When is Naïve Evaluation Possible?

Amélie Gheerbrant, Leonid Libkin, Cristina Sirangelo

When is Naïve Evaluation Possible?

Amélie Gheerbrant
School of Informatics
University of Edinburgh
agheerbr@inf.ed.ac.uk

Leonid Libkin
School of Informatics
University of Edinburgh
libkin@inf.ed.ac.uk

Cristina Sirangelo
LSV, ENS-Cachan
INRIA & CNRS
sirangel@lsv.ens-cachan.fr

ABSTRACT

The term naïve evaluation refers to evaluating queries over incomplete databases as if nulls were usual data values, i.e., to using the standard database query evaluation engine. Since the semantics of query answering over incomplete databases is that of certain answers, we would like to know when naïve evaluation computes them: i.e., when certain answers can be found without inventing new specialized algorithms. For relational databases it is well known that unions of conjunctive queries possess this desirable property, and results on preservation of formulae under homomorphisms tell us that within relational calculus, this class cannot be extended under the open-world assumption.

Our goal here is twofold. First, we develop a general framework that allows us to determine, for a given semantics of incompleteness, classes of queries for which naïve evaluation computes certain answers. Second, we apply this approach to a variety of semantics, showing that for many classes of queries beyond unions of conjunctive queries, naïve evaluation makes perfect sense under assumptions different from open-world. Our key observations are: (1) naïve evaluation is equivalent to monotonicity of queries with respect to a semantics-induced ordering, and (2) for most reasonable semantics, such monotonicity is captured by preservation under various types of homomorphisms. Using these results we find classes of queries for which naïve evaluation works, e.g., positive first-order formulae for the closed-world semantics. Even more, we introduce a general relation-based framework for defining semantics of incompleteness, show how it can be used to capture many known semantics and to introduce new ones, and describe classes of first-order queries for which naïve evaluation works under such semantics.

PODS'13, June 22–27, 2013, New York, New York, USA.

Categories and Subject Descriptors. H.2.1 [**Database Management**]: Logical Design—*Data Models*; H.2.1 [**Database Management**]: Languages—*Query Languages*; H.2.4 [**Database Management**]: Systems—*Query Processing*
General Terms. Theory, Languages, Algorithms
Keywords. Incompleteness, naive tables/evaluation, certain answers, orderings, homomorphisms

1. INTRODUCTION

Database applications need to handle incomplete data; this is especially true these days due to the proliferation of data obtained as the result of integrating or exchanging data sets, or data found on the Web. At the same time, there is a huge gap between our theoretical knowledge and the handling of incompleteness in practice:

- In SQL, the design of null-related features is one of the most criticized aspects of the language [13], due to the oversimplification of the model (which even leads to paradoxical behavior: it is consistent with SQL's semantics that $|X| > |Y|$ and $X - Y = \varnothing$, if the set Y contains nulls!)
- In theory, we understand that the proper way of evaluating queries on incomplete databases is to find *certain answers* to them. Unfortunately, for many classes of queries, even within first-order logic, this is an intractable problem [2], and even when it is tractable, there is no guarantee the algorithms can be easily implementable on top of commercial DBMSs [15].

Despite this seemingly enormous gap, there is one instance when theoretical approaches and functionalities of practical systems converge nicely. For some types of queries, evaluating them on the incomplete database itself (i.e. as if nulls were the usual data values) does produce certain answers. This is usually referred to as *naïve evaluation* [1, 19]. To give an example, consider databases with *naïve nulls* (also called marked nulls), that appear most commonly in integration and exchange scenarios, and that can very easily be supported by commercial RDBMSs. Two such relations

are shown below, with nulls indicated by the symbol $\perp$ with subscripts:

R:

A	B
1	$\perp_1$
$\perp_2$	$\perp_3$

S:

B	C
$\perp_1$	4
$\perp_3$	5

Suppose we have a conjunctive query $\pi_{AC}(R \bowtie S)$ or, equivalently, $\varphi(x, y) = \exists z \big(R(x, z) \wedge S(z, y)\big)$. Naïve evaluation says: evaluate the query directly on R and S, proceed as if nulls were usual values; they are equal only if they are syntactically the same (for instance $\perp_1 = \perp_1$ but $\perp_1 \neq \perp_2$). Then evaluating the above query results in two tuples: $(1, 4)$, and $(\perp_2, 5)$. Tuples with nulls cannot be certain answers, so we only keep the tuple $(1, 4)$.

One does not need any new functionalities of the DBMS to find the result of naïve evaluation (in fact most implementations of marked nulls are such that equality tests for them are really the syntactic equality). This is good, but in general, naïve evaluation need not compute certain answers. Recall that these are answers which hold true in all possible complete databases represented by the incomplete one, under some semantics of incompleteness.

For the query above, the tuple $(1, 4)$ is however the certain answer, under the common open-world semantics (to be properly defined later). This is true because [19] showed that if Q is a union of conjunctive queries, then naïve evaluation works for it (i.e., computes certain answers). This result is not so easy to extend: for instance, [24] showed that under the open-world semantics, if naïve evaluation works for a Boolean first-order (FO) query Q, then Q must be equivalent to a union of conjunctive queries. That result crucially relied on a preservation theorem from mathematical logic [11], and in particular on its version over finite structures [30].

This observation suggests that the limits of naïve evaluation depend on the semantics of incompleteness, and that syntactic restrictions on queries admitting such evaluation may be obtained from preservation theorems in logic. This is the starting point of our investigation. In general we would like to understand how, for a given semantics of incompleteness, we can find the class of queries for which certain answers will be found naïvely.

In slightly more detail, we would like to answer the following three questions:

1. What are the most general conditions underlying naïve evaluation, under different semantics?
2. When can naïve evaluation be characterized by preservation results?
3. How can we find relevant classes of queries that admit naïve evaluation?

We answer these three questions, by clarifying the relationship between semantics, naïve evaluation, preservation, and syntactic classes. Roughly, our results can be seen as establishing the following equivalences:

Naïve evaluation works for a query Q
$\Vert$
Q is monotone wrt the semantic ordering
$\Vert$
Q is preserved under a class of homomorphisms

We now explain the key ideas behind the main equivalences and the terminology we use.

Naïve evaluation and monotonicity For the first group of results, we deal with a very abstract setting that can be applied to many data models (relational, XML, etc) under different semantics. We assume that incomplete database objects x come with a notion of semantics $[\![x]\!]$, which is the set of complete objects they describe. We define the semantic ordering in the standard way: $x \preceq y \Leftrightarrow [\![y]\!] \subseteq [\![x]\!]$ (that is, x is less informative if it describes more objects, i.e., has more incompleteness in it). In this setting we define queries, naïve evaluation, and certain answers and prove that under very mild conditions, naïve evaluation works for a query iff it is monotone with respect to the semantic ordering. In fact, under even milder conditions, naïve evaluation corresponds to a weak notion of monotonicity, that only considers going from an object x to a more informative object $y \in [\![x]\!]$.

Monotonicity and preservation We next connect monotonicity with preservation. To start, we analyze multiple semantics of incompleteness, and come up with a uniform scheme for generating them. The key observation is that each semantics is obtained in two steps. In step one, common to all interpretations, we substitute constant values for nulls. Step two, that essentially defines the semantics, is given by a relation R showing how the result of the substitution can be modified. For instance, under the open-world semantics, tuples can be added; under the strictest form of the closed-world semantics, nothing can be changed at all.

Having done that, we prove that under some very mild condition, monotonicity of a query Q corresponds to preservation under homomorphisms that respect relation R: that is, if Q is true in D (say, for a Boolean Q), and we have a homomorphism respecting R from D to D', then Q is true in D'. Instances of such homomorphisms are the usual homomorphisms, under the open-world semantics, or onto homomorphisms, under (a version of) the closed-world semantics.

Preservation and syntactic classes We have so far established that naïve evaluation is captured by preservation under a class of homomorphisms. Such preservation results are classical in mathematical logic [11], and thus we would like to use them to find syntactic classes of queries for which naïve evaluation works.

One immediate difficulty is that classical logic results are proved for infinite structures, and they tend to fail in

the finite [4, 32], or are notoriously hard to establish (a well-known example is Rossman's theorem [30], which answered a question opened for many years). Thus, we are in general happy with good sufficient conditions for preservation, especially if they are given by nice syntactic classes corresponding to meaningful classes of database queries. The key ideas behind the classes we use are restrictions to positive formulae (admitting $\forall$ but disallowing $\neg$) or existential positive formulae (i.e., unions of conjunctive queries), and extending some of them with universally quantified *guarded* formulae.

This gives us a good understanding of what is required to make naïve evaluation work. In Sections 3–5 we carry out the program outlined above and obtain classes of FO queries for which naïve evaluation works under standard relational semantics. Also, to keep notations simple initially, in these early sections we deal with Boolean queries (all results extend to arbitrary queries easily, as we show in Section 8).

In Sections 6, 7, and 9 we offer a more detailed study of other relational semantics of incompleteness. We take a closer look at semantic orderings, explain their justification via updates that incrementally improve informativeness of a database, and compare them with known orderings on Codd databases, that model SQL's null features. We show that capturing one of such well known orderings on Codd databases leads to a new class of *powerset* semantics, and we provide preservation results for that class, using the general methodology established earlier. We then look at *minimal* semantics that find their justification in the study of various forms of the closed world assumption. For them, the notion of a core of an instance [17] plays a crucial role: for example, naïve evaluation for previously considered classes of queries is only guaranteed over cores.

Organization In Section 2, we give the main definitions. In Section 3, we explain the connection between naïve evaluation and monotonicity, and in Section 4 we relate monotonicity to preservation. In Section 5 we deal with Boolean FO queries and provide sufficient conditions for naïve evaluation. In Section 6, we study semantic orderings on incomplete databases, and in Section 7 we study naïve evaluation for the resulting new class of semantics. Section 8 shows how to lift all the results for Boolean queries to queries with free variables. In Section 9, we carry out a similar program for minimal semantics.

2. PRELIMINARIES

Incomplete databases We begin with some standard definitions. In incomplete databases there are two types of values: constants and nulls. The set of constants is denoted by Const and the set of nulls by Null. These are countably infinite sets. Nulls will normally be denoted by $\bot$, sometimes with sub- or superscripts.

A relational schema (vocabulary) is a set of relation names with associated arities. An incomplete relational instance D assigns to each k-ary relation symbol S from the vocabulary a k-ary relation over $\mathsf{Const} \cup \mathsf{Null}$, i.e., a finite subset of $(\mathsf{Const} \cup \mathsf{Null})^k$. Such incomplete relational instances are referred to as *naïve* databases [1, 19]; note that a null $\bot \in \mathsf{Null}$ can appear multiple times in such an instance. If each null $\bot \in \mathsf{Null}$ appears at most once, we speak of *Codd* databases [1, 19]. If we talk about single relations, it is common to refer to them as naïve tables and Codd tables.

We write $\mathsf{Const}(D)$ and $\mathsf{Null}(D)$ for the sets of constants and nulls that occur in a database D. The *active domain* of D is $\mathrm{adom}(D) = \mathsf{Const}(D) \cup \mathsf{Null}(D)$. A *complete* database D has no nulls, i.e., $\mathrm{adom}(D) \subseteq \mathsf{Const}$.

Homomorphisms They are crucial for us in two contexts: to define the semantics of incomplete databases, and to define the notion of preservation of logical formulae as a condition for naïve evaluation to work.

Given two relational structures D and D', a homomorphism $h : D \to D'$ is a map from the active domain of D to the active domain of D' so that for every relation symbol S, if a tuple $\bar{u}$ is in relation S in D, then the tuple $h(\bar{u})$ is in the relation S in D'.

In database literature, it is common to require that homomorphisms preserve elements of Const, i.e., the map h is also required to satisfy $h(c) = c$ for every $c \in \mathsf{Const}$. Of course this can easily be cast as a special instance of the general notion, simply by extending the vocabulary with a constant symbol for each $c \in \mathsf{Const}$. To make clear what our assumptions are, whenever there is any ambiguity, we shall talk about *database homomorphisms* if they are the identity on Const.

Given a homomorphism h and a database D, by $h(D)$ we mean the image of D, i.e., the set of all tuples $S(h(\bar{u}))$ where $S(\bar{u})$ is in D. If $h : D \to D'$ is a homomorphism, then $h(D)$ is a subinstance of D'.

Semantics and valuations We shall see many possible semantics for incomplete information, but first we review two common ones: open-world and closed-world semantics. We need the notion of a *valuation*, which assigns a constant to each null. That is, a valuation is a database homomorphism whose image contains only values in Const.

In general, the semantics $[\![D]\!]$ of an incomplete database is a set of *complete* databases D'. The semantics under the closed-world assumption (or CWA *semantics*) is defined as

$$[\![D]\!]_{\mathrm{CWA}} \;=\; \{h(D) \mid h \text{ is a valuation}\}.$$

The semantics under the open-world assumption (or OWA *semantics*) is defined as

$$[\![D]\!]_{\mathrm{OWA}} \;=\; \left\{ D' \;\middle|\; \begin{array}{l} D' \text{ is complete and} \\ \text{there is a valuation } h : D \to D' \end{array} \right\}.$$

Alternatively, $D' \in [\![D]\!]_{\mathrm{OWA}}$ iff D' is complete and contains a database $D'' \in [\![D]\!]_{\mathrm{CWA}}$ as a subinstance.

As an example, consider $D_0 = \{(\bot, \bot'), (\bot', \bot)\}$. Then $[\![D_0]\!]_{\text{CWA}}$ consists of all instances $\{(c, c'), (c', c)\}$ with $c, c' \in \mathsf{Const}$ (and possibly $c = c'$), and $[\![D_0]\!]_{\text{OWA}}$ has all complete instances containing $\{(c, c'), (c', c)\}$, for $c, c' \in \mathsf{Const}$.

Certain answers and naïve evaluation Given an incomplete database D, a semantics of incompleteness $[\![\,]\!]$, and a query Q, one normally computes *certain answers under the* $[\![\,]\!]$ *semantics*:

$$\mathsf{certain}(Q, D) = \bigcap \{Q(R) \mid R \in [\![D]\!]\},$$

i.e., answers that are true regardless of the interpretation of nulls under the given semantics. Even for first-order queries, the standard semantics are problematic in general: finding certain answers under the OWA semantics may be undecidable, and finding them under the CWA semantics may be coNP-hard [2].

Naïve evaluation of a query Q refers to a two-step procedure: first, evaluate Q on the incomplete database itself, as if nulls were values (i.e., equal iff they are syntactically the same: e.g., $\bot_1 = \bot_1$, $\bot_1 \neq \bot_2$, $\bot_1 \neq c$ for every $c \in \mathsf{Const}$), and then eliminate tuples with nulls from the result. Note that if Q is a Boolean query, the second step is unnecessary.

We say that *naïve evaluation works for* Q (under semantics $[\![\,]\!]$) if its result is exactly the certain answers under $[\![\,]\!]$, for every D.

Fact 1. (see [19, 24]) *Let Q be a union of conjunctive queries. Then naïve evaluation works for Q under both* OWA *and* CWA*. Moreover, if Q is a Boolean FO query and naïve evaluation works for Q under* OWA*, then Q is equivalent to a union of conjunctive queries.*

The last equivalence result only works under the OWA semantics. Consider again the instance D_0 and a query $\exists x, y\, D(x, y) \wedge D(y, x)$. The certain answer to this query is true under both OWA and CWA, and indeed it evaluates to true naïvely over D_0. On the other hand, a query Q given by $\forall x \exists y\, D(x, y)$ (not equivalent to a union of conjunctive queries) evaluated naïvely, returns true on D_0, but under OWA its certain answer is false. However, under CWA, its certain answer is true. This is not an isolated phenomenon: we will later see that Q belongs to a class, extending unions of conjunctive queries, for which naïve evaluation works under CWA on all databases.

Note that in this paper we assume the active domain semantics for relational first-order queries.

3. NAÏVE EVALUATION AND MONOTONICITY

The goal of this section is twofold. First we present a very general setting for talking about incompleteness and its semantics, as well as orderings representing the notion of "having more information". We formulate the notion of naïve evaluation in this setting, and show that it guarantees to compute certain answers for monotone queries.

Database domains, semantics, and ordering We now define a simple abstract setting for handling incompleteness. We operate with just four basic concepts: the set of instances, the set of complete instances, their isomorphism, and their semantics.

A *database domain* is a structure $\mathbb{D} = \langle \mathcal{D}, \mathcal{C}, [\![\cdot]\!], \approx \rangle$, where $\mathcal{D}$ is a set, $\mathcal{C}$ is a subset of $\mathcal{D}$, the function $[\![\cdot]\!]$ is from $\mathcal{D}$ to nonempty subsets of $\mathcal{C}$, and $\approx$ is an equivalence relation on $\mathcal{D}$. The interpretation is as follows:

- $\mathcal{D}$ is a set of database objects (e.g., incomplete relational databases over the same schema),
- $\mathcal{C}$ is the set of complete objects (e.g., databases without nulls);
- $[\![x]\!] \subseteq \mathcal{C}$ is the semantics of an incomplete database x, i.e., the set of all complete databases that x can represent; and
- $\approx$ is the structural equivalence relation, that we need to describe the notion of generic queries; for instance, for relational databases, $D \approx D'$ means that they are isomorphic as objects, i.e., $\pi(D) = D'$ for some 1-1 mapping on data values in D.

The semantic function of a database domain lets us describe the degree of incompleteness via an ordering defined as $x \preceq y$ iff $[\![y]\!] \subseteq [\![x]\!]$. Indeed, the less we know about an object, the more other objects it can potentially describe. This setting is reminiscent of the ideas in programming semantics, where partial functions are similarly ordered [16], and such orderings have been used to provide semantics of incompleteness in the past [9, 23, 24, 26, 31]. Note that $\preceq$ is a *preorder*.

Queries and certain answers For now we look at Boolean queries in the most abstract setting (we will generalize them later). Given a database domain $\mathbb{D} = \langle \mathcal{D}, \mathcal{C}, [\![\cdot]\!], \approx \rangle$, a *query* is a mapping $Q : \mathcal{D} \to \{0, 1\}$. We use 0 to represent *false* and 1 to represent *true*, as usual. A query is *generic* if $Q(x) = Q(y)$ whenever $x \approx y$.

For each $x \in \mathcal{D}$, the certain answer (under $[\![\,]\!]$) is

$$\mathsf{certain}(Q, x) = \bigwedge \{Q(c) \mid c \in [\![x]\!]\}$$

We say that *naïve evaluation works* for Q if $Q(x) = \mathsf{certain}(Q, x)$ for every x.

Saturation property We now impose an additional property on database domains saying, essentially, that there are enough complete objects. A database domain $\mathbb{D} = \langle \mathcal{D}, \mathcal{C}, [\![\cdot]\!], \approx \rangle$ is *saturated* if every object has a complete object in its semantics that is isomorphic to it: that is, for each $x \in \mathcal{D}$ there is $y \in [\![x]\!]$ such that $x \approx y$.

In the case of the usual semantics of incompleteness, this property trivially holds: if we have an instance D

with nulls $\bot_1, \ldots, \bot_n$, we simply replace them with distinct constants $c_1, \ldots, c_n$ that do not occur elsewhere in D, to obtain a complete database isomorphic to D.

Naïve evaluation and monotonicity We say that a query Q is *weakly monotone* if

$$y \in [\![x]\!] \quad \Rightarrow \quad Q(x) \preccurlyeq Q(y).$$

That is, if y is a complete object representing x, and Q is already true on x, then Q must be true on y. This property characterizes naïve evaluation over saturated database domains.

Theorem 1. *Let $\mathbb{D}$ be a database domain with the saturation property, and Q a generic Boolean query. Then naïve evaluation works for Q iff Q is weakly monotone.*

Of course one can also look at the natural definition of monotonicity: a query Q is *monotone* if $x \preceq y$ implies $Q(x) \preccurlyeq Q(y)$. Recall that $x \preceq y$ means that $[\![y]\!] \subseteq [\![x]\!]$. This condition turns out to be equivalent to weak monotonicity in database domains that satisfy one additional property. To state it, note that there is a natural duality between preorders and semantics: each semantics $[\![\,]\!]$ gives rise to the ordering $x \preceq y \;\Leftrightarrow [\![y]\!] \subseteq [\![x]\!]$, and conversely any preorder $\preccurlyeq$ on $\mathcal{D}$ gives a semantics $[\![x]\!]_{\preccurlyeq} = \{y \in \mathcal{C} \mid x \preccurlyeq y\}$. We say that a database domain is *fair* if $[\![\,]\!]$ and its ordering $\preceq$ agree: that is, the semantics that the ordering $\preceq$ gives rise to is $[\![\,]\!]$ itself. Fair domains can be easily characterized:

Proposition 1. *A database domain $\mathbb{D}$ is fair iff the following conditions hold:*

1. *$c \in [\![c]\!]$ for each $c \in \mathcal{C}$;*
2. *if $c \in [\![x]\!]$, then $[\![c]\!] \subseteq [\![x]\!]$.*

The standard semantics – including all those seen in the previous section – satisfy these conditions. The first condition says that the semantics of a complete object should contain at least that object. The second says that by removing incompleteness from an object, we cannot get one that denotes more objects. Note also that in a fair domain, $y \in [\![x]\!]$ implies $x \preceq y$, so weak monotonicity is indeed weaker than monotonicity.

In fair database domains, we can extend Theorem 1:

Proposition 2. *Let $\mathbb{D}$ be a fair database domain with the saturation property, and Q a generic Boolean query. Then the following are equivalent:*

1. *Naïve evaluation works for Q;*
2. *Q is monotone;*
3. *Q is weakly monotone.*

Theorem 1 and Proposition 2 establish the promised connection between monotonicity and naïve evaluation. Extension to non-Boolean queries is given in Section 8.

4. SEMANTICS, RELATIONS, AND HOMOMORPHISMS

We have seen that getting naïve evaluation to work (at least for Boolean queries), is equivalent to their (weak) monotonicity. To apply this to concrete semantics, we need to understand how different semantics can be defined. We explain that most of them are obtained by composing two types of relations: one corresponds to applying valuations to nulls, and the other to specific semantic assumptions such as open or closed-world. After that, we show a connection between naïve evaluation and preservation under a class of homomorphisms.

Semantics via relations

We have already seen two concrete relational semantics: the OWA semantics $[\![D]\!]_{\text{OWA}}$ and the CWA semantics $[\![D]\!]_{\text{CWA}}$. What is common to them is that they are all defined in two steps. First, valuations are applied to nulls (i.e., nulls are replaced by values). Second, the resulting database may be modified in some way (left as it was for CWA, or expanded arbitrarily for OWA). Our idea is then to capture this via two relations. We now define them in the setting of database domains and then show how they behave in concrete cases.

Given a database domain $\mathbb{D} = \langle \mathcal{D}, \mathcal{C}, [\![\,]\!], \approx \rangle$, we consider a pair $\mathcal{R} = (\mathcal{R}_{\text{val}}, \mathcal{R}_{\text{sem}})$ of relations:

The *valuation* relation $\mathcal{R}_{\text{val}} \subseteq \mathcal{D} \times \mathcal{C}$ between arbitrary databases and complete databases. Intuitively, a pair (x, c) is in $\mathcal{R}_{\text{val}}$ if c is obtained from x by replacing nulls by constants. The restriction of $\mathcal{R}_{\text{val}}$ to $\mathcal{C}$ is the identity: $\mathcal{R}_{\text{val}} \cap (\mathcal{C} \times \mathcal{C}) = \{(c, c) \mid c \in \mathcal{C}\}$ (if there are no nulls, there is no substitution). And since for every object there is some way to replace nulls by constants, $\mathcal{R}_{\text{val}}$ is total.

The *semantic* relation $\mathcal{R}_{\text{sem}}$ is a reflexive binary relation on $\mathcal{C}$ (i.e., $\mathcal{R}_{\text{sem}} \subseteq \mathcal{C} \times \mathcal{C}$). Intuitively, this corresponds to the modification step such as extending complete relations by new tuples. Since, at the very least, one can do nothing with the result of the substitution of nulls by constants, such a relation must be reflexive.

We say that $[\![\,]\!]$ *is given by* $\mathcal{R}$ if $\mathcal{R}$ satisfies the above conditions, and $y \in [\![x]\!]$ iff $(x, y) \in \mathcal{R}_{\text{val}} \circ \mathcal{R}_{\text{sem}}$.

Proposition 3. *Let $\mathbb{D}$ be a database domain whose semantics $[\![\,]\!]$ is given by a pair $\mathcal{R} = (\mathcal{R}_{\text{val}}, \mathcal{R}_{\text{sem}})$. Then $\mathbb{D}$ is fair iff $\mathcal{R}_{\text{sem}}$ is transitive.*

Relational databases When we deal with relational databases, the most natural valuation relation is $\mathcal{R}^{\text{rdb}}_{\text{val}}$ defined as follows:

$$(D, D') \in \mathcal{R}^{\text{rdb}}_{\text{val}} \quad \Leftrightarrow \quad D' = v(D) \text{ for some valuation } v.$$

So we assume, for now, that in relational semantics of incompleteness, the valuation relation is $\mathcal{R}_{\text{val}}^{\text{rdb}}$, and thus such semantics are defined by relation $\mathcal{R}_{\text{sem}}$. For OWA and CWA, these are particularly easy:

- For CWA, $\mathcal{R}_{\text{sem}}$ is the identity (i.e., $=$);
- For OWA, $\mathcal{R}_{\text{sem}}$ is the subset relation (i.e., $\subseteq$).

The special form of relation $\mathcal{R}_{\text{val}}^{\text{rdb}}$ implies the saturation property. Therefore we have:

Proposition 4. *For an arbitrary relational semantics given by relation $\mathcal{R}_{\text{sem}}$, and an arbitrary generic Boolean query Q, naïve evaluation works for Q iff Q is weakly monotone.*

Naïve evaluation via homomorphism preservation

We shall now relate weak monotonicity and preservation under homomorphisms (at least for relational semantics).

Consider relational databases over constants. Given two such databases D and D', a mapping h defined on the active domain $\text{adom}(D)$ of D is an $\mathcal{R}_{\text{sem}}$-*homomorphism* from D to D' if $(h(D), D') \in \mathcal{R}_{\text{sem}}$.

A query Q is *preserved* under $\mathcal{R}_{\text{sem}}$-homomorphisms if for every database D and every $\mathcal{R}_{\text{sem}}$-homomorphism h from D to D', if Q is true in D, then Q is true in D'.

Proposition 5. *If a relational semantics is given by a relation $\mathcal{R}_{\text{sem}}$ and Q is a generic Boolean query, then Q is weakly monotone iff it is preserved under $\mathcal{R}_{\text{sem}}$-homomorphisms.*

Putting everything together, we have our first key result for naïve evaluation over incomplete databases.

Theorem 2. *For a relational incompleteness semantics given by a semantic relation $\mathcal{R}_{\text{sem}}$, and a generic Boolean query Q, naïve evaluation works for Q iff Q is preserved under $\mathcal{R}_{\text{sem}}$-homomorphisms.*

Homomorphisms for relational semantics Theorem 2 connects naïve evaluation with homomorphism preservation. We now investigate what these $\mathcal{R}_{\text{sem}}$-homomorphisms are.

CWA **semantics** In this case $\mathcal{R}_{\text{sem}}$ is the identity, and the definition states that h is an $\mathcal{R}_{\text{sem}}$-homomorphism from D to D' if $D' = h(D)$. That is, under CWA, $\mathcal{R}_{\text{sem}}$-homomorphisms are the *strong onto* homomorphisms, i.e., homomorphisms from D to $h(D)$.

OWA **semantics** In this case $\mathcal{R}_{\text{sem}}$ is $\subseteq$, and the definition states that h is an $\mathcal{R}_{\text{sem}}$-homomorphism from D to D' if $h(D) \subseteq D'$. That is, under OWA, $\mathcal{R}_{\text{sem}}$-homomorphisms are just the usual homomorphisms.

Another well known notion of homomorphisms is that of *onto* homomorphisms. When used in the database context, an onto homomorphism h from D to D' is a homomorphism between D and D' so that $h(\text{adom}(D)) = \text{adom}(D')$. For instance, if $D = \{(1,2)\}$, and $h(1) = 3, h(2) = 4$, then h is a strong onto homomorphism from D to $D' = \{(3,4)\}$, and an onto homomorphism to $D'' = \{(3,4),(4,3)\}$. Note that while D'' contains more than $h(D)$, all the tuples in D'' only use elements that occur in $h(D)$.

A semantics of incompleteness that corresponds to this notion, that we refer to as *weak* CWA, or WCWA semantics, was actually previously studied [27] (in a slightly different, deductive-database context). We define it as follows:

$$[\![D]\!]_{\text{WCWA}} = \left\{ D' \;\middle|\; \begin{array}{l} D' \text{ is complete and} \\ \text{there is a valuation } h : D \to D' \\ \text{so that } \text{adom}(D') = \text{adom}(h(D)) \end{array} \right\}.$$

For this semantics, $\mathcal{R}_{\text{sem}}$ contains all pairs (D, D') so that $D \subseteq D'$ and $\text{adom}(D) = \text{adom}(D')$. That is, D can be expanded only within its active domain. Thus, $\mathcal{R}_{\text{sem}}$-homomorphisms are exactly onto homomorphisms.

For this relation $\mathcal{R}_{\text{sem}}$, the notion of preservation under $\mathcal{R}_{\text{sem}}$-homomorphisms is exactly the notion of preservation under onto homomorphisms. Thus, the WCWA semantics, defined long time ago, also corresponds to a very natural logical notion of preservation.

Note that $[\![D]\!]_{\text{CWA}} \subseteq [\![D]\!]_{\text{WCWA}} \subseteq [\![D]\!]_{\text{OWA}}$, and in general inclusions can be strict.

Naïve evaluation and relational semantics

We can finally state the equivalence of naïve evaluation and homomorphism preservation for three concrete semantics of incomplete relational databases:

Corollary 1. *Let Q be a Boolean generic query. Then:*

- *Under* OWA*, naïve evaluation works for Q iff Q is preserved under homomorphisms.*
- *Under* CWA*, naïve evaluation works for Q iff Q is preserved under strong onto homomorphisms.*
- *Under* WCWA*, naïve evaluation works for Q iff Q is preserved under onto homomorphisms.*

5. NAÏVE EVALUATION AND PRESERVATION FOR FO QUERIES

Corollary 1 reduces the problem of checking whether naïve evaluation works to preservation under homomorphisms. Thus, for FO queries, we deal with a very well known notion in logic [11]. However, what we need is preservation on *finite* structures, and those notions are well known to behave differently from their infinite counterpart. In fact, it was only proved recently by

Rossman that for FO sentences, preservation under arbitrary homomorphisms in the finite is equivalent to being an existential positive formula [30]. In database language, this means being a union of conjunctive queries, which led to an observation [24] that naïve evaluation works for a Boolean FO query Q iff Q is equivalent to a union of conjunctive queries.

The difficulty in establishing preservation results in the finite is due to losing access to classical logical tools such as compactness. Rossman's theorem, for instance, was a major open problem for many years. To make matters worse, even some existing infinite preservation results [21] have holes in their proofs.

Thus, it is unrealistic for a single paper to settle several very hard problems concerning preservation results in the finite (sometimes even without infinite analogs!). What we shall do instead is settle for classes of queries that *imply* preservation, and at the same time are easy to describe syntactically.

Positive and existential positive formulae Recall that the class Pos of *positive* formulae is defined inductively as follows:

- *true* and *false* are in Pos;
- every positive atomic formula (i.e., $R(\bar{x})$ or $x = y$) is in Pos;
- if $\varphi, \psi \in \mathsf{Pos}$, then $\varphi \vee \psi$ and $\varphi \wedge \psi$ are in Pos;
- if φ is in Pos, then $\exists x\varphi$ and $\forall x\psi$ are in Pos.

If only $\exists x\varphi$ remains in the class, we obtain the class $\exists\mathsf{Pos}$ of *existential positive formulae*. Formulae from $\exists\mathsf{Pos}$ are also known as unions of conjunctive queries.

Rossman's theorem [30] says that an FO sentence φ is preserved under homomorphisms over finite structures iff φ is equivalent to a sentence from $\exists\mathsf{Pos}$. Lyndon's theorem [11] says that an FO sentence φ is preserved under onto homomorphisms (over arbitrary structures) iff φ is equivalent to a sentence from Pos. Lyndon's theorem fails in the finite [4, 32] but the implication from being positive to preservation is still valid.

A characterization of preservation under strong onto homomorphisms was stated in [20, 21], but the syntactic class had a rather messy definition and was limited to a single binary relation. Even worse, we discovered a gap in one of the key lemmas in [21]. So instead we propose a simple extension of positive formulae that gives preservation under strong onto homomorphisms.

Extensions with universal guards The fragment $\mathsf{Pos} + \forall\mathsf{G}$, whose definition is inspired by [12], extends Pos with universal guards. It is defined as a fragment closed under all the formation rules for Pos and, in addition, the following rule:

- for a $\mathsf{Pos} + \forall\mathsf{G}$ formula φ, a tuple of distinct variables $\bar{x}$, and a relation symbol R (possibly the equality relation), the formula $\forall\bar{x}\big(R(\bar{x}) \rightarrow \varphi\big)$ is in $\mathsf{Pos} + \forall\mathsf{G}$.

Clearly we have $\exists\mathsf{Pos} \subsetneq \mathsf{Pos} \subsetneq \mathsf{Pos} + \forall\mathsf{G}$.

Proposition 6. *Sentences in* $\mathsf{Pos} + \forall\mathsf{G}$ *are preserved under strong onto homomorphisms.*

We now combine all the previous implications (preservation → monotonicity → naïve evaluation) to show that naïve evaluation can work beyond unions of conjunctive queries under realistic semantic assumptions.

Theorem 3. *Let Q be a Boolean FO query. Then:*

- *If Q is in $\exists\mathsf{Pos}$, then naïve evaluation works for Q under* OWA.
- *If Q is in Pos, then naïve evaluation works for Q under* WCWA.
- *If Q is in $\mathsf{Pos} + \forall\mathsf{G}$, then naïve evaluation works for Q under* CWA.

Contrast this with the result of [24] saying that under OWA, the first statement is 'if and only if', i.e., one cannot go beyond $\exists\mathsf{Pos}$. Now we see that, under other semantics, one can indeed go well beyond that class, essentially limiting only unrestricted negation, and still use naïve evaluation.

One immediate question is what happens with non-Boolean queries. There is a simple answer: *all results extend to non-Boolean queries*. We explain how this is done in Section 8, once we have looked at other semantics (as the lifting will apply to all of them).

6. SEMANTIC ORDERINGS

In this section we study semantic orderings arising from the usual relational semantics of incompleteness. We recall known results about the study of such orderings in the context of Codd databases [9, 23, 26, 31]. Such results are of two kinds: they connect orderings based on incompleteness with well-known orderings from the field of programming semantics, and they describe those via elementary *updates* that increase the information content of an instance.

Codd databases

SQL uses a single value `null` for missing information. As comparisons of a null with other values in SQL do not evaluate to true (technically, they evaluate to *unknown*, as SQL uses three-valued logic), this is properly modeled by a special kind of naïve databases, called *Codd databases*, in which nulls do not repeat.

For tuples $t = (a_1, \ldots, a_n)$ and $t' = (a'_1, \ldots, a'_n)$ over $\mathsf{Const} \cup \mathsf{Null}$ in which nulls do not repeat, we write $t \sqsubseteq t'$ if $a_i \in \mathsf{Const}$ implies $a'_i = a_i$. The meaning is that t'

is at least as informative as t. There are two standard ways of lifting $\sqsubseteq$ to sets:

$$D \sqsubseteq^{\mathrm{H}} D' \;\Leftrightarrow\; \forall t \in D\ \exists t' \in D' : t \sqsubseteq t'$$
$$D \sqsubseteq^{\mathrm{P}} D' \;\Leftrightarrow\; \forall t' \in D'\ \exists t \in D : t \sqsubseteq t' \text{ and } D \sqsubseteq^{\mathrm{H}} D'$$

Superscripts H and P stand for Hoare and Plotkin, who first studied these orderings in the context of the semantics of concurrent processes, cf. [16].

These had been previously accepted as the correct orderings to represent the OWA and the CWA semantics over Codd databases [9, 23, 26, 31]. This can be justified by considering updates that affect informativeness of incomplete databases. Consider, for example, two tuples $(1, 2)$ and $(2, 2)$, and assume that we somehow lose the value of the first attribute. SQL has a unique null value, so both tuples become $(\mathtt{null}, 2)$, which thus must represent the instance $\{(1, 2), (2, 2)\}$ even under CWA, since no tuples were lost, only individual values. Alternatively, one can view this as an *allowed update*, under CWA, from $(\mathtt{null}, 2)$, that produces a more informative instance $\{(1, 2), (2, 2)\}$ by replacing the null twice. In the case of OWA, one can have updates that add arbitrary new tuples.

Let D be a database, R a relation in it, t a tuple, and i a position in that tuple that contains a null $\bot$. Then by $D[v/R(t.i)]$ we mean D in which that occurrence of $\bot$ is replaced by $v \in \mathsf{Const} \cup \mathsf{Null}$, and by $D^+[v/R(t.i)]$ we mean D to which a tuple obtained from t by replacing the occurrence of $\bot$ in the ith position with v is added (i.e., the original t is retained). Now we consider updates $D \rightarrowtail^{\mathrm{codd}} D'$ of two kinds:

- Codd CWA updates: $D \rightarrowtail^{\mathrm{codd}}_{\mathrm{CWA}} D[v/R(t.i)]$ and and $D \rightarrowtail^{\mathrm{codd}}_{\mathrm{CWA}} D^+[v/R(t.i)]$;
- OWA update: $D \rightarrowtail^{\mathrm{codd}}_{\mathrm{OWA}} D \cup R(t)$ that adds a tuple to a relation in a database.

It is known [23] that the reflexive-transitive closure

- of $\rightarrowtail^{\mathrm{codd}}_{\mathrm{CWA}} \cup \rightarrowtail^{\mathrm{codd}}_{\mathrm{OWA}}$ is exactly $\sqsubseteq^{\mathrm{H}}$; and
- of $\rightarrowtail^{\mathrm{codd}}_{\mathrm{CWA}}$ is exactly $\sqsubseteq^{\mathrm{P}}$,

over Codd databases. Our next goal is to describe orderings corresponding to OWA and CWA for naïve databases, and to give an update semantics for them.

Naïve databases

Firstly we describe the semantic orderings $\preceq_*$ given by the semantics $[\![\,]\!]_*$, where $*$ is OWA, CWA, or WCWA. They are characterized via database homomorphisms as follows (the first item was already shown in [24]).

Proposition 7. *$D \preceq_{\mathrm{OWA}} D'$ (respectively $D \preceq_{\mathrm{CWA}} D'$ or $D \preceq_{\mathrm{WCWA}} D'$) iff there is a database homomorphism (respectively, strong onto, or onto database homomorphism) from D to D'.*

Next, we provide update justification for these orderings. OWA updates just add tuples as before; we denote them by $\rightarrowtail_{\mathrm{OWA}}$. CWA updates are different, to account for repetition of nulls. In particular, once a null is replaced by some value v, *all* its occurrences must be replaced. Formally, if $\bot$ is a null that occurs in D, then $D[v/\bot]$ is D in which $v \in \mathsf{Const} \cup \mathsf{Null}$ replaces $\bot$ everywhere. The CWA update is now an update $D \rightarrowtail_{\mathrm{CWA}} D[v/\bot]$.

Let * stand for the transitive-reflexive closure of a relation (i.e., a sequence of updates). Then we have:

Theorem 4. • $\rightarrowtail^*_{\mathrm{CWA}} = \preceq_{\mathrm{CWA}}$;
• $(\rightarrowtail_{\mathrm{CWA}} \cup \rightarrowtail_{\mathrm{OWA}})^* = \preceq_{\mathrm{OWA}}$.

In other words, D is less informative than D' iff D' is obtained from D by a sequence of

- CWA updates, under CWA;
- CWA and OWA updates, under OWA.

What are the orderings $\preceq_{\mathrm{OWA}}$ and $\preceq_{\mathrm{CWA}}$ when we restrict them to Codd databases? One would expect them to be $\sqsubseteq^{\mathrm{H}}$ and $\sqsubseteq^{\mathrm{P}}$, corresponding to OWA and CWA for the Codd semantics, but this is only partly true. In fact, [24] proved that over Codd databases,

- $\preceq_{\mathrm{OWA}}$ and $\sqsubseteq^{\mathrm{H}}$ coincide;
- $D \preceq_{\mathrm{CWA}} D'$ iff $D \sqsubseteq^{\mathrm{P}} D'$ and relation $\sqsubseteq$ has a perfect matching from D' to D.

So this leads to a question: is there is a "natural" semantic ordering over naïve databases that, when restricted to Codd databases, coincides precisely with $\sqsubseteq^{\mathrm{P}}$? In the next section, we present such an ordering, and show that it gives rise to a whole new family of semantics of incompleteness.

7. POWERSET SEMANTICS

Our search for the answer to the question at the end of the previous section leads us to consider a new class of semantics of incompleteness, in which not one, but several valuations can be applied to nulls. In other words, we produce several valuations (hence the name *powerset semantics*), and then combine them into a single one. Notationally, we distinguish them by using $(\!|\ |\!)$ brackets.

We start with a semantics defined as follows:

$$(\!|D|\!)_{\mathrm{CWA}} = \left\{ h_1(D) \cup \ldots \cup h_n(D) \;\middle|\; \begin{array}{l} h_1, \ldots, h_n \text{ are} \\ \text{valuations}, n \geqslant 1 \end{array} \right\}.$$

That is, $D' \in (\!|D|\!)_{\mathrm{CWA}}$ iff there exists a set of valuations $h_1, \ldots, h_n$ on D so that $D' = \bigcup\{h_i(D) \mid 1 \leqslant i \leqslant n\}$. We call it the CWA powerset semantic.

Next, we describe the ordering $\mathbb{E}_{\mathrm{CWA}}$ induced by this semantics: that is, $D \mathbb{E}_{\mathrm{CWA}} D'$ iff $(\!|D'|\!)_{\mathrm{CWA}} \subseteq (\!|D|\!)_{\mathrm{CWA}}$).

To updates used as the justification of orderings in the previous section, we now add a new type. A *copying* CWA *update* is of the form

$$D \rightarrowtail\!\!\!\!\!\rightarrow_{\text{CWA}} D[v/\bot] \cup D^{\text{fresh}},$$

where D^{fresh} is a copy of D in which all nulls are replaced by fresh ones. This is a relaxation of CWA: we can add tuples in an update, but only in a very limited way, if they mimic the original database.

It turns out that the ordering $\mathbb{E}_{\text{CWA}}$ can be seen as a sequence of regular and copying CWA updates, and that when restricted to Codd databases, it coincides precisely with $\sqsubseteq^{\text{P}}$. That is, we have the following.

Theorem 5. • *$D \mathbb{E}_{\text{CWA}} D'$ iff there exists a set of database homomorphisms $h_1, \ldots, h_n$ defined on D so that $D' = \bigcup\{h_i(D) \mid 1 \leqslant i \leqslant n\}$.*

- $(\rightarrowtail_{\text{CWA}} \cup \rightarrowtail\!\!\!\!\!\rightarrow_{\text{CWA}})^* = \mathbb{E}_{\text{CWA}}$.
- *Over Codd databases, $\mathbb{E}_{\text{CWA}}$ and $\sqsubseteq^{\text{P}}$ coincide.*

Preservation for powerset semantics

Our next goal is to understand how we can make naïve evaluation work under the powerset semantics. For the standard semantics of incompleteness, we related naïve evaluation to preservation of queries under homomorphisms. We shall do the same here, but the setting for homomorphisms will be a bit different.

Recall that before we looked at relational semantics defined by two relations, relation $\mathcal{R}^{\text{rdb}}_{\text{val}} = \{(D, v(D)) \mid v \text{ is a valuation}\}$ and relation $\mathcal{R}_{\text{sem}}$ between complete databases. We now replace $\mathcal{R}^{\text{rdb}}_{\text{val}}$ with

$$\boldsymbol{\mathcal{R}}^{\text{rdb}}_{\text{val}} = \{(D, \{v_1(D), \ldots, v_n(D)\}) \mid v_i\text{'s are valuations}\},$$

and consider relations $\boldsymbol{\mathcal{R}}_{\text{sem}}$ between a finite set of complete databases and a single complete database. We require that $\boldsymbol{\mathcal{R}}_{\text{sem}}$ be total and contain pairs $(\{c\}, c)$ for all complete objects c. An example is the relation $\boldsymbol{\mathcal{R}}_{\cup} = \{(\mathcal{X}, X) \mid X = \bigcup \mathcal{X}\}$, corresponding to taking the union of databases.

A powerset semantics $(\!|\ |\!)$ is *given by relation* $\boldsymbol{\mathcal{R}}_{\text{sem}}$ if

$$D' \in (\!|D|\!) \;\Leftrightarrow\; (D, D') \in \boldsymbol{\mathcal{R}}^{\text{rdb}}_{\text{val}} \circ \boldsymbol{\mathcal{R}}_{\text{sem}}.$$

For instance, the semantics $(\!|\ |\!)_{\text{CWA}}$ is given by the relation $\boldsymbol{\mathcal{R}}_{\cup}$.

Consider complete relational databases D and D'. An $\boldsymbol{\mathcal{R}}_{\text{sem}}$*-homomorphism* between D and D' is a set $\{h_1, \ldots, h_n\}$ of mappings defined on $\text{adom}(D)$ so that $\{h_1(D), \ldots, h_n(D)\}\boldsymbol{\mathcal{R}}_{\text{sem}}D'$. Note that if $n = 1$, this is exactly the notion of $\mathcal{R}_{\text{sem}}$-homomorphisms seen earlier. The connection between naïve evaluation and homomorphism preservation now extends to powerset semantics.

Proposition 8. *For every powerset semantics given by a relation $\boldsymbol{\mathcal{R}}_{\text{sem}}$, naïve evaluation works for a generic Boolean query Q iff Q is preserved under $\boldsymbol{\mathcal{R}}_{\text{sem}}$-homomorphisms.*

Let us now look at the semantics $(\!|\ |\!)_{\text{CWA}}$ given by relation $\boldsymbol{\mathcal{R}}_{\cup}$. The notion of preservation under $\boldsymbol{\mathcal{R}}_{\cup}$-homomorphisms is preservation *under union of strong onto homomorphisms*: if Q is true in D, and $h_1, \ldots, h_n$ are homomorphisms defined on D, then Q is true in $h_1(D) \cup \ldots \cup h_n(D)$.

For previous preservation results among FO queries, we looked at classes Pos and $\exists\mathsf{Pos}$ of positive and existential positive queries, and the class $\mathsf{Pos}{+}\forall\mathsf{G}$ of positive queries with universal guards. Now let $\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$ be the class of existential positive queries extended with *Boolean universal guards*, i.e., universally guarded formulae which are sentences. More precisely, if $\bar{x}$ is a tuple of distinct variables, $\varphi(\bar{y})$ is a formula in $\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$, where all $\bar{y}$ variables are contained in $\bar{x}$, and R is a relation symbol (possibly the equality relation), then $\forall\bar{x}\ (R(\bar{x}) \rightarrow \varphi(\bar{y}))$ is in $\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$.

Lemma 1. *Sentences in $\exists\mathsf{Pos}{+}\forall\mathsf{G}^{\text{bool}}$ are preserved under unions of strong onto homomorphisms.*

Combining, we get the following result.

Corollary 2. *If Q is a Boolean query from the class $\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$, then naïve evaluation works for Q under the $(\!|\ |\!)_{\text{CWA}}$ semantics.*

Semantics similar to $(\!|\ |\!)_{\text{CWA}}$ did appear in the literature. In fact, the closest comes from the study of CWA in the context of data exchange [5]. It was presented in [18] (and based in turn on a semantics from [25]), and essentially boils down to the $(\!|\ |\!)_{\text{CWA}}$ semantics, but based on a restricted notion of valuations, namely minimal valuations. We shall study those in Section 9.

8. LIFTING TO NON-BOOLEAN QUERIES

So far our results dealt with Boolean queries. Now we show how to lift them to the setting of arbitrary k-ary relational queries. The basic idea is to consider database domains where objects are pairs consisting of a database and a k-tuple of constants. This turns queries into Boolean, and we apply our results. This requires more technical development than seems to be implied by the simple idea, but it can be carried out for all the semantics. We sketch now how the extension works.

A k-ary query Q maps a database D to a subset of $\text{adom}(D)^k$. It is *generic* if, for each one-to-one map f : $\text{adom}(D) \rightarrow \mathsf{Const} \cup \mathsf{Null}$, we have $Q(f(D)) = f(Q(D))$.

Given a semantics $[\![\ ]\!]$, certain answers to Q are defined as $\mathsf{certain}(Q, D) = \bigcap\{Q(D') \mid D' \in [\![D]\!]\}$. Naïve evaluation *works* for Q if $\mathsf{certain}(Q, D)$ is precisely the set of tuples in $Q(D)$ that do not have nulls. We refer to this set (i.e., $Q(D) \cap \mathsf{Const}^k$) as $Q^{\mathsf{C}}(D)$.

As before, Q is monotone if $D \preceq D'$ implies $Q^{\mathsf{C}}(D) \subseteq Q^{\mathsf{C}}(D')$ for the semantic ordering $\preceq$, and Q is weakly monotone if the above is true whenever $D' \in [\![D]\!]$.

We will need a stronger form of saturation property. A relational database domain is *strongly saturated* if every database has "sufficiently" many complete instances in its semantics that are isomorphic to it. More precisely, for each database D, and each finite set $C \subset \mathsf{Const}$, there is an isomorphic instance $D' \in [\![D]\!]$ such that both the isomorphism from D to D' and its inverse are the identity on C.

If we deal, as before, with relational semantics given by pairs $\mathcal{R} = (\mathcal{R}^{\text{rdb}}_{\text{val}}, \mathcal{R}_{\text{sem}})$, we say that a k-ary query is *weakly preserved* under a class of $\mathcal{R}_{\text{sem}}$-homomorphisms if for every database D, a k-tuple t of constants, and an $\mathcal{R}_{\text{sem}}$-homomorphism $h : D \to D'$ from the class that is the identity on t, the condition $t \in Q(D)$ implies $t \in Q(D')$. Note that for Boolean queries this is the same as preservation under $\mathcal{R}_{\text{sem}}$-homomorphisms.

Then the main connections continue to hold.

Lemma 2. *Let $\mathbb{D}$ be a relational database domain with the strong saturation property, and Q a k-ary generic query. Then the following are equivalent:*

1. *naïve evaluation works for Q;*
2. *Q is weakly monotone; and*
3. *(if the semantics is given by a relation $\mathcal{R}_{\text{sem}}$): Q is weakly preserved under $\mathcal{R}_{\text{sem}}$-homomorphisms.*

One can then check that for all the classes of FO formulae considered here, preservation results hold when extended to formulae with free variables. In addition, one can develop similar transfer technique for powerset semantics and conclude that all the results remain true for non-Boolean queries.

Theorem 6. *Let Q be a k-ary FO query, $k \geqslant 0$. Then:*

- *If Q is in $\exists\mathsf{Pos}$, then naïve evaluation works for Q under* OWA.
- *If Q is in Pos, then naïve evaluation works for Q under* WCWA.
- *If Q is in $\mathsf{Pos} + \forall\mathsf{G}$, then naïve evaluation works for Q under* CWA.
- *If Q is in $\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$, then naïve evaluation works for Q under $(\!|\ |\!)_{\text{CWA}}$.*

9. MINIMAL VALUATIONS SEMANTICS

So far all the semantics that we saw allowed arbitrary valuations to be applied to instances with nulls. These are not the only possible semantics. In fact [18], based on earlier work in the area of logic programming [25], proposed a powerset semantics that is based on *minimal* valuations. We now introduce it in our context (as [18] defined it in the context of data exchange).

For now we deal with database homomorphisms, i.e., $h(c) = c$ for each $c \in \mathsf{Const}$. We say that a homomorphism h defined on an instance D is *D-minimal* if no proper subinstance of $h(D)$ is a homomorphic image of D; equivalently, there is no other homomorphism h' so that $h'(D) \subsetneq h(D)$. If h is a valuation, then we talk about a D-minimal valuation.

Not every valuation (or homomorphism) is minimal. Consider an incomplete table $D = \{(\bot, \bot), (\bot, \bot')\}$ and a valuation $v(\bot) = 1$, $v(\bot') = 2$. This is not minimal: take for instance $v'(\bot) = v'(\bot') = 1$ and we have $v'(D) \subsetneq v(D)$. The valuation v' is minimal.

The semantics of [18] is defined as

$$(\!|D|\!)^{\min}_{\text{CWA}} = \left\{ \bigcup_{h \in \mathcal{H}} h(D) \;\middle|\; \begin{array}{l} \mathcal{H} \text{ is a nonempty set of} \\ D\text{-minimal valuations.} \end{array} \right\}.$$

This is a powerset-based semantics, and the semantic relation it uses is the union relation $\boldsymbol{\mathcal{R}}_{\cup}$, the same as in Section 7. However the valuation relation is no longer $\boldsymbol{\mathcal{R}}^{\text{rdb}}_{\text{val}}$, allowing all valuations, but rather $\boldsymbol{\mathcal{R}}^{\min}_{\text{val}}$ containing all pairs $(D, \{h(D) \mid h \in \mathcal{H}\})$ with $\mathcal{H}$ ranging over nonempty sets of D-minimal valuations.

The fact that we no longer allow all valuations makes the equivalence of naïve evaluation and preservation of $\boldsymbol{\mathcal{R}}_{\text{sem}}$-homomorphisms invalid (we shall see an example soon). The main reason is that the saturation property does not longer hold, and therefore Theorem 1 is no longer applicable. The solution to the problem lies in establishing connections between minimal homomorphisms, naïve evaluations, and *cores* of database instances, which we do next.

Minimal homomorphisms and cores

Recall that a *core* of a structure D (in our case, a relational database of vocabulary σ) is a substructure $D' \subseteq D$ such that D' is a homomorphic image of D but no proper subinstance of D' is. In other words, there is a homomorphism $h : D \to D'$ but there is no homomorphism $g : D \to D''$ for $D'' \subsetneq D'$. It is known that a core is unique up to isomorphism, so we can talk of *the* core of D, and denote it by $\text{core}(D)$. A structure is called a core if $D = \text{core}(D)$. The cores are commonly used over graphs [17]; here we use them with the database notion of homomorphism that preserves constants (for which all results about cores remain true [14]).

Even if minimal homomorphisms are related to cores, their images cannot be described precisely in terms of cores, as shown next. We strengthen results given in several examples in [18] (where constants were used in an essential way):

Proposition 9. *If h is D-minimal, then $h(D)$ is a core and $h(D) = h(\text{core}(D))$. However, there is a core D and a homomorphism h defined on it so that $h(D)$ is a core, but h is not D-minimal. This also holds if both D and $h(D)$ contain only nulls, and if D is a graph.*

This also shows that $[\![D]\!]^{\min}_{\text{CWA}}$ need not be the same as $[\![\text{core}(D)]\!]_{\text{CWA}}$. Nevertheless, cores do play a crucial

role in the study of minimal semantics. Recall that a generic Boolean query Q is weakly monotone under $(\!|\,|\!)^{\min}_{\text{CWA}}$ if $Q(D) = 1$ and $D' \in (\!|D|\!)^{\min}_{\text{CWA}}$ imply $Q(D') = 1$.

Theorem 7. *Let Q be a generic Boolean query. Then naïve evaluation works for Q under $(\!|\,|\!)^{\min}_{\text{CWA}}$ iff*

1. *Q is weakly monotone under $(\!|\,|\!)^{\min}_{\text{CWA}}$, and*
2. *$Q(D) = Q(\text{core}(D))$ for every D.*

Hence, the crucial new condition for naïve evaluation under minimal semantics is that Q cannot distinguish a database from its core.

Preservation and naïve evaluation We now relate weak monotonicity to homomorphism preservation. For this, we consider minimality for instances D over Const. For such an instance, and a homomorphism h defined on D, we let $\text{fix}(h, D) = \{c \in \mathsf{Const}(D) \mid h(c) = c\}$. Then h is called D-minimal if there is no homomorphism g with $\text{fix}(h, D) \subseteq \text{fix}(g, D)$ and $g(D) \subsetneq h(D)$. Note that database homomorphisms fix precisely the set of constants in D, so the first condition was not necessary.

Given a Boolean query Q, we say that it is *preserved under unions of minimal homomorphisms* if, whenever D is a database over Const and $\mathcal{H}$ is a nonempty set of D-minimal homomorphisms such that $\text{fix}(h, D) = \text{fix}(g, D)$ whenever $f, g \in \mathcal{H}$, we have that $Q(D) = 1$ implies $Q(\bigcup\{h(D) \mid h \in \mathcal{H}\}) = 1$.

Proposition 10. *Let Q be a generic Boolean query. Then it is weakly monotone under $(\!|\,|\!)^{\min}_{\text{CWA}}$ iff it is preserved under unions of minimal homomorphisms.*

From this, we can derive the following result. We say that naïve evaluation works for Q over D if the certain answer to Q over D coincides with $Q(D)$.

Theorem 8. *Let Q be a Boolean query from $\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$. Then, under the $(\!|\,|\!)^{\min}_{\text{CWA}}$:*

- *naïve evaluation works for Q over D if D is a core; and*
- *naïve evaluation works for Q iff $Q(D) = Q(\text{core}(D))$ for all D.*

Note that the second statement of the theorem is an immediate corollary of Proposition 10 and the fact that minimal homomorphisms are homomorphisms.

The preconditions that D be a core, or that $Q(D) = Q(\text{core}(D))$, are essential. To see this, consider an incomplete instance $D = \{(\bot, \bot), (\bot, \bot')\}$. Every D-minimal valuation h must satisfy $h(\bot) = h(\bot')$, i.e., their images are precisely the instances $\{(c, c)\}$ for $c \in \mathsf{Const}$. Hence, under $(\!|\,|\!)^{\min}_{\text{CWA}}$, the certain answer to $\forall x, y\ (D(x, y) \to x = y)$ is true, while evaluating this formula on D produces false. The reason naïve evaluation does not return certain answers (although the formula is in $\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$) is that $Q(D) \neq Q(\text{core}(D))$, since $\text{core}(D) = \{(\bot, \bot)\}$.

Non-boolean queries As before, results shown here extend to non-Boolean queries. Specifically, we can show the following, using the notion of weak preservation and techniques of Section 8.

Proposition 11. *Given a generic k-ary query Q, naïve evaluation works for Q under $(\!|\,|\!)^{\min}_{\text{CWA}}$ iff Q is weakly preserved under unions of minimal homomorphisms and $Q^{\mathsf{C}}(D) = Q^{\mathsf{C}}(core(D))$, for each database D.*

In particular, if Q is an $\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$ query, then naïve evaluation works for Q over cores; and furthermore, naïve evaluation works for Q over all databases iff $Q^{\mathsf{C}}(D) = Q^{\mathsf{C}}(core(D))$, for every D.

10. FUTURE WORK

We now present the main directions in which we would like to extend this work.

Other data models So far we looked at either a very general setting, which can subsume practically every data model, or at relational databases. We would like to extend our results to XML. At this time, we have a good understanding of the semantics of incomplete XML documents and the complexity of answering queries over them [3, 8, 15] that can serve as a good starting point.

Other languages When we dealt with relations, we studied FO as the main query language. However, our structural results are in no way limited to FO. In fact it is known that naïve evaluation works for datalog (without negation). Given the toolkit of this paper, we would like to consider queries in languages that go beyond FO and admit naïve evaluation.

Preservation results There are open questions related to preservation results in both finite and infinite model theory. We already mentioned that the results of [21] about preservation under strong onto homomorphisms are limited to a simple vocabulary, and even then appear to be problematic. We would like to establish a precise characterization in the infinite case, and see whether it holds or fails in the finite. We also want to look at preservation on restricted classes of structures, following [7] which looked at bounded treewidth (but does not capture XML with data).

The impact of constraints Constraints (e.g., keys and foreign keys) have a huge impact on the complexity of finding certain answers [10, 33], so it is thus natural to ask how they affect good classes we described in this paper. Constraints appear in another model of incompleteness – conditional tables [19] – that in general have higher complexity of query evaluation [2] but are nonetheless useful in several applications [6].

Applications In applications such as data integration and exchange, finding certain answers is the standard query answering semantics [5, 22]. In fact one of our semantics came from data exchange literature [18].

Semantics	symbol	Naïve evaluation works for
open world	$[\![\,]\!]_{\text{OWA}}$	$\exists\mathsf{Pos}$ = unions of CQs
weak closed-world	$[\![\,]\!]_{\text{WCWA}}$	Pos
closed world:	$[\![\,]\!]_{\text{CWA}}$	$\mathsf{Pos} + \forall\mathsf{G}$
powerset closed-world	$(\!|\,|\!)_{\text{CWA}}$	$\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$
minimal, powerset closed-world	$(\!|\,|\!)^{\text{min}}_{\text{CWA}}$	$\exists\mathsf{Pos} + \forall\mathsf{G}^{\text{bool}}$, over cores

Figure 1: Summary of naïve evaluation results for FO queries

We would like to see whether our techniques help find classes of queries for which query answering becomes easy in exchange and integration scenarios.

Minimal semantics: why cores? What makes cores so special for minimal semantics? Can results of Section 9 be extended to other types of semantics, with different constructions playing the role of cores? And are there other natural semantics based on the notion of minimality?

Bringing back the infinite We have used a number of results from infinite model theory to get our syntactic classes. Another way of appealing to logic over infinite structures to handle incompleteness was advocated by Reiter [27, 29] three decades ago. In that approach, an incomplete database D is viewed as a logical theory T_D, and finding certain answers to Q amounts to checking whether T_D entails Q. This is in general an undecidable problem, and entailment in the finite is known to be more problematic than unrestricted one. This is reminiscent of the situation with homomorphism preservation results, but we saw that we can use infinite results to obtain useful sufficient conditions. Motivated by this, we would like to revisit Reiter's proof-theoretic approach and connect it with our semantic approach.

Acknowledgments Work partly supported by EPSRC grants G049165 and J015377.

11. References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] S. Abiteboul, P Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. *TCS*, 78(1):158–187, 1991.

[3] S. Abiteboul, L. Segoufin, and V. Vianu. Representing and querying XML with incomplete information. *ACM TODS*, 31(1):208–254, 2006.

[4] M. Ajtai, Y. Gurevich. Monotone versus positive. *J. ACM* 34(4):1004-1015 (1987).

[5] M. Arenas, P. Barceló, L. Libkin, F. Murlak. *Relational and XML Data Exchange*. Morgan & Claypool, 2010.

[6] M. Arenas, J. Pérez, J. Reutter. Data exchange beyond complete data. In *PODS'11*, pages 83–94.

[7] A. Atserias, A. Dawar, P. Kolaitis. On preservation under homomorphisms and unions of conjunctive queries. *J. ACM* 53(2):208-237 (2006).

[8] P. Barceló, L. Libkin, A. Poggi, and C. Sirangelo. XML with incomplete information. *J. ACM* 58(1): 1–62 (2010).

[9] P. Buneman, A. Jung, A. Ohori. Using powerdomains to generalize relational databases. *TCS* 91 (1991), 23–55.

[10] A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *PODS'03*, pages 260–271.

[11] C.C. Chang and H.J. Keisler. *Model Theory*. North Holland, 1990.

[12] K. Compton. Some useful preservation theorems. *J. Symb. Logic* 48 (1983), 427–440.

[13] C. Date and H. Darwin. *A Guide to the SQL Standard*. Addison-Wesley, 1996.

[14] R. Fagin, P. Kolaitis, L. Popa. Data exchange: getting to the core. *ACM TODS* 30(1):174-210 (2005).

[15] A. Gheerbrant, L. Libkin, T. Tan. On the complexity of query answering over incomplete XML documents. *ICDT'12*, pages 169-181.

[16] C. Gunter. *Semantics of Programming Languages*. The MIT Press, 1992.

[17] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.

[18] A. Hernich. Answering non-monotonic queries in relational data exchange. *LMCS* 7(3) (2011).

[19] T. Imieliński and W. Lipski. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[20] H. J. Keisler. Finite approximations of infinitely long formulas. *Symp. Theory of Models*, 1965, pages 158–169.

[21] H. J. Keisler. Some applications of infinitely long formulas. *Journal of Symbolic Logic*, 1965, pages 339–349.

[22] M. Lenzerini. Data integration: a theoretical perspective. In *PODS'02*, pages 233–246.

[23] L. Libkin. A semantics-based approach to design of query languages for partial information. In *Semantics in Databases*, LNCS 1358, 1998, pages 170–208.

[24] L. Libkin. Incomplete information and certain answers in general data models. *PODS'11*, pages 59–70.

[25] J. Minker. On indefinite databases and the closed world assumption. In *CADE'82*, pages 292–308.

[26] A. Ohori. Semantics of types for database objects. *Theoretical Computer Science* 76 (1990), 53–91.

[27] R. Reiter. On closed world data bases. In *Logic and Data Bases*, 1977, pages 55–76.

[28] R. Reiter. Equality and domain closure in first-order databases. *J. ACM* 27(2): 235-249 (1980).

[29] R. Reiter. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling*, 1982, pages 191–233.

[30] B. Rossman. Homomorphism preservation theorems. *J. ACM* 55(3): (2008).

[31] B. Rounds. Situation-theoretic aspects of databases. In *Situation Theory and Appl.*, CSLI vol. 26, 1991, pages 229-256.

[32] A. Stolboushkin. Finitely monotone properties. In *LICS'95*, pages 324–330.

[33] M. Vardi. On the integrity of databases with incomplete information. In *PODS'86*, pages 252–266.