Def. 61, its condition (2.) is equivalent to

$$V \wedge R \models Q, \tag{E18}$$

which can be easily verified from the definition of gwsc and properties of projection. The following Prop. 62 ensures the characteristic properties of rewritings.

Proposition 62 (Characteristics of Rewriting). *Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. Then for all formulas R the following statements are equivalent:*

1. *R is a rewriting of Q with respect to $\mathcal{V}$.*
2. *$R \Subset U$, and for all formulas $B \Subset D$ and $E \Subset U$ such that E is a sound view extension of $\mathcal{V}$ for B and $E \models R$ it holds that $B \models Q$.*

Proof (Prop. 62). Consider the table below. Assume (1) and (2), which are preconditions of the proposition. Further, assume (3), which is a condition in both statements of the proposition, implicit via Def. 61 in statement (1.) and explicit in statement (2.). Step (4), a lemma that is used later on, follows from the assumptions (3) and (2).

(1)	$\mathcal{V} = \langle V, U, D\rangle$ is a view definition.	assumption
(2)	$Q \Subset D$.	assumption
(3)	$R \Subset U$.	assumption
(4)	$\mathsf{gsnc}_D(V, R) \equiv \mathsf{gwsc}_D(V, R)$.	by (3), (1) Prop. 46

We now show the equivalence of the two statements in the proposition by proceeding from statement (2.) to statement (1.). If $E \Subset U$, then $\mathsf{gsnc}_U(V, B) \models E$ is equivalent to *E is a sound view extension of $\mathcal{V}$ for B*. Thus, statement (2.) can be expressed as step (5). According to Def. 61, the final step (15) conjoined with assumption (3) is equivalent to statement (1.).

(5)	$\dot\forall B \Subset D : \dot\forall E \Subset U :$ $\mathsf{gsnc}_U(V, B) \models E \mathbin{\dot\wedge} E \models R \Rightarrow B \models Q$	
(6)	iff $\dot\forall B \Subset D : V \wedge B \models R \Rightarrow B \models Q$	
(7)	iff $\dot\forall B \Subset D : B \models \neg V \vee R \Rightarrow B \models Q$	
(8)	iff $\neg\mathsf{project}_{\overline{D}}(V \wedge \neg R) \models Q$	by Prop. 11.iii
(9)	iff $\mathsf{gwsc}_D(V, R) \models Q$	by con. gwsc
(10)	iff $\mathsf{gsnc}_D(V, R) \models Q$	by (4)
(11)	iff $V \wedge R \models Q$	by (2), Prop. 17.i
(12)	iff $R \models \mathsf{gwsc}_U(V, Q)$.	by (3), Prop. 17.ii

□

So far, a rewriting is, aside of the requirement on its scope, only constrained with respect to a formula that is must *entail*. Any stronger formula in the scope is also a rewriting. If $\mathcal{C}$ is some class of formulas, then a *$\mathcal{C}$-maximal rewriting* [CGLV07] is a rewriting that is in $\mathcal{C}$ and such that there is no weaker formula in $\mathcal{C}$ which is also a rewriting. Typical such classes are defined by syntactic constraints such as conjunctive queries, regular-path queries, and conjunctive regular-path queries. With our toolkit of semantic second-order operators, we can not express in a straightforward way such syntactic properties. We thus consider here just the semantics of the $\top$-maximal rewriting, with respect to the unconstrained class $\top$ of all formulas. It is defined in Def. 63 below, with its characteristic properties stated in Prop. 64. Actually, the $\top$-maximal rewriting is identical to the certain query (Def. 59), which means that the difference between view-based query answering and view-based query rewriting is only essential if strictly constraining classes $\mathcal{C}$ are considered.

Definition 63 ($\top$-Maximal Rewriting). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. Then the *$\top$-maximal rewriting of Q with respect to $\mathcal{V}$* is the formula

$$\mathsf{gwsc}_U(V, Q).$$

Proposition 64 (Characteristics of the $\top$-Maximal Rewriting). *Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. Then for all formulas R the following statements are equivalent:*

1. *R is the $\top$-maximal rewriting of Q with respect to $\mathcal{V}$.*
2. *The following holds:*
   a. *$R \Subset U$.*
   b. *R is a rewriting of Q with respect to $\mathcal{V}$.*
   c. *For all formulas $R' \Subset U$ it holds that if R' is a rewriting of Q with respect to $\mathcal{V}$, then $R' \models R$.*

Proof (Proposition 64). By Def. 62, considering entailment (E18), statement (2.) is equivalent to:

$$\begin{array}{ll} a.\ R \Subset U. & \\ b.\ V \wedge R \models Q. & \text{(E19)} \\ c.\ \text{For all } R' \Subset U \text{ it holds that if } V \wedge R' \models Q, \text{ then } R' \models R. & \end{array}$$

By Prop. 13.viii the statement (E19) is equivalent to $R \equiv \mathsf{gwsc}_U(V, Q)$, hence, by Def. 63 also equivalent to statement (1.), that is, R is the $\top$-maximal rewriting of Q with respect to $\mathcal{V}$.

□

24 Exactness, Losslessness and Perfectness

Three different notions of losslessness in view-based query processing are specified and related in [CGLV07], *lossless views*, *exact rewritings* and *perfect rewritings*. We provide here definitions of these concepts in terms of definability, definition, and GWSC, respectively, and show their coincidence with the characterizations in [CGLV07]. As discussed towards the end of this section, with our formal reconstruction, relationships between these concepts can be shown, which do not fully coincide with the informally presented observations in [CGLV07].

We first consider exact rewritings, that is, rewriting that are equivalent to the original query modulo the view definition. Definition 65 below correspondingly characterizes an exact rewriting just as a definition in the sense of Def. 16. Proposition 66 then ensures that an exact rewriting is indeed a rewriting in the sense of Def. 61. By Prop. 67, an exact rewriting can be characterized in the same way as in [CGLV07].

Definition 65 (Exact Rewriting). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. A formula R is then called an *exact rewriting of Q with respect to $\mathcal{V}$* if and only if R is a definition of Q in terms of U within V.

Proposition 66 (An Exact Rewriting is a Rewriting). *Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. A formula R that is an exact rewriting of Q with respect to $\mathcal{V}$ is also a rewriting of Q with respect to $\mathcal{V}$.*

Proof (Proposition 66). Assume that R is an exact rewriting of Q with respect to $\mathcal{V}$. Then, by Def. 65, R is a definition of Q in terms of U within V. Hence from Def. 16 it follows that $R \Subset U$ and that $R \models \mathsf{gwsc}_U(V, Q)$. According to Def. 61, these two conditions characterize R as a rewriting of Q with respect to $\mathcal{V}$. □

Proposition 67 (Characteristics of Exact Rewritings). *Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. Then for all formulas R the following statements are equivalent*

1. *R is an exact rewriting of Q with respect to $\mathcal{V}$.*
2. *$R \Subset U$, and for all formulas $B \Subset D$ the entailment-based extensional answer to R with respect to the exact view extension of $\mathcal{V}$ for B and the entailment-based extensional answer to Q with respect to B are equivalent.*

Proof (Proposition 67). Statement (1.) expands into *$R \Subset U$, and*

$$\mathsf{gsnc}_U(V, Q) \models R \text{ and } R \models \mathsf{gwsc}_U(V, Q). \tag{E20}$$

Statement (2.) expands into *$R \Subset U$, and for all formulas $B \Subset D$ it holds that*

$$\mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V, B), R) \equiv \mathsf{gwsc}_\emptyset(B, Q). \tag{E21}$$

Consider the following table. Assume (1) and (2), the preconditions of the proposition. Further assume (3), which is asserted by both statements in the proposition, implicitly via Def. 65 and Def. 16 in statement (1.), and explicitly in statement (2.).

(1)	$\mathcal{V} = \langle V, U, D\rangle$ is a view definition.	assumption
(2)	$Q \Subset D$.	assumption
(3)	$R \Subset U$.	assumption
(4)	$\mathsf{gsnc}_D(V, R) \equiv \mathsf{gwsc}_D(V, R)$.	by (3), (1), Prop. 46

Under assumptions (2) and (3) the equivalence (E21) is equivalent to $\mathsf{gwsc}_D(V, R) \equiv Q$, where B is not referenced:

(5)	$\dot\forall B \Subset D :\ \mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V, B), R) \equiv \mathsf{gwsc}_\emptyset(B, Q)$.	
(6)	iff $\dot\forall B \Subset D :\ \mathsf{project}_\emptyset(\mathsf{project}_U(V \wedge B) \wedge \neg R) \equiv \mathsf{project}_\emptyset(B \wedge \neg Q)$.	by exp. gwsc, gsnc
(7)	iff $\dot\forall B \Subset D :\ \mathsf{project}_\emptyset(V \wedge B \wedge \neg R) \equiv \mathsf{project}_\emptyset(B \wedge \neg Q)$	by (3), Prop. 4.v, 9.i
(8)	iff $\dot\forall B \Subset D :\ \mathsf{gwsc}_\emptyset(B, \neg(V \wedge \neg R)) \equiv \mathsf{gwsc}_\emptyset(B, Q)$	by con. gwsc
(9)	iff $\mathsf{project}_{\overline{D}}(V \wedge \neg R) \equiv \mathsf{project}_{\overline{D}}(\neg Q)$	by Prop. 15.ii
(10)	iff $\mathsf{project}_{\overline{D}}(V \wedge \neg R) \equiv \neg Q$	by (2), Prop. 4.xvii
(11)	iff $\mathsf{gwsc}_D(V, R) \equiv Q$.	by con. gwsc

We conclude the proof by showing that the statement of step (11) is, under the given preconditions, equivalent to (E20). Actually, the left-to-right direction of (11) is equivalent to $R \models \mathsf{gwsc}_U(V, Q)$:

(12) $\mathsf{gwsc}_D(V,R) \models Q$
(13) iff $\mathsf{gsnc}_D(V,R) \models Q$ by (4)
(14) iff $V \wedge R \models Q$ by (2), Prop. 17.i
(15) iff $R \models \mathsf{gwsc}_U(V,Q)$. (3), by Prop. 17.ii

The right-to-left direction of (11) is equivalent to $\mathsf{gsnc}_U(V,Q) \models R$:

(16) $Q \models \mathsf{gwsc}_D(V,R)$
(17) iff $V \wedge Q \models R$ by (2), Prop. 17.ii
(18) iff $\mathsf{gsnc}_U(V,Q) \models R$. by (3), Prop. 17.i

□

While exact rewritings correspond to definitions, lossless view definitions correspond to *definability*, that is, the existence of a definition, without the requirement to make that definition explicit or materialize it. Definition 68 below characterizes lossless view definitions correspondingly in terms of definability, completely analogous to Def. 65 for exact rewritings. Proposition 69 then relates Def. 68 to a characterization of *lossless* that follows [CGLV07].

Definition 68 (Lossless View Definition). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. The $\mathcal{V}$ is called *lossless with respect to* Q if and only if Q is definable in terms of U within V.

Proposition 69 (Characteristics of Lossless View Definitions). *Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. Then the following statements are equivalent*

1. *$\mathcal{V}$ is lossless with respect to Q.*
2. *For all formulas $B \Subset D$ it holds that the entailment-based extensional answer to the certain query for Q and $\mathcal{V}$, with respect to the exact view extension of $\mathcal{V}$ for B, is equivalent the entailment-based extensional answer to Q with respect to B.*

Proof (Proposition 69). Statement (1.) expands into

$$\mathsf{gsnc}_U(V,Q) \models \mathsf{gwsc}_U(V,Q). \tag{E22}$$

Statement (2.) expands into *for all formulas $B \Subset D$ it holds that*

$$\mathsf{gwsc}_\emptyset(\mathsf{gsnc}_U(V,B), \mathsf{gwsc}_U(V,Q)) \equiv \mathsf{gwsc}_\emptyset(B,Q). \tag{E23}$$

Consider the following table. Assume (1) and (2), the preconditions of the proposition.

(1) $\mathcal{V} = \langle V, U, D\rangle$ is a view definition. assumption
(2) $Q \Subset D$. assumption
(3) $\mathsf{gsnc}_D(V, \mathsf{gwsc}_U(V,Q)) \equiv \mathsf{gwsc}_D(V, \mathsf{gwsc}_U(V,Q))$. by (1), Prop. 46, 13.x

Analogously to steps (5)-(11) in the proof of Prop. 67, but with $\mathsf{gwsc}_U(V,Q)$ in place of R, we can derive that equivalence (E23) is equivalent to

$$\mathsf{gwsc}_D(V, \mathsf{gwsc}_U(V,Q)) \equiv Q. \tag{E24}$$

Thus, we have to show the equivalence of statement (E24) to (E22) under the assumptions (1)–(3). The left-to-right direction of (E24) is under the assumption (3) always true.

$$
\begin{array}{ll}
& \mathsf{gwsc}_D(V, \mathsf{gwsc}_U(V,Q)) \models Q \\
\text{iff} & \mathsf{gsnc}_D(V, \mathsf{gwsc}_U(V,Q)) \models Q \quad \text{by (3)} \\
\text{iff} & V \wedge \mathsf{gwsc}_U(V,Q) \models Q \quad \text{by (2), Prop. 17.i} \\
\text{iff} & V \wedge \neg Q \models \mathsf{project}_{\overline{U}}(V \wedge \neg Q) \quad \text{by exp. gwsc} \\
\text{iff} & \mathsf{true}. \quad \text{by Prop. 4.i}
\end{array}
$$

The right-to-left direction of (E24) is equivalent to (E22), which completes the proof:

$$
\begin{array}{ll}
& Q \models \mathsf{gwsc}_D(V, \mathsf{gwsc}_U(V,Q)) \\
\text{iff} & V \wedge Q \models \mathsf{gwsc}_U(V,Q)) \quad \text{by (2), Prop. 17.ii} \\
\text{iff} & \mathsf{gsnc}_U(V \wedge Q) \models \mathsf{gwsc}_U(V,Q)). \quad \text{by Prop. 17.i, 13.x}
\end{array}
$$

□

According to Def. 61, a query rewriting is a formula in the view scope that entails the certain query. The answer to a rewriting with respect to a sound view extension then entails the answer to the certain query, or, in other words, if the corresponding sets of tuples are considered as answers, the answer to the rewriting is a (not necessarily proper) subset of the answer to the certain query. In this sense query rewriting is an approximation to query answering. A *perfect rewriting* [CGLV00] is a query rewriting that is equivalent to the certain query. The definition of *perfect rewriting* in [CGLV07] takes in addition a class $\mathcal{C}$ of formulas into account, requiring that a perfect rewriting is a member of $\mathcal{C}$. However, in the semantic characterization of [CGLV07, p. 176], the class $\mathcal{C}$ actually does not play any role. It becomes relevant in investigations about whether a perfect rewriting in a certain class does exist. In fact, perfect rewritings are characterized for a given view definition and query uniquely up to equivalence, such that we can speak of *the* perfect rewriting. In the following definition we straightforwardly identify the perfect rewriting with the certain query. Proposition 71 then renders the semantic characterization of *perfect rewriting* from [CGLV07], but without taking into consideration that it must be in some formula class.

Definition 70 (Perfect Rewriting). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. Then the *perfect rewriting* of Q with respect to $\mathcal{V}$ is defined as

$$\mathsf{gwsc}_U(V, Q).$$

Proposition 71 (Characteristics of the Perfect Rewriting). *Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. Let R be a rewriting of Q with respect to $\mathcal{V}$. The following statements are then equivalent:*

1. *R is the perfect rewriting of Q with respect to $\mathcal{V}$.*
2. *For all formulas $B \Subset D$ and sound view extensions E of $\mathcal{V}$ for B it holds that the entailment-based extensional answer to the certain query for Q and $\mathcal{V}$, with respect to E, is equivalent to the entailment-based extensional answer to R with respect to E.*

Proof (Proposition 71). Statement (1.) expands into

$$R \equiv \mathsf{gwsc}_U(V, Q). \tag{E25}$$

Statement (2.) expands into: For all formulas $B \Subset D$ and $E \Subset U$ such that $\mathsf{gsnc}_U(V, B) \models E$ it holds that

$$\mathsf{gwsc}_\emptyset(E, \mathsf{gwsc}_U(V, Q)) \equiv \mathsf{gwsc}_\emptyset(E, R). \tag{E26}$$

Consider the following table. Assume (1) and (2), which are preconditions of the proposition. Steps (6) and (7) follow from them.

(1)	$\mathcal{V} = \langle V, U, D\rangle$ is a view definition.	assumption
(2)	R is a rewriting with respect to $\mathcal{V}$.	assumption
(3)	$\dot\forall E' \Subset U : E$ is definable in terms of D within V.	by (1)
(4)	$\dot\forall E' \Subset U : \dot\exists B' \Subset D : V \wedge E' \equiv V \wedge B'$.	by (3)
(5)	$\dot\forall E' \Subset U : \dot\exists B' \Subset D : V \wedge B' \models E'$.	by (4)
(6)	$\dot\forall E' \Subset U : \dot\exists B' \Subset D : \mathsf{gsnc}_U(V, B') \models E'$.	by (5), Prop. 17.i
(7)	$R \Subset U$.	by (2)

Given (6) and (7), the equivalence of (E25), expressed as the first step in the table below, and (E26), the last step, can be shown as follows:

(8)	$\dot\forall B \Subset D : \dot\forall E \Subset U : \mathsf{gsnc}_U(V, B) \models E \Rightarrow$ $\mathsf{gwsc}_\emptyset(E, \mathsf{gwsc}_U(V, Q)) \equiv \mathsf{gwsc}_\emptyset(E, R)$	
(9)	iff $\dot\forall E \Subset U : (\dot\exists B \Subset D : \mathsf{gsnc}_U(V, B) \models E) \Rightarrow$ $\mathsf{gwsc}_\emptyset(E, \mathsf{gwsc}_U(V, Q)) \equiv \mathsf{gwsc}_\emptyset(E, R)$	
(10)	iff $\dot\forall E \Subset U : \mathsf{gwsc}_\emptyset(E, \mathsf{gwsc}_U(V, Q)) \equiv \mathsf{gwsc}_\emptyset(E, R)$	by (6)
(13)	iff $\mathsf{project}_{\overline{U}}(\neg\mathsf{gwsc}_U(V, Q)) \equiv \mathsf{project}_{\overline{U}}(\neg R)$	by Prop. 15.ii
(15)	iff $R \equiv \mathsf{gwsc}_U(V, Q)$.	by (7), Prop. 13.x, 4.xvii

□

Clearly losslessness (that is, definability) is equivalent to the existence of an exact rewriting (that is, a definition). The following Proposition 72 shows a relation between all three concepts, perfectness, exactness and losslessness. In [CGLV07, p. 171] it is suggested that exactness is the conjunction of perfectness and losslessness. From Proposition 72 it follows that the conjunction of perfectness and losslessness implies exactness. However, as demonstrated by Example 73, the converse of this implication does not hold in the general setting considered here.

Proposition 72 (Perfectness Implies Equivalence of Exactness and Losslessness). *Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition and let $Q \Subset D$ be a formula. Let R be the perfect rewriting of Q with respect to $\mathcal{V}$. Then the following statements are equivalent*

1. *R is an exact rewriting of Q with respect to $\mathcal{V}$.*
2. *$\mathcal{V}$ is lossless with respect to Q.*

Proof (Proposition 72). Consider the following table. Assume step (1), the precondition of the proposition.

(1) R is the perfect rewriting of Q w.r.t. $\mathcal{V}$. assumption
(2) $R \equiv \mathsf{gwsc}_U(V, Q)$. by (1), Def. 70
(3) $R \Subset U$. by (2), Prop. 13.x
(4) $R \models \mathsf{gwsc}_U(V, Q)$. by (2)

The equivalence of statement (1.) to statement (2.) then can be derived in the following steps:

(5) R is an exact rewriting of Q with respect to $\mathcal{V}$
(6) iff $\mathsf{gsnc}_U(V, Q) \models R$ by (4), (3), Def. 65, Prop. 16
(7) iff $\mathsf{gsnc}_U(V, Q) \models \mathsf{gwsc}_U(V, Q)$ by (2)
(8) iff $\mathcal{V}$ is lossless with respect to Q. by Def. 68, Prop. 19

□

Example 73 (Exactness does not Imply Perfectness). Let $\mathcal{V} = \langle V, U, D\rangle$ be a view definition, where $V = ((\mathsf{p} \leftrightarrow \mathsf{a}) \wedge (\mathsf{q} \leftrightarrow \mathsf{a}))$, $U = \{\mathsf{p}, \mathsf{q}\}$ and $D = \{\mathsf{a}\}$, and let $Q = \mathsf{a}$. It then holds that

$$\mathsf{gsnc}_U(V, Q) \equiv (\mathsf{p} \wedge \mathsf{q}) \models (\mathsf{p} \vee \mathsf{q}) \equiv \mathsf{gwsc}_U(V, Q). \tag{E27}$$

From (E27) it is immediate that $(\mathsf{p} \wedge \mathsf{q})$ is an exact rewriting but is not equivalent to the perfect rewriting $(\mathsf{p} \vee \mathsf{q})$. The situation in the example is that Q is definable – but not uniquely definable – in terms of U within V, as can be derived from (E27) with the respective definitions Def. 19 and 23.

25 Summary

Table 1 summarizes the concepts involved in view-based query processing. It shows their formal definitions in terms of GWSC/GSNC, as well as their various verbal names, used here and in the literature on view-based query processing. Some of the items in the table require additional explanation: In the specification of the involved formulas we use the $\Subset$ symbol to express that the respective formula is in the indicated scope. For *view definition*, the shown characterization incorporates Prop. 46. Maximal and perfect rewriting differ from the certain query in subtle respects that are not captured in the table: The maximal rewriting with respect to the class $\top$ of all formulas is equivalent to the certain query, but with respect to other classes of formulas it is not necessarily equivalent. The perfect rewriting might be coupled with additional non-semantic conditions. If there is no formula that meets these conditions and is equivalent to the certain query, then one would say that no perfect rewriting exists.

Table 1. Concepts of View-Based Query Processing – Summary

The involved scopes are: U (view scope), D (database scope)

The involved formulas are: V (view specification), $Q \Subset D$ (query), $R \Subset U$ (rewritten query), $B \Subset D$ (database), $E \Subset U$ (view extension)

$\mathcal{V} = \langle V, U, D \rangle$ is a **view definition** iff

$$\text{For all } R \Subset U : \mathsf{gsnc}_D(V, R) \equiv \mathsf{gwsc}_D(V, R) \text{ and}$$
$$\text{for all } B \Subset D : \mathsf{gsnc}_D(V, B) \equiv B$$

$\mathsf{gwsc}_U(V, Q)$ is
- the **globally weakest sufficient condition** of Q on U within V
- the **certain query** w.r.t. Q and $\mathcal{V}$
- the **perfect rewriting** of Q w.r.t. $\mathcal{V}$
- the $\top$**-maximal rewriting** of Q w.r.t. $\mathcal{V}$

A formula R s.t. $R \Subset U$ and $R \models \mathsf{gwsc}_U(V, Q)$ is
- a **rewriting** of Q w.r.t. $\mathcal{V}$

A formula R s.t. $R \Subset U$, $\mathsf{gsnc}_U(V, Q) \models R$ and $R \models \mathsf{gwsc}_U(V, Q)$ is
- a **definition** of Q i.t.o. U within V
- an **exact rewriting** of Q w.r.t. $\mathcal{V}$

$\mathsf{gsnc}_U(V, Q) \models \mathsf{gwsc}_U(V, Q)$ holds iff
- Q is **definable** i.t.o. U within V
- $\mathcal{V}$ is **lossless** w.r.t. Q

$\mathsf{gsnc}_U(V, B)$ is
- the **globally strongest necessary condition** of B on U within V
- the **exact view extension** of $\mathcal{V}$ for B

A formula E s.t. $E \Subset U$ and $\mathsf{gsnc}_U(V, B) \models E$ is
- a **sound view extension** of $\mathcal{V}$ for B

An n-tuple $\mathbf{c}$ of constants s.t. $\mathsf{gwsc}_D(V, E) \models Q(\mathbf{c})$ is
- a **certain answer tuple** to $Q(\boldsymbol{x})$ w.r.t. $\mathcal{V}$ and E

Chapter 6

Split Rewritings

26 Characterization of Split Rewritings

In this chapter we consider the scenario where a query against a combination of a database with a knowledge base in some expressive logic is evaluated in two steps, by first constructing from the given query and knowledge base a database query, and then evaluating this database query against the database with standard technologies, which has been investigated in [BdBF$^+$10,FKN12,FKN13]. Notice that, unlike in the approach of view-based query processing discussed in Chap. 5, the vocabulary of the original query is not assumed to be in the database vocabulary, but can include concepts from the knowledge base part. The underlying model comprises exact view extensions and exact rewritings. This model has also been used in the investigations of [Mar07,NSV10], however in the different scenario of view-based query processing described in Sect. 20. We call here the database query obtained in the first step from the given query and knowledge base a *split rewriting*, defined as follows:

Definition 74 (Split Rewriting). Let D be a scope, let K, Q be formulas and let $\mathcal{C}$ be a class of formulas. A formula R is called a *split rewriting* of Q in terms of D with respect to K and $\mathcal{C}$ if and only if

1. $R \Subset D$,
2. For all formulas $B \in \mathcal{C}$ it holds that the entailment-based extensional answer to R with respect to B is equivalent to the entailment-based extensional answer to Q with respect to $(B \wedge K)$.

Condition (2.) in Def. 74 expands into *for all $B \in \mathcal{C}$ it holds that*

$$\mathsf{gwsc}_{\emptyset}(B, R) \equiv \mathsf{gwsc}_{\emptyset}(B \wedge K, Q). \tag{E28}$$

The intuition in Def. 74 is that scope D represents the set of database predicates, class $\mathcal{C}$ specifies the conditions on formulas B to be considered as databases, for example, that B uniquely defines D, formula K is an arbitrary formula, expressing additional knowledge such as an ontology, constraints or definitions of view predicates, possibly involving predicates not in D, and Q, again an arbitrary formula, expresses the query.

In case $\mathcal{C}$ includes the condition that its members B define D uniquely in terms of $\emptyset$, by Prop. 39 it follows that the *consistency*-based answer to a split rewriting R is equivalent to the entailment-based answer. Definition 74 then could be also expanded to *for all $B \in \mathcal{C}$ it holds that*

$$\mathsf{gsnc}_{\emptyset}(B, R) \equiv \mathsf{gwsc}_{\emptyset}(B \wedge K, Q). \tag{E29}$$

27 The GWSC and Definitions as Split Rewritings

Proposition 76 and 77 below each provide the foundation of an alternate approach to compute split rewritings. These propositions ensure that certain formulas, which might involve second-order operators, are split rewritings with respect to certain classes of database formulas. They are based on the following auxiliary properties of the GWSC and GSNC:

Proposition 75 (Splitting Lemma). *Let S, D be scopes and let $B \Subset D, K$, Q be formulas. It then holds that*

(i) *If $S \subseteq D$, then $\mathsf{gwsc}_S(B \wedge K, Q) \equiv \mathsf{gwsc}_S(B, \mathsf{gwsc}_D(K, Q))$.*
(ii) *If $S \subseteq \overline{D}$, then $\mathsf{gsnc}_S(B \wedge K, Q) \equiv \mathsf{gsnc}_S(B, \mathsf{gsnc}_{\overline{D}}(K, Q))$.*

Proof (Proposition 75).

(75.i) Assume the preconditions of the proposition, that is:

(A1) $S \subseteq D$.
(A2) $B \Subset D$.

The proposition then can be derived in the following steps:

$$
\begin{aligned}
& \mathsf{gwsc}_S(B \wedge K, Q) \\
\equiv\ & \neg\mathsf{project}_{\overline{S}}(B \wedge K \wedge \neg Q) \\
\equiv\ & \neg\mathsf{project}_{\overline{S}}(\mathsf{project}_{\overline{D}}(B \wedge K \wedge \neg Q)) && \text{by (A1), Prop. 4.v} \\
\equiv\ & \neg\mathsf{project}_{\overline{S}}(\mathsf{project}_{\overline{D}}(B \wedge \mathsf{project}_{\overline{D}}(K \wedge \neg Q))) && \text{by (A2), Prop. 9.i} \\
\equiv\ & \neg\mathsf{project}_{\overline{S}}(B \wedge \mathsf{project}_{\overline{D}}(K \wedge \neg Q)) && \text{by (A1), Prop. 4.v} \\
\equiv\ & \neg\mathsf{project}_{\overline{S}}(B \wedge \neg\mathsf{gwsc}_D(K, Q)) \\
\equiv\ & \mathsf{gwsc}_S(B, \mathsf{gwsc}_D(K, Q)).
\end{aligned}
$$

(75.ii) Assume the preconditions of the proposition, that is:

(A1) $S \subseteq \overline{D}$.
(A2) $B \Subset D$.

The proposition then can be derived in the following steps:

$$
\begin{aligned}
& \mathsf{gsnc}_S(B \wedge K, Q) \\
\equiv\ & \mathsf{project}_S(B \wedge K \wedge Q) \\
\equiv\ & \mathsf{project}_S(\mathsf{project}_{\overline{D}}(B \wedge K \wedge Q)) && \text{by (A1), Prop. 4.v} \\
\equiv\ & \mathsf{project}_S(\mathsf{project}_{\overline{D}}(B \wedge \mathsf{project}_{\overline{D}}(K \wedge Q))) && \text{by (A2), Prop. 9.i} \\
\equiv\ & \mathsf{project}_S(B \wedge \mathsf{project}_{\overline{D}}(K \wedge Q)) && \text{by (A1), Prop. 4.v} \\
\equiv\ & \mathsf{project}_S(B \wedge \mathsf{gsnc}_{\overline{D}}(K, Q)) \\
\equiv\ & \mathsf{gsnc}_S(B, \mathsf{gsnc}_{\overline{D}}(K, Q)).
\end{aligned}
$$

Proposition 76 (The GWSC as a Split Rewriting). *Let D be a scope, let $\mathcal{C} = \{B \mid B \Subset D\}$ and let K, Q be formulas. Then*

$$\mathsf{gwsc}_D(K, Q)$$

is a split rewriting of Q in terms of D with respect to K and $\mathcal{C}$.

Proof (Proposition 76). By Prop. 13.x it holds that $\mathsf{gwsc}_D(K, Q) \Subset D$, corresponding to condition (1.) of Def. 74. Condition (2.), which expands into *for all $B \Subset D$ it holds that* $\mathsf{gwsc}_\emptyset(B, \mathsf{gwsc}_D(K, Q)) \equiv \mathsf{gwsc}_\emptyset(B \wedge K, Q)$, follows from Prop. 75.i.

Proposition 77 (Definitions as Split Rewritings). *Let D be an atom scope, let*

$$\mathcal{C} = \{B \mid B \Subset D \text{ and } B \text{ uniquely defines } D \text{ in terms of } \emptyset\}$$

and let K, Q be formulas such that for all formulas $B \in \mathcal{C}$ it holds that

$$\models \mathsf{project}_{\emptyset}(B \wedge K).$$

Any definition R of Q in terms of D within K is then a split rewriting of Q in terms of D with respect to K, and $\mathcal{C}$.

Proof (Proposition 77). Condition (1.) of the definition of *split rewriting* is immediate from Def. 16. It remains to show condition (2.), which expands into *for all $B \Subset D$ such that B uniquely defines D in terms of $\emptyset$ it holds that*

$$\mathsf{gwsc}_{\emptyset}(B, R) \equiv \mathsf{gwsc}_{\emptyset}(B \wedge K, Q). \tag{E30}$$

Consider the table below. Assume steps (1) and (2) which are preconditions of the proposition. Let B be a formula in $\mathcal{C}$, that is, assume that B satisfies (3) and (4). In addition, assume that K satisfies the precondition with respect to all members of $\mathcal{C}$ stated in the proposition. Then (5) must hold. Steps (6)–(8) follow from (2).

(1) D is an atom scope.	assumption
(2) R is a definition of Q in terms of D within K.	assumption
(3) $B \Subset D$.	assumption
(4) B uniquely defines D in terms of $\emptyset$.	assumption
(5) $\models \mathsf{project}_{\emptyset}(B \wedge K)$.	assumption
(6) $R \Subset D$.	by (2)
(7) $\mathsf{gsnc}_D(K, Q) \models R$.	by (2)
(8) $R \models \mathsf{gwsc}_D(K, Q)$.	by (2)

By (8), Prop. 13.xiv, (3), and Prop. 75.i it holds that

$$\mathsf{gwsc}_{\emptyset}(B, R) \models \mathsf{gwsc}_{\emptyset}(B, \mathsf{gwsc}_D(K, Q)) \equiv \mathsf{gwsc}_{\emptyset}(B \wedge K, Q), \tag{E31}$$

which justifies the left-to-right direction of (E30). From (3), (1), Prop. 75.ii, (7) and Prop. 13.xiii it follows that

$$\mathsf{gsnc}_{\emptyset}(B \wedge K, Q) \equiv \mathsf{gsnc}_{\emptyset}(B, \mathsf{gsnc}_D(K, Q)) \models \mathsf{gsnc}_{\emptyset}(B, R). \tag{E32}$$

It follows from (6) and (4) that R is uniquely definable in terms of $\emptyset$ within B, and thus

$$\mathsf{gsnc}_{\emptyset}(B, R) \equiv \mathsf{gwsc}_{\emptyset}(B, R). \tag{E33}$$

From (E33), (E32) and (E31) it follows that

$$\mathsf{gsnc}_{\emptyset}(B \wedge K, Q) \models \mathsf{gwsc}_{\emptyset}(B \wedge K, Q), \tag{E34}$$

that is, definability of Q in terms of $\emptyset$ within $(B \wedge K)$. From (A4) it then follows by Prop. 26 that Q is also *uniquely* definable, that is

$$\mathsf{gsnc}_{\emptyset}(B \wedge K, Q) \equiv \mathsf{gwsc}_{\emptyset}(B \wedge K, Q). \tag{E35}$$

The right-to-left direction of (E30) now follows from (E35), (E33) and (E32). □

If B and K are sentences, then the condition $\models \mathsf{project}_{\emptyset}(B \wedge K)$ in the preconditions of Prop. 77 is equivalent to $(B \wedge K)$ *is satisfiable*. While Prop. 76 and 77 show that particular formulas are split rewritings, neither of them does conversely give a characterizations of *all* split rewritings with respect to certain classes. This remains an issue for further investigations.

For first-order knowledge bases, Prop. 77 suggests to compute split rewritings by applying the reduction of definability to validity, and the correspondence of definitions to interpolants that can be extracted from proofs, as sketched in Sect. 14. This is the approach of [BdBF$^+$10,FKN12,FKN13]. Prop. 77 justifies that *any* definition is a split rewriting, permitting to choose the most appropriate one from these, with respect to further criteria. In [BdBF$^+$10,FKN12,FKN13], domain independence and the safe-range property have been identified as such criteria, since they ensure processability by standard database engines. Tableau calculi that facilitate the construction of domain independent interpolants by preserving syntactic properties guaranteeing domain independence are investigated there.

Proposition 76 suggests to apply second-order quantifier elimination or the computation of *uniform* interpolants to construct split rewritings. So far, this approach seems not to have been investigated in the database community.

Chapter 7

Conclusion

An answer to a query with respect to a knowledge base can be understood generically as a definition of the query, within the knowledge base, and meeting certain application specific requirements, such as restrictions on the used vocabulary. This notion of *answer* can be formally modeled with second-order operators. As we have seen, such a modeling provides a basis for reconstructions of view-based query processing and a related approach, called here *split rewriting*, where knowledge bases that are combined from parts in languages with different expressivity are considered. The reconstructions relate many of the database specific concepts to general logic concepts, such as *globally weakest sufficient condition* and *definition*, show several subtle new aspects and provide a ground for further investigations.

This semantic foundation should make techniques of view-based query processing and split rewriting transferable to knowledge representation systems in general. Various restrictions of component knowledge bases could be investigated, including second-order properties such as finiteness of predicate extensions, with the idea that in applications they are passed to a context where they can be dropped or easily be eliminated. We have seen an example for this with the circumscription operator, which can be dropped if consequences are considered with respect to datalog semantics. Also the exchange of techniques with other fields that can be characterized on the same formal basis should be facilitated. This concerns in particular abductive reasoning in logic programming [KKT98,DLS01,Wer13] and investigations of uniform interpolation in description logics [GLW06,KWW09,LW11,CM12,KS13].

As we have seen, particular query rewritings can be characterized as globally strongest necessary and weakest sufficient condition, respectively, and thus – under suitable language restrictions – can be computed by second-order quantifier elimination. It seems that this has so far not been considered as a computational method in research on view-based query processing. It is, however, promising in particular in presence of the recent results on methods for and completeness of uniform interpolation in description logics and since it opens up an alternative approach to compute split rewritings that does, in contrast to the methods described in the literature, not depend on Tarski's reduction of first-order definability to provability, which is well-known to fail in the finite.

Projection allows to express semantic properties as well as restrictions of the allowed symbols and polarity of predicate occurrences. Seeming limitations of the presented approach become apparent when properties that can not be straightforwardly expressed with projection come into play, for example, formula classes that are characterized by syntactic restrictions. Exploration of possibilities to

encode relevant syntactic properties into the semantics-based tools is one of the issues for future research.

Split rewriting, in particular, requires further semantic investigations: We have seen preconditions under which definitions and the globally weakest sufficient condition are split rewritings, however, converse properties, leading from split rewritings to definitions, are still missing.

Some straightforward semantic characterizations with second-order operators are up to a range of formulas, for example, *definitions* or *exact rewritings* are all the formulas that are stronger than a particular formula with second-order operators and weaker than another. It remains an issue for future research to investigate possibilities to take preference criteria for picking a solution from such a range into account, for example, to pick a solution that ensures domain independence, as investigated in the literature on split rewriting.

The generic modeling of *answer* used in the framework covers various forms of answers, including also the standard forms of extensional answers to relational and constraint query languages. It is actually closely related to abductive explanations, suggesting a further direction of research: Extending the framework to a general approach to question answering in knowledge representation and relating it to interrogative approaches from philosophy such as [Hin07].

References

[AD98] Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Conference on Principles of Database Systems, PODS '98*, pages 254–263, 1998.

[BdBF+10] Alex Borgida, Jos de Bruijn, Enrico Franconi, Inanç Seylan, Umberto Straccia, David Toman, and Grant Weddell. On finding query rewritings under expressive constraints. In *Proceedings of the 18th Italian Symposium on Advanced Database Systems, SEBD 2010*, 2010.

[CGLV00] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. What is query rewriting? In *Cooperative Information Agents IV, The Future of Information Agents in Cyberspace, 4th International Workshop, CIA 2000*, volume 1860 of *LNCS*, pages 51–59. Springer, 2000.

[CGLV07] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. View-based query processing: On the relationship between rewriting, answering and losslessness. *Theoretical Computer Science*, 371(3):169–182, 2007.

[CM12] B. Cuenca Grau and B. Motik. Reasoning over ontologies with hidden content: The import-by-query approach. *Journal of Artificial Intelligence Research*, 45, 2012.

[Cra57] W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.

[DŁS01] P. Doherty, W. Łukaszewicz, and A. Szałas. Computing strongest necessary and weakest sufficient conditions of first-order formulas. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI-01*, pages 145–151. Morgan Kaufmann, 2001.

[FKN12] Enrico Franconi, Volha Kerhet, and Nhung Ngo. Exact query reformulation with first-order ontologies and databases. In *Logics in Artificial Intelligence: 13th European Conference, JELIA 2012*, volume 7519 of *LNAI*, pages 202–214. Springer, 2012.

[FKN13] Enrico Franconi, Volha Kerhet, and Nhung Ngo. Exact query reformulation over databases with first-order and description logics ontologies. *Journal of Artificial Intelligence Research*, 48:885–922, 2013.

[GLW06] S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? A case for conservative extensions in description logics. In *Proc. 10th Int. Conf. on Princ. of Knowledge Rep. and Reasoning, KR 2006*, pages 187–197. AAAI Press, 2006.

[GSS08] D. M. Gabbay, R. A. Schmidt, and A. Szałas. *Second-Order Quantifier Elimination: Foundations, Computational Aspects and Applications*. College Publications, London, 2008.

[Hal01] A. Y. Halevy. Answering queries using views: a survey. *The VLDB Journal*, 10(4):270–294, 2001.

[Hin07] Jakko Hintikka. *Socratic Epistemology: Explorations of Knowledge-Seeking by Questioning*. Cambridge University Press, 2007.

[KKR95] Paris C. Kanellakis, Gabriel M. Kuper, and Peter Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995.

[KKT98] A. C. Kakas, R. A. Kowalski, and F. Toni. The role of abduction in logic programming. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors,

Handbook of Logic in Artifical Intelligence, volume 5, pages 235–324. Oxford University Press, 1998.

[KS13] Patrick Koopmann and Renate A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In *Logic for Programming, Artificial Intelligence and Reasoning: 19th International Conference, LPAR-19*, volume 8312 of *LNCS (LNAI)*, pages 552–567. Springer, 2013.

[KWW09] Boris Konev, Dirk Walther, and Frank Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proceedings of the 21nd International Joint Conference on Artificial Intelligence, IJCAI-09*, pages 830–835. AAAI Press, 2009.

[Lin01] Fangzhen Lin. On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128:143–159, 2001.

[LLM03] Jérôme Lang, Paolo Liberatore, and Pierre Marquis. Propositional independence – formula-variable independence and forgetting. *Journal of Artificial Intelligence Research*, 18:391–443, 2003.

[LW11] Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-11*, pages 989–995. AAAI Press, 2011.

[Mar07] Maarten Marx. Queries determined by views: pack your views. In *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '07*, pages 23–30. ACM, 2007.

[Mot94] Amihai Motro. Intensional answers to database queries. *IEEE Transactions on Knowledge and Data Engineering*, 6(3):444–454, 1994.

[NSV10] Alan Nash, Luc Segoufin, and Victor Vianu. Views and queries: Determinacy and rewriting. *ACM Transactions on Database Systems*, 35(3), 2010.

[Tar35] Alfred Tarski. Einige methologische Untersuchungen zur Definierbarkeit der Begriffe. *Erkenntnis*, 5:80–100, 1935.

[Ull89] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I and II. Computer Science Press, 1988, 1989.

[Wer08] Christoph Wernhard. Literal projection for first-order logic. In *Logics in Artificial Intelligence: 11th European Conference, JELIA 08*, volume 5293 of *LNAI*, pages 389–402. Springer, 2008.

[Wer09] Christoph Wernhard. *Automated Deduction for Projection Elimination*. Number 324 in Dissertationen zur Künstlichen Intelligenz (DISKI). AKA/IOS Press, Heidelberg/Amsterdam, 2009.

[Wer12] Christoph Wernhard. Projection and scope-determined circumscription. *Journal of Symbolic Computation*, 47:1089–1108, 2012.

[Wer13] Christoph Wernhard. Abduction in logic programming as second-order quantifier elimination. In *9th International Symposium on Frontiers of Combining Systems, FroCoS 2013*, volume 8152 of *LNAI*, pages 103–119. Springer, 2013.