Characterizing certain answers via compactness

Christophe Rey, Elias Tahhan-Bittar, Jerzy Tomasik

Characterizing certain answers via compactness
Caractérisation des réponses certaines par compacité

Christophe Rey
Université Clermont Auvergne
LIMOS, IUT d'Allier, CNRS
Clermont-Ferrand, France
christophe.rey@uca.fr

Elias Tahhan-Bittar
Universidad Simon Bolivar
Caracas, Venezuela
etahhan@usb.ve

Jerzy Tomasik
Université Clermont Auvergne
LIMOS, CNRS
Clermont-Ferrand, France
jerzy.tomasik@udamail.fr

ABSTRACT

In relational data integration, some approaches rely on logic programming to get answers for one query from several sources. More precisely, in the mediation setting where a virtual global schema linked to all local source schemas is assumed, some algorithms are based on SLD-resolution to generate rewritings of a query expressed on the global schema, so that it can be evaluated on the many sources. The aim of this evaluation is to obtain the certain answers [1]. These answers are the answers that are always obtained when a user evaluates his query on any materialization of the global schema which reflects the source data [9]. Of course, it has been shown that the generated rewritings, especially via SLD-resolution, could compute the certain answers. However, from a more general point of view, and up to our knowledge, no certain answers characterization w.r.t. answers of a definite program formalizing a mediation setting has been proposed yet. This is the main contribution of this paper, which is based on a new compactness result that characterizes certain answers w.r.t. an extension to infinite models of certain answers.

RÉSUMÉ

En intégration de données relationnelles, certaines approches font usage de la programmation logique pour obtenir des réponses à une requête posée sur plusieurs sources. Plus précisément, dans les contextes de médiation, où l'on suppose l'existence d'un schéma global virtuel reliant tous les schémas des sources locales, certains algorithmes sont basés sur la SLD-resolution pour générer des réécritures des requêtes exprimées sur le schéma global, et ce afin de pouvoir les évaluer sur les sources locales. Le but de cette évaluation est d'obtenir les réponses certaines [1]. Ces réponses sont celles qui seraient toujours obtenues si l'utilisateur pouvait évaluer sa requête sur toutes les matérialisations possibles du schéma global en accord avec les données contenues dans les sources [9]. Bien entendu, il a été démontré que les réécritures générées, notamment par SLD-resolution, permettaient d'obtenir les réponses certaines. Cependant, dans une perspective plus générale, il n'existe pas encore, à notre connaissance, de caractérisation des réponses certaines en fonction des réponses d'un programme défini formalisant un contexte de médiation. C'est la contribution principale de cet article, qui s'appuie sur un résultat nouveau de compacité caractérisant les réponses certaines en fonction d'une extension aux modèles infinis de ces dernières.

CCS CONCEPTS

• **Information systems → Mediators and data integration**; *Relational database model*;

KEYWORDS

certain answers ; compactness ; definite program ; local-as-view ; mediation ; data integration

ACM Reference format:
Christophe Rey, Elias Tahhan-Bittar, and Jerzy Tomasik. 2017. Characterizing certain answers via compactness Caractérisation des réponses certaines par compacité. In *Proceedings of Unpublished, , November 2017,* 14 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The query answering problem in relational data integration is the problem of getting answers to a single user query from many heterogeneous relational data sources without forcing the user to formulate herself her query on each source schema. In this context, the mediation setting assumes a virtual global schema to which the user can express her query, and then studies how to automatically get answers from this single query. A well-studied algorithmical approach to solve this is query rewriting: the user query is rewritten into as many formulas as there are sources to query. The union of all answers to all these rewritings evaluated on the corresponding sources is the answer to the initial query. The set of obtained answers must be the set of certain answers [1]: computing these is the aim of the query answering problem, in data integration systems that follow the mediation setting.

Previously, it has been shown that the generated rewritings, especially via SLD-resolution, could compute the certain answers. However, from a more general point of view, and up to our knowledge, no certain answers characterization w.r.t. answers of a definite program formalizing a mediation setting has been proposed yet. In this paper, we propose such a characterization. In other words, we show the completeness of SLD-resolution for answering conjunctive queries over local-as-view (LAV) mappings. And this result is obtained through a compactness result that says that certain answers do not change when evaluated on infinite or finite models.

Such results can appear to be well-known. Despite that they implicitly arise from many previous works [1, 4, 9, 12, 14], we

argue that they have never been fully detailed and proved, and that detailing them is not trivial. In [1, 4, 14], the fact that SLD-resolution is complete is assumed for different settings (e.g. assuming LAV, global-as-view GAV or global-and-local-as-view GLAV mappings), but its justification is always more or less implicit and not formally given. In [12], theorem 1 shows that SLD-resolution, applied on a rewriting that is a definite program, generates answers that are answers to the original query. However this soundness proof does not come with a completeness one. Moreover it is not clearly related to the notion of certain answers. Another paper [14] uses SLD-resolution for query answering in a mediation setting. But since it is not the contribution of the paper, no proof is given that all certain answers are computed.

In [9], some results are given that may suggest that a certain answer for finite models can be mapped to a certain answer for any infinite model. Intuitively, the explanation follows the following steps: (i) a certain answer for finite models is a certain answer for the chase, (ii) the chase terminates on source-to-target dependencies (which can express e.g. LAV mappings), (iii) the result of the chase is a universal solution, (iv) certain answers can be decided on this universal solution by dropping nulls, and (v) all the previous arguments seem to be straightforwardly extendable to infinite models (i.e. infinite universal solutions). Here again, we believe that the reasoning is correct, but we argue that a precise proof is needed.

In this paper, we give such a detailed proof. Trying to start from previous results, we quickly realized that we needed to redefine many basic notions in order to be as precise as possible in our results. That's why, before the compactness theorem and the certain answers characterization, a large part of this paper is dedicated to the recall of some formal notions from definite clauses to data integration concepts. We believe the obtained framework is interesting as itself and consider it as the third contribution of this paper.

Now, let's explain the idea underlying our certain answers characterization. To better understand the link between certain answers and answers of a definite program, let's illustrate it with a detailed example taken from [13].

Example 1.1. Assume that we want to build an integration system $\mathcal{I}$ that deals with bibliographic references of scientific papers in a mediation setting. The global schema $\mathcal{G}$ is modeled with two predicates[1], $cites^{(2)}$ and $sameTopic^{(2)}$: $cites(X, Y)$ indicates that the paper X cites the paper Y, and $sameTopic(X, Y)$ means that papers X and Y share the same topic. So $\mathcal{G} = \{cites, sameTopic\}$. Then, we suppose we have three data sources, each described by one source predicate, namely $v4^{(1)}$, $v5^{(2)}$ and $v6^{(2)}$. We can consider these three data sources as one, and then the source schema is $\mathcal{S} = \{v4, v5, v6\}$. To describe the content of these sources according to the global schema, three mappings are defined:
$q4(X) \leftarrow cites(X, Y), cites(Y, X)$ with $v4 \subseteq q4$,
$q5(X, Y) \leftarrow sameTopic(X, Y)$ with $v5 \subseteq q5$, and
$q6(X, Y) \leftarrow cites(X, Z), cites(Z, Y), sameTopic(X, Z)$ with $v6 \subseteq q6$.
The first mapping interprets as follows: (i) $q4$ gives the papers that cite another paper which cites them, and (ii) if the global schema was materialized, evaluating $q4$ would give a set of tuples that would contain tuples of $v4$. So we say that $v4$ is contained in $q4$. Similarly, $v5$ is contained in $q5$ describing couples of papers that have a same topic, and $v6$ in $q6$ describing couples of papers (X, Y) such that X cites a paper Z of the same topic which cites Y. Suppose now the source database $\mathcal{D}$ is the following set of tables:

$v4$
1
2
3

$v5$	
1	4
2	3
5	3

$v6$	
5	5
2	5

As proved thereafter, this mediation setting can be modeled as the following definite program $P(\mathcal{I}, \mathcal{D})$:

Facts	Rules
$v4(1).$	$cites(X, f_1(X)) \leftarrow v4(X).$
$v4(2).$	$cites(f_1(X), X) \leftarrow v4(X).$
$v4(3).$	$sameTopic(X, Y) \leftarrow v5(X, Y).$
$v5(1, 4).$	$cites(X, f_3(X, Y)) \leftarrow v6(X, Y).$
$v5(2, 3).$	$cites(f_3(X, Y), Y) \leftarrow v6(X, Y).$
$v5(5, 3).$	$sameTopic(X, f_3(X, Y)) \leftarrow v6(X, Y).$
$v6(5, 5).$	
$v6(2, 5).$	

Now, we have the user conjunctive query q expressed on $\mathcal{G}$: $q(X) \leftarrow cites(X, Y), cites(Y, X), sameTopic(X, Y)$ that looks for papers that cite other papers that cite them, having the same topic.

On one hand, in first order logic, answers of q w.r.t. $P(\mathcal{I}, \mathcal{D})$ are all substitutions θ such that
$\forall X \forall Y((cites(X, Y) \wedge cites(Y, X) \wedge sameTopic(X, Y))\theta)$ is a logical consequence of $P(\mathcal{I}, \mathcal{D})$. These can be computed by SLD-resolution [16]. Here, there is only one: $\theta : X \mapsto 5$. Thus, by soundness of SLD-resolution:
$P(\mathcal{I}, \mathcal{D}) \models$
$\forall Y(cites(5, Y) \wedge cites(Y, 5) \wedge sameTopic(5, Y))$
i.e. all models of $P(\mathcal{I}, \mathcal{D})$ are models of $\forall Y(cites(5, Y) \wedge cites(Y, 5) \wedge sameTopic(5, Y))$. Moreover, by completeness of SLD-resolution, any other answer substitution is an instance of θ.

On the other hand, the certain answers of q w.r.t. $\mathcal{I}$ and $\mathcal{D}$ is the intersection of all $q(\mathcal{D}')$ where $\mathcal{D}'$ is a database on $\mathcal{G}$ such that all mappings of $\mathcal{I}$ are satisfied w.r.t. $\mathcal{D}$, and $q(\mathcal{D}')$ is the evaluation of q over $\mathcal{D}'$. This intersection may be infinite. But each $\mathcal{D}'$ is finite.

The ambition of this paper is to prove that certain answers, defined as an intersection of finite models, exactly correspond to ground answer substitutions which definition involve both finite and infinite models.

The outline is the following. In section 2 we recall the semantics of definite programs. We focus on a tuple point of view (instead of a substitution one) so that linking it with answers of a database query is made easier. In section 3, we recall notions of the relational model from the answer tuple p.o.v., especially the characterization of conjunctive query evaluation. In section 4 we present our contribution: we extend the notion of certain answers (mainly to infinite models), show that they coincide with classical certain answers (we called this result *certain answers compactness*), and then use this compactness to show classical certain answers exactly coincide to ground answer tuples. We then conclude.

2 DEFINITE PROGRAM SEMANTICS FROM THE ANSWER TUPLE P.O.V.

The aim of this section is to give (definition 2.6) the definite program semantics in terms of answer tuples (instead of answer substitutions) since it is closer to certain answers (cf. section 4). Thus section

[1]The number between brackets is the predicate arity.

2.1 recalls many basic definitions, many of which being taken from [16]. Then, in section 2.2 we recall what is an answer to a definite goal w.r.t. a definite program, from both the (classical) substitution the tuple p.o.v. We assume the reader is familiar with first order logic syntax and semantics.

2.1 Preliminary notations and definitions

The set of variables that are used throughout the paper is denoted by *Var* and the set of terms is denoted by *Term*. Any constant is denoted by a lowercase letter such as a, b, c or indexed ones a_i, b_i, c_i. Any variable is denoted by a capital letter such as X, Y, Z or by indexed ones X_i, Y_i or Z_i. Any generic term (constant, variable or functional term) is denoted by a calligraphic capital letter such as $\mathcal{T}$ or indexed ones $\mathcal{T}_i$. Atomic formulas are denoted by capital letters such as A, B or indexed ones A_i, B_i.

Given $n \in \mathbb{N}$, an *n-tuple* of terms is an element of $Term^n$; a *tuple* is an n-tuple for some $n \in \mathbb{N}$. When a tuple is given by its elements, those are written into $< \cdots >$. For instance $< \mathcal{T}_1, \cdots, \mathcal{T}_i, \cdots, \mathcal{T}_n >$ is a tuple: its i^{th} term is denoted by $< \mathcal{T}_1, \cdots, \mathcal{T}_i, \cdots, \mathcal{T}_n > (i) := \mathcal{T}_i$, its *range* is the set $\{\mathcal{T}_1, \cdots, \mathcal{T}_i, \cdots, \mathcal{T}_n\}$. Any tuple of terms is denoted by an overlined calligraphic capital letter such as $\overline{\mathcal{T}}$. Any tuple of constants is denoted by an overlined lowercase letter, such as $\overline{c}$. Any tuple of variables is denoted by an overlined capital letter, such as $\overline{X}$. The *tuple concatenation* $\overline{X} \cdot \overline{Z}$ of two variable tuples $\overline{X} =< X_1, \ldots, X_n >$ and $\overline{Z} =< Z_1, \ldots, Z_m >$, such that $\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, m\}, X_i \neq Z_j$, is the tuple $< X_1, \ldots, X_n, Z_1, \ldots, Z_m >$. The size of a tuple of terms $\mathcal{T}$ is the number of all terms that are in $\mathcal{T}$ and is noted $|\mathcal{T}|$.

A *term-substitution* or *substitution* σ is an application that maps each variable to a term (a constant, another variable or a functional term). We use the postfix or left to right mapping notation for substitutions, for instance the image by σ of X is denoted by $X\sigma$. The set $\{X \in Var | X\sigma \neq X\}$, called *support of* σ and denoted by $\text{supp}(\sigma)$ is finite. The set of term-substitutions is denoted by TS. A substitution is ground when images of variables in its support do not contain any variable. The set of ground term-substitutions is noted GTS. Let $\overline{X} =: < X_1, \ldots, X_n >$ be a n-tuple of pairwise distinct variables and $\overline{\mathcal{T}} =: < \mathcal{T}_1, \ldots, \mathcal{T}_n >$ be a term n-tuple, the substitution which support is the range of $\overline{X}$ and such that it maps the i^{th} variable of $\overline{X}$ to the i^{th} term of $\overline{\mathcal{T}}$, for each $i \in \{1, .., n\}$, is denoted by $\{\overline{X}/\overline{\mathcal{T}}\}$ or $\{X_1/\mathcal{T}_1, \cdots, X_n/\mathcal{T}_n\}$. For any substitution θ and any n-tuple of variables $\overline{X}$ knowing that $\overline{X} =< X_1, \ldots, X_i, \ldots, X_n >$ the *image of* $\overline{X}$ *by* θ is the n-tuple $\overline{X}\theta :=< X_1\theta, \ldots, X_i\theta, \ldots, X_n\theta >$. For any substitution σ and tuple of variables $\overline{Y}$ the substitution $\left\{\overline{Y}/\overline{Y}\sigma\right\}$ is called the *restriction of the substitution* σ *to the tuple of variables* $\overline{Y}$ and is denoted $\sigma\restriction_{\overline{Y}}$. A *variable-substitution* θ is a term-substitution that maps variables to variables.

A *valuation* v is an application that maps each variable in *Var* to an element of a set which is the domain of a structure (see definition 3.1). v is said to be associated to this structure. Let v a valuation associated to a structure. Let $\overline{X} =< X_1, \ldots, X_n >$ be a n-tuple of pairwise distinct variables of *Var*. We use the same notation as the one for substitution to denote the precise images by v of the variables in $\overline{X}$. For example, with $\overline{X} =< X_1, X_2, X_3, X_4 >$, v can be defined as $\{X_1/a, X_2/b, X_3/c, X_4/a\}$. Otherwise we use the prefix notation for valuations. For instance, we note the previous example $v(\overline{X}) =< a, b, c, a >$. Since Var is infinite, the domain of a valuation is also infinite. However, we usually use a valuation in a precise context where there is a finite number of variables. The previous substitution-like notation for valuations suits well to these contexts.

The set of variables of a conjunction of atoms $A_1 \wedge \cdots \wedge A_m$ is noted $var(A_1 \wedge \cdots \wedge A_m)$. The set of variable of a variable tuple $\overline{Y}$ is noted $var(\overline{Y})$. The tuple of variables of a set E of variables is noted $\overline{E}$.

We now recall some notions about clauses, goals and programs. We assume there is an always true (i.e. in every structure) nullary connective noted $\blacksquare$ and an always false (i.e. in every structure) nullary connective noted $\square$.

Definition 2.1. Clause, definite clause
A *clause* is a first-order logic formula $\forall L_1 \vee \cdots \vee L_n$ where each L_i is an atomic formula (a positive literal) or the negation of an atomic formula (a negative literal)[2]. A *definite clause* is a clause that contains exactly one positive literal, so having the form: $\forall(A_0 \vee \neg A_1 \vee \cdots \vee \neg A_n)$. The notational convention is to write such a definite clause as a so-called rule: $A_0 \leftarrow A_1, \ldots, A_n$ for $n \geq 0$. In this rule notation, the universal quantifier is omitted. A_0 is called the *head* of the rule and the expression $A_1, \ldots, A_n$ the *body* of the rule, which corresponds to the formula $A_1 \wedge \cdots \wedge A_n$.

When the body is empty (i.e. when $n = 0$) the implication arrow may be omitted. In this case, the clause is called a fact. It corresponds to the case where the body is $\blacksquare$, i.e. A_0 is equivalent to $A_0 \leftarrow$ and to $A_0 \leftarrow \blacksquare$. Note that there are ground facts, i.e. facts that do not contain any variable, and non-ground facts, i.e. facts that contain universally quantified variables. When the head is empty (but not the body), the clause is called a definite goal (cf definition 2.2 below). The empty definite clause is the degenerated case when both the body and the head are empty. It is written $\square$ since it corresponds to $\leftarrow \blacksquare$, which is equivalent to $\neg\blacksquare$.

Variables in the body and not in the head may be seen as existential variables, since, for instance, the following formulas are all equivalent [16]:
$\forall X \forall Y \forall Z(gdchild(X, Y) \leftarrow child(X, Z) \wedge child(Z, Y))$
$\forall X \forall Y \forall Z(gdchild(X, Y) \vee \neg(child(X, Z) \wedge child(Z, Y)))$
$\forall X \forall Y(gdchild(X, Y) \vee \forall Z \neg(child(X, Z) \wedge child(Z, Y)))$
$\forall X \forall Y(gdchild(X, Y) \vee \neg\exists Z(child(X, Z) \wedge child(Z, Y)))$
$\forall X \forall Y(gdchild(X, Y) \leftarrow \exists Z(child(X, Z) \wedge child(Z, Y)))$
Let $\Pi : B_0 \leftarrow B_1, \ldots, B_n$ be a definite clause in P, the set of variables of Π is noted $var(\Pi) = \cup_{i=0}^{n} var(B_i)$. Let's define now what is a definite goal, a definite query and a definite program.

Definition 2.2. Definite goal, query, program
A *definite goal* is a clause that contains no positive literal, and at least one negative, so having the form: $\forall(\neg A_1 \vee \cdots \vee \neg A_n)$, for $n \geq 1$, also written $\leftarrow A_1, \ldots, A_n$. A *definite query* is a formula of the following kind $\exists\overline{Z}\, A_1 \wedge \ldots \wedge A_n$ where: (i) the A_is are atomic formulas, (ii) $\overline{Z}$ contains the existentially quantified variables of the formula (these variables are thus called existential), (iii) the free variables of the formulas are contained in $\overline{X}$ (these variables are also called distinguished), and (iv) w.l.o.g. we pose $\overline{X} \cdot \overline{Z} = \overline{var(A_1 \wedge \cdots \wedge A_n)}$.

[2]The universal quantifier means that each variable of the formula is universally quantified.

The name of such a definite query is a rounded capital letter followed by $\left(\overline{X};\overline{Z}\right)$, e.g. $Q\left(\overline{X};\overline{Z}\right)$. The tuple $\overline{Y} = \overline{X} \cdot \overline{Z}$ is called the *query variables tuple*, $\overline{X}$ is called the *distinguished query variables tuple* and $\overline{Z}$ is called the *existential query variables tuple* of $Q\left(\overline{X};\overline{Z}\right)$. The *definite goal associated to* $Q\left(\overline{X};\overline{Z}\right)$ is $Q\left(\overline{X};\overline{Z}\right) : \leftarrow A_1, \ldots, A_m$. Both the definite query and its associated definite goal have the same name, which is $Q\left(\overline{X};\overline{Z}\right)$. Let $Q\left(\overline{X};\overline{Z}\right) : \exists\overline{Z}, A_1 \wedge \ldots \wedge A_n$ be a definite query. Let $Q_2\left(\overline{X};\overline{Z}\right) : \leftarrow B_1, \ldots, B_m$ be a definite goal. We define the notion of "body" as follows: (i) Body($Q\left(\overline{X};\overline{Z}\right)$) is the formula $A_1 \wedge \ldots \wedge A_n$ (without $\exists\overline{Z}$), and (ii) Body($Q_2\left(\overline{X};\overline{Z}\right)$) is the following expression $B_1, \ldots, B_m$ (without the left arrow). A *definite program* is a finite set of definite clauses and/or facts. An *extended definite program* is the union of a finite set of definite clauses and a (possibly infinite) set of facts.

2.2 Goal answers and query answers

In this section, we define what is a goal answer, first in terms of a substitution (definition 2.3), and second in terms of a tuple of terms (definition 2.4). These two definitions are semantical in the sense that they refer to logical implication only (they do not refer to any syntactical procedure to compute these answers).

As usual, the logical implication symbol $\models$ has two meanings. First, $\mathcal{I} \models \varphi$ means that the structure $\mathcal{I}$ satisfies the closed formula φ. We say also φ is true in $\mathcal{I}$, and also $\mathcal{I}$ is a model of φ. Moreover, for a set of formulas $\mathcal{P}$: $\mathcal{I} \models \mathcal{P}$ iff $\mathcal{I}$ is a model of each formula in $\mathcal{P}$. Second, for a set of formulas $\mathcal{P}$, $\mathcal{P} \models \varphi$ iff $\mathcal{I} \models \varphi$, for all structure $\mathcal{I}$ s.t. $\mathcal{I} \models \mathcal{P}$. As for rules, when writing logical implication, for a sake of simplicity, the universal quantifier will usually be omitted.

Definition 2.3. Goal answer substitution
Let P be an extended definite program. A P *answer substitution* to a definite goal $Q\left(\overline{X};\overline{Z}\right) : \leftarrow A_1, \ldots, A_m$
is a substitution σ such that: $P \models (A_1 \wedge \ldots \wedge A_m)\sigma$.
The set of P answer substitutions to a goal
$Q\left(\overline{X};\overline{Z}\right) : \leftarrow A_1, \ldots, A_m$ is noted AS($Q\left(\overline{X};\overline{Z}\right), P$):
$$\mathrm{AS}(Q\left(\overline{X};\overline{Z}\right), P) = \{\sigma \in \mathrm{TS} \mid P \models (A_1 \wedge \ldots \wedge A_m)\sigma\} .$$

Note that, every extended definite program P implies logically the conjunction of an empty set of formulas denoted by $\blacksquare$. Thus AS($\square, P$) = TS since AS($\square, P$) = $\{\sigma \in \mathrm{TS} \mid P \models \blacksquare\sigma\}$ = TS.

Definition 2.4. Goal answer tuple
Let P be an extended definite program. A *goal answer tuple* of a definite goal $Q\left(\overline{X};\overline{Z}\right) :\leftarrow A_1, \ldots, A_m$ w.r.t. P is a tuple of terms $(\overline{X} \cdot \overline{Z})\sigma$ where σ is a P answer substitution to $Q\left(\overline{X};\overline{Z}\right)$. The set of goal answer tuples of $Q\left(\overline{X};\overline{Z}\right)$ w.r.t. P, is noted GAT($Q\left(\overline{X};\overline{Z}\right), P$):
$$\mathrm{GAT}(Q\left(\overline{X};\overline{Z}\right), P) = \{(\overline{X} \cdot \overline{Z})\sigma \mid \sigma \in \mathrm{AS}(Q\left(\overline{X};\overline{Z}\right), P)\} .$$

Now we know what is a goal answer tuple, we can define what is a query answer tuple. Roughly speaking, it is a restriction of a goal answer tuple over the distinguished query variable tuple.

Definition 2.5. Query answer tuple
Let P be an extended definite program, $Q\left(\overline{X};\overline{Z}\right)$ a definite query, and σ a substitution. $\overline{X}\sigma$ is a *query answer tuple to* $Q\left(\overline{X};\overline{Z}\right)$ *w.r.t.* P, if $(\overline{X} \cdot \overline{Z})\sigma$ is a goal answer tuple of $Q\left(\overline{X};\overline{Z}\right)$. I.e. the set of query answer tuples of the definite query $Q\left(\overline{X};\overline{Z}\right)$ w.r.t. P, noted QAT($Q\left(\overline{X};\overline{Z}\right), P$), is:
$$\mathrm{QAT}(Q\left(\overline{X};\overline{Z}\right), P) = \{\overline{X}\sigma \mid \sigma \in \mathrm{AS}(Q\left(\overline{X};\overline{Z}\right), P)\} .$$

Therefore each goal answer tuple corresponds to one query answer tuple, and each query answer tuple corresponds to at least one goal answer tuple. In the following, we focus on ground query answer tuples.

Definition 2.6. Ground query answer tuple
Let P be an extended definite program and $Q\left(\overline{X};\overline{Z}\right)$ be a definite query . We define the set of *ground query answer tuples* of $Q\left(\overline{X};\overline{Z}\right)$ w.r.t. P as:
GQAT($Q\left(\overline{X};\overline{Z}\right), P$) =
$\{\overline{\mathcal{T}} \in \mathrm{QAT}(Q\left(\overline{X};\overline{Z}\right), P) \mid \overline{\mathcal{T}}$ is ground$\}$

Example 2.7. Going on with our running example, the definite query $Q(\langle X\rangle ; \langle Y\rangle)$ equivalent to the conjunctive query q is:
$\leftarrow \exists Y\, cites(X, Y) \wedge cites(Y, X) \wedge sameTopic(X, Y)$
for which there is:
GAT($Q(\langle X\rangle ; \langle Y\rangle), P(\mathcal{I}, \mathcal{D})$) = $\{\langle 5, f_3(5, 5)\rangle\}$,
QAT($Q(\langle X\rangle ; \langle Y\rangle), P(\mathcal{I}, \mathcal{D})$) = $\{\langle 5\rangle\}$, and
GQAT($Q(\langle X\rangle ; \langle Y\rangle), P(\mathcal{I}, \mathcal{D})$) = $\{\langle 5\rangle\}$.

3 RECALLS IN RELATIONAL DATABASES

Based on recalls from [3, 4, 6], the aim of this section is to prove that definite queries have the same semantics as conjunctive queries when evaluated on finite or infinite relational structures (theorem 3.15). This result is not new but is given here from the answer tuple point of view. Usually, in relational databases, conjunctive queries are evaluated on a relational database which is, roughly, a finite, relational Herbrand model. To fill the gap with finite or infinite relational structures, we mainly introduce, in section 3.1, the notion of name extension for a first order language and a relational structure. This allows each element of a relational structure to be the interpretation of a constant in the structure (either itself or a preexisting constant), thus getting close to a Herbrand model (cf. definition 3.4 and lemma 4.2). In section 3.2, we use name extension to define relational databases. In section 3.3, we recall that a relational structure can be viewed as an extended definite program and we show that logical implication w.r.t. this structure coincides, for definite queries, with logical implication w.r.t. the corresponding program. Then, in section 3.4, after defining conjunctive queries, we give the characterization of their semantics in terms of ground query answer tuples. We end this section by giving a characterization of conjunctive query inclusion in terms of logical implication.

3.1 Relational structures and name extensions

Definition 3.1. Relational structure, active domain
A *FO relational language* $\mathcal{L}$ is defined as a couple $(\mathcal{R}, C)$ where $\mathcal{R}$ is a non empty finite set of relation symbols and C is a (possibly infinite and possibly empty) set of constant symbols such that $\mathcal{R} \cap C = \emptyset$. When C is empty, we talk about the relational schema $\mathcal{R}$ instead of the FO relational language $\mathcal{L}$.

Assuming a FO relational language $\mathcal{L} = (\mathcal{R}, C)$. A *relational structure* $\mathcal{D}$ for $\mathcal{L}$ (or $\mathcal{L}$-structure) is a triple $\langle |\mathcal{D}|, \mathcal{L}, .^{\mathcal{D}} \rangle$, defined as follows:

- $|\mathcal{D}|$ is a (possibly infinite) set called the universe (or the domain) of $\mathcal{D}$.
- $\mathcal{L} = (\mathcal{R}, C)$ is the signature (or the alphabet) of $\mathcal{D}$. $\mathcal{R}$ is called the schema of $\mathcal{D}$. Each predicate symbol $r_i \in \mathcal{R}$ is associated to an integer ≥ 1 called its arity and noted $arity(r_i)$.
- $.^{\mathcal{D}}$ is an interpretation function
  - which assigns an element of $|\mathcal{D}|$ to each constant symbol $c \in C$, and
  - which assigns a subset of $|\mathcal{D}|^{arity(r_i)}$ to each relation symbol $r_i \in \mathcal{R}$. This subset is noted $r_i^{\mathcal{D}}$ and is called the interpretation of r_i in $\mathcal{D}$.

When $C = \emptyset$, we say $\mathcal{D}$ is a relational structure *for the schema* $\mathcal{R}$. The structure is *finite* when all $r_i^{\mathcal{D}}$ is finite.

The *active domain of* $\mathcal{D}$, denoted by adom($\mathcal{D}$), is the set of elements of the domain which occur in some interpretation of a relation of $\mathcal{D}$.

Definition 3.2. Name extension
Given a FO relational language $\mathcal{L} = (\mathcal{R}, C)$ and a structure $\mathcal{D}$ for $\mathcal{L}$. The *name extension* of the language $\mathcal{L}$ by $\mathcal{D}$ is the FO relational language $(\mathcal{L})^{\nu} = (\mathcal{R}, C^{\nu})$ such that C^{ν} is the union of C (the set of constants in $\mathcal{L}$) with the set of elements of $|\mathcal{D}|$ which are not equal to an interpretation of a constant in C.

Given a FO relational language $\mathcal{L} = (\mathcal{R}, C)$ and a relational structure $\mathcal{D} = \langle |\mathcal{D}|, \mathcal{L}, .^{\mathcal{D}} \rangle$ for $\mathcal{L}$.
$\mathcal{D}^{\nu} = \langle |\mathcal{D}^{\nu}|, \mathcal{L}^{\nu}, .^{\mathcal{D}^{\nu}} \rangle$ is the *name extension* of $\mathcal{D}$ if:

- $|\mathcal{D}^{\nu}| = |\mathcal{D}|$
- $\mathcal{L}^{\nu}$ is the name extension of $\mathcal{L}$ by $\mathcal{D}$
- and $.^{\mathcal{D}^{\nu}}$:
  - $c \mapsto c^{\mathcal{D}^{\nu}} = c^{\mathcal{D}}, \forall c \in C$
  - $c \mapsto c^{\mathcal{D}^{\nu}} = c, \forall c \in C^{\nu} \backslash_C$
  - $r_i \mapsto r_i^{\mathcal{D}^{\nu}} = r_i^{\mathcal{D}}, \forall r_i \in \mathcal{R}$.

Definition 3.3. Name, name substitution and name set
Given a FO relational language $\mathcal{L} = (\mathcal{R}, C)$, and $\mathcal{D}$ be an structure for $\mathcal{L} = (\mathcal{R}, C)$. For each $c \in |\mathcal{D}|$ we say that a constant $c' \in C^{\nu}$ is a *name* for c if $(c')^{\mathcal{D}^{\nu}} = c$. The notation $N(c', c)$ means that c' is a name for c.

Let v be a valuation from the set of variables *Var* to $|\mathcal{D}|$, we say that a substitution v' is a *name substitution* for v if for each variable X, Xv' is a name for $v(X)$ (i.e. $N(Xv', v(X))$); we abbreviate by $N(v', v)$.

Let $E \subseteq |\mathcal{D}|^n$. We define the *name set* of E as follows: $\mathcal{N}(E) = \{\langle c'_1, \ldots, c'_n \rangle \in (C^{\nu})^n \mid \exists \langle c_1, \ldots, c_n \rangle \in E \text{ s.t. } N(c'_i, c_i), \forall i \in \{1, \ldots, n\}\}$.

In other words, we have $N(c', c)$ iff either ($c \in |\mathcal{D}| \backslash C^{\mathcal{D}}$ and $c = c'$), or ($c' \in C$ and $c'^{\mathcal{D}} = c$). Moreover, a valuation has always at least one name substitution, and may have several name substitutions if the interpretation by $\mathcal{D}$ of constants in C is not one to one. By contrast, a ground substitution can be a name substitution for only one valuation.

3.2 Relational databases

We can now define what is a database.

Definition 3.4. Relational database
A *relational database* (or simplier a *database*) is the structure name extension of a relational structure which language has no constant and where all predicate interpretations are finite.

Note that:

- Since a relational database is a relational structure then its universe may be infinite as in example 3.5.
- For the sake of simplicity, we will denote databases $\mathcal{D}^{\nu}$ as relational structures $\mathcal{D}$, i.e. without the $^{\nu}$ superscript. For instance, $\mathcal{D}$ will be used to denote either a database or a relational structure.
- If $\mathcal{D}$ is a database and $\mathcal{R}$ is the set of relation names, then we say $\mathcal{D}$ is a (relational) database for the schema $\mathcal{R}$.

Example 3.5. Going on with our running example, we suppose we have the relational language $\mathcal{L} = (\mathcal{R}, C)$ with $\mathcal{R} = \{v4, v5, v6\}$ and $C = \emptyset$, and the relational structure $\mathcal{D} = \langle |\mathcal{D}|, \mathcal{L}, .^{\mathcal{D}} \rangle$ for $\mathcal{L}$ defined as follows: $|\mathcal{D}| = \mathbb{N}$ and $.^{\mathcal{D}}$ is such that $v4^{\mathcal{D}} = \{\langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle\}$, $v5^{\mathcal{D}} = \{\langle 1, 4 \rangle, \langle 2, 3 \rangle, \langle 5, 3 \rangle\}$, and $v6^{\mathcal{D}} = \{\langle 5, 5 \rangle, \langle 2, 5 \rangle\}$. Then the structure name extension of $\mathcal{D}$ is a finite structure $\mathcal{D}^{\nu} = \langle |\mathcal{D}^{\nu}|, \mathcal{L}^{\nu}, .^{\mathcal{D}^{\nu}} \rangle$ defined as follows: $|\mathcal{D}^{\nu}| = |\mathcal{D}| = \mathbb{N}$, $\mathcal{L}^{\nu} = (\{v4, v5, v6\}, \mathbb{N})$, and $.^{\mathcal{D}^{\nu}}$ is such that:
$i \mapsto i^{\mathcal{D}^{\nu}} = i, \ \forall i \in \mathbb{N}$,
$v4 \mapsto v4^{\mathcal{D}^{\nu}} = \{\langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle\} = v4^{\mathcal{D}}$,
$v5 \mapsto v5^{\mathcal{D}^{\nu}} = \{\langle 1, 4 \rangle, \langle 2, 3 \rangle, \langle 5, 3 \rangle\} = v5^{\mathcal{D}}$, and
$v6 \mapsto v6^{\mathcal{D}^{\nu}} = \{\langle 5, 5 \rangle, \langle 2, 5 \rangle\} = v6^{\mathcal{D}}$.

Let $\mathcal{D}$ be a relational database for the schema $\mathcal{R}$. That means the relational language for which $\mathcal{D}$ is a relational structure is $\mathcal{L} = (\mathcal{R}, |\mathcal{D}|)$. Since $\mathcal{D}$ is itself the name extension of a relational structure, then it is clear that the name extension of $\mathcal{L}$ by $\mathcal{D}$ is itself and that the name extension of $\mathcal{D}$ is itself as well. So it is clear that we have $N(c, c)$ for all $c \in |\mathcal{D}|$.

Lemma 3.6. *Let $\mathcal{D}$ be a database for $\mathcal{R}$. We have:*

Thus, for any valuation v from *Var* to $|\mathcal{D}|$, there is only one name substitution v', which is equal to v. Henceforth, for databases, we may denote by the same way a valuation and its name substitution, except we keep the postpositioned notation for substitutions and the prepositioned notation for valuations.

3.3 From a structure to its ground theory

In this section, we define the ground theory $\mathcal{D}_{def}$ of a relational structure $\mathcal{D}$. Informally, this is the set of all ground atomic formulas that derive from relations interpretation that define $\mathcal{D}$. We show the link between this set and the logical consequences of the name extension of $\mathcal{D}$. Since $\mathcal{D}_{def}$ can be viewed as extended definite program, this allow, in next section, to link the answers of a query over a relational structure to the answers of a definite goal over a extended definite program.

Definition 3.7. Ground theory of a relational structure
Let $\mathcal{L} = (\mathcal{R}, C)$ be a FO relational language and $\mathcal{D}$ be a relational structure for $\mathcal{L}$. The *ground theory* $\mathcal{D}_{def}$ *of* $\mathcal{D}$ is the following set of ground atomic formulas of $\mathcal{L}^v = (\mathcal{R}, C^v)$:
$\mathcal{D}_{def} := \{r(c'_1, \ldots, c'_n) \mid r \in \mathcal{R}$ and $\exists \langle c_1, \ldots, c_n \rangle \in r^{\mathcal{D}}$
s.t. $N(c'_i, c_i), \forall i \in \{1, \ldots, n\}\}$

Example 3.8. In the running example, we have:
$\mathcal{D}_{def} = \{v4(1), v4(2), v4(3), v5(1, 4), v5(2, 3), v5(5, 3), v6(5, 5), v6(2, 5)\}$.

Let's make some remarks about this definition :

- For one tuple $\langle c_1, \ldots, c_n \rangle \in r^{\mathcal{D}}$ there may be several ground atomic formulas $r(c'_1, \ldots, c'_n)$ such that $\forall i \in \{1, \ldots, n\}$, we have $N(c'_i, c_i)$.
- The FO language $\mathcal{L}_{def}$ used to build formulas that are in $\mathcal{D}_{def}$ is relational. Indeed, it is clear by construction that $\mathcal{L}_{def} = (\mathcal{R}, C^v)$, which is the same as the one of $\mathcal{D}^v$.
- It is easy to see that $\mathcal{D}^v$ is a model of $\mathcal{D}_{def}$. Moreover, it is a minimal model (w.r.t. inclusion). However $\mathcal{D}^v$ is not the only minimal model of $\mathcal{D}_{def}$ since there can be models on other domains.
- On the contrary of $\mathcal{D}^v$, $\mathcal{D}$ might not be a model of $\mathcal{D}_{def}$. Indeed, if $C^v \neq C$ then there will be atomic formulas in $\mathcal{D}_{def}$ expressed with a different language than the one for which $\mathcal{D}$ is a relational structure. Thus, $\mathcal{D}$ cannot be a model for such formulas.

Recall that, in a database $\mathcal{D}$ for a schema $\mathcal{R}$ all relation interpretations are finite. So, when these are defined by extension, then the set of ground formulas $\mathcal{D}_{def}$ is finite and can be viewed as an extended definite program (made of ground atomic formulas only).

Now, we can express logical implication for atomic formulas in terms of our previous definitions.

Lemma 3.9. *Let $\mathcal{D}$ be a relational structure for a relational language $\mathcal{L} = (\mathcal{R}, C)$. Let $\mathcal{L}^v$ be the language $(\mathcal{R}, C^v)$. Let $v : Var \to |\mathcal{D}|$ be a valuation. Let $\psi, \psi_1, \ldots, \psi_m$ be atomic formulas of $\mathcal{L}$. We have:*

(1) $(\mathcal{D} \models_v \psi) \Leftrightarrow (\mathcal{D}^v \models_v \psi)$
(2) $\left(\mathcal{D} \models_v \exists\overline{Z}(\psi_1 \wedge \ldots \wedge \psi_m)\right)$
$\Leftrightarrow \left(\mathcal{D}^v \models_v \exists\overline{Z}(\psi_1 \wedge \ldots \wedge \psi_m)\right)$

Proof of previous lemma is given in appendix A.

Lemma 3.10. *Let $\mathcal{D}$ be a relational structure for a relational language $\mathcal{L} = (\mathcal{R}, C)$. Let $\mathcal{L}^v$ be the language $(\mathcal{R}, C^v)$. Let $u' : Var \to C^v$ be a ground substitution. Let $v : Var \to |\mathcal{D}|$ be a valuation and $v' : Var \to C^v$ be one of its name substitutions. Let $\varphi, \varphi_1, \ldots, \varphi_n$ be atomic formulas of $\mathcal{L}^v$. We have:*

(1) $\left(\varphi u' \in \mathcal{D}_{def}\right) \Leftrightarrow \left(\mathcal{D}_{def} \models \varphi u'\right)$
(2) $(\mathcal{D}^v \models_v \varphi) \Leftrightarrow \left(\varphi v' \in \mathcal{D}_{def}\right)$
(3) $(\mathcal{D}^v \models_v \varphi_1 \wedge \ldots \wedge \varphi_n)$
$\Leftrightarrow \left(\mathcal{D}_{def} \models (\varphi_1 \wedge \ldots \wedge \varphi_n)v'\right)$
(4) $\left(\mathcal{D}^v \models_v \exists\overline{Z}(\varphi_1 \wedge \ldots \wedge \varphi_n)\right)$
$\Leftrightarrow \left(\mathcal{D}_{def} \models (\exists\overline{Z}(\varphi_1 \wedge \ldots \wedge \varphi_n))v'\right)$

Proof of previous lemma is given in appendix A.

From point 1. of lemma 3.10, we have that, for any ground atomic formula φ of of $\mathcal{L}^v$: $\varphi \notin \mathcal{D}_{def} \Leftrightarrow \mathcal{D}_{def} \not\models \varphi$. This leads us to consider the so-called Open and Closed World Assumptions, as follows:

- the Closed World Assumption (CWA) is assumed when we consider as true each formula in the atomic ground formula completion $\overline{\mathcal{D}_{def}}$ of $\mathcal{D}_{def}$, i.e.
$\overline{\mathcal{D}_{def}} = \mathcal{D}_{def} \cup \{\neg\varphi \mid \varphi$ is a ground atomic formula and $\varphi \notin \mathcal{D}_{def}\}$
- the Open World Assumption (OWA) is assumed when we consider that for any ground atomic formula φ such that $\varphi \notin \mathcal{D}_{def}$, we have nothing more than $\mathcal{D}_{def} \not\models \varphi$ (i.e. there exists a structure $\mathcal{I}$ s.t. $\mathcal{I} \models \mathcal{D}_{def}$ and $\mathcal{I} \not\models \varphi$).

We will reuse these notions in section 4.

By Lemmas 3.9 and 3.10, we conclude on the equivalence, for definite queries, of logical implication w.r.t. a relational structure and logical implication w.r.t. associated extended definite program.

Corollary 3.11. *Let $\mathcal{D}$ be a relational structure for a relational language $\mathcal{L} = (\mathcal{R}, C)$. Let $\mathcal{L}^v$ be the language $(\mathcal{R}, C^v)$. Let $v : Var \to |\mathcal{D}|$ be a valuation and $v' : Var \to C^v$ be one of its name substitutions. Let $\psi, \psi_1, \ldots, \psi_m$ be atomic formulas of $\mathcal{L}$. We have:*
$\left(\mathcal{D} \models_v \exists\overline{Z}(\psi_1 \wedge \ldots \wedge \psi_m)\right)$
$\Leftrightarrow \left(\mathcal{D}_{def} \models (\exists\overline{Z}(\psi_1 \wedge \ldots \wedge \psi_n))v'\right)$.

3.4 Conjunctive queries

In this section, we recall what is a conjunctive query (CQ) in the relational structures context. Informally, a conjunctive query is a function that associates a subset of the universe to a relational structure. It is defined by a conjunction of positive literals, which arguments may be constants, existential variables or free variables. We'll see thereafter that it corresponds to the definition of a definite query (without function symbol).

Definition 3.12. Conjunctive query syntax [2, 5]
Let $(\mathcal{R}, C)$ be a FO relational language. A conjunctive query q on $(\mathcal{R}, C)$ of arity n is an expression of the following form:
$q(\overline{X}) \leftarrow p_1(\overline{X_1}), \ldots, p_m(\overline{X_m})$
such that:

- $m \geq 1$,
- q is a symbol which belongs to a new alphabet, the alphabet of queries, disjoint from $\mathcal{R}$ and C,
- $\forall i \in \{1, \ldots, m\}, p_i \in \mathcal{R}$,
- $\forall i \in \{1, \ldots, m\}, |\overline{X_i}| = arity(p_i)$ and $|\overline{X}| = n = arity(q)$,
- $\overline{X_1}, \ldots, \overline{X_m}$ are tuples containing variables from *Var* and/or constants from C,
- w.l.o.g. $\overline{X}$ contains only variables, and
- $var(\overline{X}) \subseteq \bigcup_{i \in \{1,\ldots,m\}} var(\overline{X_i})$ (i.e. q is "safe").

The part that is on the left of the $\leftarrow$ symbol is called the *head* of the query. The part that is on the right of the $\leftarrow$ symbol is called the *body* of the query.

Now, to capture the intuition that a conjunctive query corresponds to a definite query (or goal) without function symbol, we need the following definition.

Definition 3.13. Associated definite query
Let $q(\overline{X}) \leftarrow p_1(\overline{X_1}), \ldots, p_m(\overline{X_m})$ be a conjunctive query on a FO relational language $(\mathcal{R}, C)$. The *definite query (resp. goal)* $Q\left(\overline{X};\overline{Z}\right)$ *associated to* q is the following expression:
$Q\left(\overline{X};\overline{Z}\right) : \exists\overline{Z}, A_1 \wedge \ldots \wedge A_m$
(resp.) $Q\left(\overline{X};\overline{Z}\right) : \leftarrow A_1, \ldots, A_m$, where:

- $A_i = p_i(\overline{X_i}), \forall i \in \{1, \ldots, m\}$
- $var(\overline{Z}) = (\bigcup_{i\in\{1,\ldots,m\}} var(\overline{X_i})) \setminus var(\overline{X})$

We can now introduce the following notation for conjunctive queries. A conjunctive query q on a schema $(\mathcal{R}, C)$ may now be noted
$q(\overline{X}) \leftarrow \exists\overline{Z}, \text{Body}(Q\left(\overline{X};\overline{Z}\right))$ or
$q(\overline{X}) \leftarrow \exists\overline{Z}, A_1 \wedge \ldots \wedge A_m$
knowing that $Q\left(\overline{X};\overline{Z}\right) : \exists\overline{Z}, A_1 \wedge \ldots \wedge A_m$ is its associated definite query.

Now, we give the definition of conjunctive query semantics w.r.t. a relational structure.

Definition 3.14. Conjunctive query semantic w.r.t. a relational structure
Let $\mathcal{D}$ be a relational structure for a FO relational language $(\mathcal{R}, C)$. Let q be the conjunctive query $q(\overline{X}) \leftarrow \exists\overline{Z}, A_1 \wedge \ldots \wedge A_m$ of arity n on a FO relational language $(\mathcal{R}', C')$. The *answer to* q *over* $\mathcal{D}$, denoted $q^{\mathcal{D}}$, is defined only for $\mathcal{R}' \subseteq \mathcal{R}$ and $C' \subseteq C$ as follows:
$q^{\mathcal{D}} = \{\langle c_1, \ldots, c_n\rangle \in |\mathcal{D}|^n \mid$
$\mathcal{D} \models_{\{X_1/c_1,\ldots,X_n/c_n\}} \left(\exists\overline{Z}, A_1 \wedge \ldots \wedge A_m\right)\}$
When the FO relational language $(\mathcal{R}', C')$ is not given, then it is assumed to be $(\mathcal{R}, C)$.

In the previous definition, we do see the purpose of name extension: $q^{\mathcal{D}} \subseteq q^{\mathcal{D}^v}$ and they may be different whenever some elements of the domain used in the interpretation of predicates are not themselves the interpretation of any constant. So dealing with the name extension allows to get all possible answer tuples during the evaluation of q. Definition 3.14 applies also to databases, excepted that in this case we have $q^{\mathcal{D}} = q^{\mathcal{D}^v}$ (since $\mathcal{D} = \mathcal{D}^v$ from lemma 3.6).

The previous definition shows that the $\leftarrow$ symbol used to write a conjunctive query in definition 3.12 has no logical meaning, otherwise it would be the equivalence $\leftrightarrow$ symbol. Nevertheless, it can be interpreted with an operational meaning: to effectively compute the answer to q, we have to achieve computations from what we know about predicates used in the body of q. The $\leftarrow$ denotes the way this computation is done.

We've said that the semantics of (or the answer to) a conjunctive query q with arity n over $\mathcal{D}$ for the schema $\mathcal{R}$ is denoted $q^{\mathcal{D}}$. We recall the same notation $r^{\mathcal{D}'}$ is used to denote the interpretation of a predicate symbol r of a schema $\mathcal{R}'$ in a given database $\mathcal{D}'$. So there is a common notation to denote the intentionally defined query predicate q and the extensionally defined predicate r. Whatever kind of predicate r belongs to, $r^{\mathcal{D}}$ is said to be the extension of r in $\mathcal{D}$ for the schema $\mathcal{R} \cup \{r\}$.

Since a definite query is associated to each conjunctive query, then it is natural to remark that evaluating a conjunctive query w.r.t. a database $\mathcal{D}$ must be very close to evaluating a definite query w.r.t. the ground theory $\mathcal{D}_{def}$ of $\mathcal{D}$. The following proposition clarifies this point for relational structures.

Theorem 3.15. **CQ** evaluation characterization
Let $\mathcal{D}$ be a relational structure for a FO relational language $(\mathcal{R}, C)$. Let q be the conjunctive query of arity n, associated to the definite query $Q\left(\overline{X};\overline{Z}\right)$, on the FO relational language $(\mathcal{R}', C')$ with $\mathcal{R}' \subseteq \mathcal{R}$ and $C' \subseteq C$. We have:
$\text{GQAT}(Q\left(\overline{X};\overline{Z}\right), \mathcal{D}_{def}) = \mathcal{N}(q^{\mathcal{D}})$
where (cf. def. 3.3):
$\mathcal{N}(q^{\mathcal{D}}) = \{\langle c'_1, \ldots, c'_n\rangle \in (C^v)^n \mid$
$\exists \langle c_1, \ldots, c_n\rangle \in q^{\mathcal{D}} \text{ s.t. } N(c'_1, c_1), \ldots, N(c'_n, c_n)\}$.

The proof of this theorem is given in appendix A.

Applied to relational databases, the previous result, which is databases and logic programming folklore now, is important since it shows that conjunctive queries have the same semantics as their associated definite query, which can be computed by logic programming algorithms as SLD-resolution for example.

Before focusing on data integration issues, we recall the notion of query inclusion.

Definition 3.16. Conjunctive query inclusion
Let q_1 and q_2 be two conjunctive queries on the relational language $(\mathcal{R}, C)$. We say that q_1 is contained into q_2, noted $q_1 \subseteq q_2$, if for every relational structure $\mathcal{D}$ for the language $(\mathcal{R}, C)$ we have $q_1^{\mathcal{D}} \subseteq q_2^{\mathcal{D}}$.

Query inclusion can be characterized w.r.t. logical implication as follows.

Lemma 3.17. *Let q_1 and q_2 be two conjunctive queries, with the same arity, on the same relational language $(\mathcal{R}, C)$ and let $Q_1\left(\overline{X};\overline{Z}\right)$ and $Q_2\left(\overline{W};\overline{V}\right)$ be their respective associated definite query. For every relational structure $\mathcal{D}$ for $(\mathcal{R}, C)$ we have:*
$q_1^{\mathcal{D}} \subseteq q_2^{\mathcal{D}} \Leftrightarrow \mathcal{D} \models \forall\overline{X}(\ (\exists\overline{Z} Body\left(Q_1\left(\overline{X};\overline{Z}\right)\right)) \rightarrow$
$(\exists\overline{V} Body\left(Q_2\left(\overline{X};\overline{V}\right)\right))\ )$

The proof of this lemma is given in appendix A.

4 CERTAIN ANSWERS IN MEDIATION

The aim of this section is to characterize (theorem 4.26) the certain answers of a conjunctive query q in an integration system by showing their equality with the ground query answer tuples of a definite query corresponding to q w.r.t. a definite program modeling the integration system.

The main issue to obtain this characterization comes from the fact that certain answers are defined as a (possibly infinite) intersection of databases (i.e. finite relational Herbrand models) and that the goal query answer tuples are defined w.r.t. all models of the definite program. Three elements are used to solve this issue. The first is the Herbrand model theorem (theorem 4.5 [10]) that says a first order sentence is true in all structures iff it is true in all Herbrand models. The second is a new compactness theorem

(theorem 4.20) that says (classical) certain answers are equal to a new extended notion of certain answers, namely Herbrand certain answers (definition 4.18). The third is the clausal form theorem (theorem 4.24 [7]) that says a first order sentence is true w.r.t. a set of formulas iff it is true w.r.t. the clausal form of the conjunction of all these formulas.

These three elements are presented according to the following outline. In section 4.1, we recall the definition of Herbrand model and some associated properties, including the Herbrand model theorem. Then, in section 4.2, we define what is a data integration system using Hebrand models. We show some simple associated properties. At last, in section 4.3, we define the classical and extended forms of certain answers, before showing their correspondance through the compactness theorem. Then we recall the notion of clausal form and the associated theorem, before ending the section with the certain answers characterization.

4.1 Herbrand Models

We follow [10] for Herbrand models presentation. Herbrand models are structures characterized by the fact that their universe is the set of ground terms deriving from the used first order language. Thus relational Herbrand models have their set of constants as their universe. Usually, for instance in relational databases, this set of constants is supposed to be infinite. To cope with any first order language, especially those having a finite set of constants, we need the extra notion of $\mathcal{L}$-parameters extension [10] (of a relational language) to define Herbrand model in the most general way.

Definition 4.1. Relational Herbrand model
Given a relational first order language $\mathcal{L} = (\mathcal{R}, C)$. Assuming a fixed non empty infinite countable set of constant symbols $\mathcal{U}$ (called the domain) such that $\mathcal{R} \cap \mathcal{U} = \emptyset$, $C \subset \mathcal{U}$ and $\mathcal{U} \backslash C$ is infinite. The language $\mathcal{L}^H = (\mathcal{R}, \mathcal{U})$ is called a *$\mathcal{L}$-parameters extension.* A *$\mathcal{L}^H$-Herbrand model* $\mathcal{D}$ is a relational structure $\langle \mathcal{U}, (\mathcal{R}, \mathcal{U}), .^{\mathcal{D}} \rangle$, defined as follows:

- $\mathcal{U}$ is the (Herbrand) universe of $\mathcal{D}$, often noted $|\mathcal{D}|$.
- $(\mathcal{R}, \mathcal{U})$ is the signature (or the alphabet) of $\mathcal{D}$. The set $\mathcal{R}$ is called the schema of $\mathcal{D}$. Each predicate symbol r_i is associated to an integer ≥ 1 called its arity. We note it by $arity(r_i)$.
- $.^{\mathcal{D}}$ is an interpretation function
  - which assigns each constant symbol $c \in \mathcal{U}$ to itself, and
  - which assigns a subset of $\mathcal{U}^{arity(r_i)}$ to each relation symbol $r_i \in \mathcal{R}$. This subset is noted $r_i^{\mathcal{D}}$ and is called the interpretation of r_i in $\mathcal{D}$.

From definitions 3.4 and 4.1, it is easy to derive the following lemma.

Lemma 4.2. *Given a relational first order language $\mathcal{L} = (\mathcal{R}, C)$. Let $\mathcal{L}^H = (\mathcal{R}, \mathcal{U})$ be a $\mathcal{L}$-parameters extension. Let $\mathcal{D}$ be a $\mathcal{L}^H$-Herbrand model. If* adom$(\mathcal{D})$ *is finite, then $\mathcal{D}$ is a relational database.*

Moreover, from definition 3.3, lemma 3.6 and 3.10, it is easy to derive the following results.

Lemma 4.3. *Given a relational first order language $\mathcal{L} = (\mathcal{R}, C)$, a $\mathcal{L}$-parameters extension $\mathcal{L}^H = (\mathcal{R}, \mathcal{U})$ and a $\mathcal{L}^H$-Herbrand model $\mathcal{D}$. Let $v : Var \rightarrow |\mathcal{D}|$ be a valuation. Then v is also a name substitution for v (i.e. for itself) and we have: $\left(\varphi v \in \mathcal{D}_{def}\right) \Leftrightarrow \left(\mathcal{D}_{def} \models \varphi v\right)$ for every atomic $\mathcal{L}^H$-formula φ. Moreover, for any $n \in \mathbb{N}$, if $F \subseteq |\mathcal{D}|^n$ then $\mathcal{N}(F) = F$.*

We end this section by the Herbrand model theorem, expressed using the Herbrand validity notation.

Definition 4.4. Herbrand Validity
Given a sentence φ of the relational first order language $\mathcal{L} = (\mathcal{R}, C)$. Let $\mathcal{L}^H = (\mathcal{R}, \mathcal{U})$ be a $\mathcal{L}$-parameters extension:
$\models^H \varphi$ iff $\mathcal{D} \models \varphi$ for every $\mathcal{L}^H$-Herbrand model $\mathcal{D}$.

Theorem 4.5. Herbrand Model. Theo. 5.9.4 from [10]
Given a sentence φ of the relational first order language $\mathcal{L} = (\mathcal{R}, C)$. There is:
$\models \varphi$ *iff* $\models^H \varphi$.

4.2 Data integration system

We refer the reader to [4, 6] for studies on data integration systems. In this paper, we focus on the source-centric approach of data integration, also known as the Local-As-View (LAV) mediation setting, using the LAV mappings, with which the source schema is expressed w.r.t. the global schema. More precisely, each source is considered as a relation that is defined by a view which is a conjunctive query expressed with the predicate symbols of the global schema, and all these relations constitute the source schema. In this approach, the global schema is supposed not to change when data sources join or leave the integration system [3]. On the contrary, sources may be updated easily by changing the query that defines them.

In the mediation setting, the global schema is virtual: it means we know the set of predicates that defines it, but we do not know any instance for it. Contrary to the global schema, the source schema is materialized: it means that we know both the source predicate set and the interpretation of each source predicate. In this context, a LAV mapping is a query on the global schema associated to a source predicate. The answer to this query is linked by a subset relation to the known interpretation of the associated source predicate. In other words, we know a query on the global schema (the LAV mapping), but we do not know any database for the global schema, so we do not know the answer to this query. But this answer is linked by a subset relation to a known set of tuples (the interpretation of the corresponding source predicate). This subset relation gives information about the content of the database for the global schema.

Definition 4.6. Data integration system [4, 6]
A DIS (data integration system) $\mathcal{I}$ is a triple
$\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ where:

- $\mathcal{G}$ is a finite set of predicate symbols, and is called the global schema,
- $\mathcal{S}$ is a finite set of predicate symbols, disjoint from $\mathcal{G}$ and called the source (or local) schema, and
- $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ is a set of mappings between $\mathcal{G}$ and $\mathcal{S}$ which will be defined further.

Definition 4.7. DIS source database and global instance
Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be a DIS.

- Let $\mathcal{D}$ be a database for the schema $\mathcal{S}$. $\mathcal{D}$ is called a *source database* for $\mathcal{I}$.

- Let $\mathcal{L} = (\mathcal{S} \cup \mathcal{G}, |\mathcal{D}|)$ and let $\mathcal{L}^H = (\mathcal{S} \cup \mathcal{G}, \mathcal{U})$ be a $\mathcal{L}$-parameters extension. We say that $\mathcal{L}^H$ is a Herbrand language for $(\mathcal{I}, \mathcal{D})$.
- Let $\mathcal{D}' = \left\langle \mathcal{U}, (\mathcal{S} \cup \mathcal{G}, \mathcal{U}), .^{\mathcal{D}'} \right\rangle$ be a $\mathcal{L}^H$-Herbrand model; $\mathcal{D}'$ is called a *global instance for* $\mathcal{I}$ (or a *global database for* $\mathcal{I}$ if $\mathcal{D}'$ is finite) when:
  $\mathcal{D}' \models \left(\wedge \mathcal{D}_{def}\right)$

In this context, we say that $(\mathcal{D}, \mathcal{D}')$ is a *source database and global instance pair* for $\mathcal{I}$. Notice that if $(\mathcal{D}, \mathcal{D}')$ is a source database and global instance pair for $\mathcal{I}$, then for every $s \in \mathcal{S}$ we have $s^{\mathcal{D}} \subseteq s^{\mathcal{D}'}$. Whether $s^{\mathcal{D}} = s^{\mathcal{D}'}$ or not determines the notion of conservative global instance.

Definition 4.8. Conservative global instance
Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be a DIS and $(\mathcal{D}, \mathcal{D}')$ a source database and global instance pair for $\mathcal{I}$. $\mathcal{D}'$ is a *conservative global instance* for $(\mathcal{I}, \mathcal{D})$ if for every $s \in \mathcal{S}$ we have $s^{\mathcal{D}} = s^{\mathcal{D}'}$.

A case of conservative global instance is given in example 4.13. [3] gives a proof that a conservative global instance can always be built.

Proposition 4.9. Existence of conservative instances
Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be a DIS. Let $\mathcal{D}$ be a source database. Let $\mathcal{L}^H$ be a Herbrand language for $(\mathcal{I}, \mathcal{D})$. Let $\mathcal{D}'$ be the $\mathcal{L}^H$-Herbrand model defined by
$\forall s \in \mathcal{S},\ s^{\mathcal{D}'} = s^{\mathcal{D}}$ and $\forall g \in \mathcal{G},\ g^{\mathcal{D}'} = |\mathcal{D}|^{arity(g)}$. $(\mathcal{D}, \mathcal{D}')$ is a source database and global instance pair for $\mathcal{I}$ such that $\mathcal{D}'$ is a conservative global instance for $(\mathcal{I}, \mathcal{D})$.

To complete the definition of DIS, we need to explicit the notion of LAV mapping.

Definition 4.10. LAV mapping
Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system. Let $s \in \mathcal{S}$. A *LAV mapping* between s and the global schema $\mathcal{G}$ is an expression among the following three:

- $s \subseteq q$ (sound LAV mapping),
- $s \supseteq q$ (complete LAV mapping),
- $s \equiv q$ (exact LAV mapping), which is a shortcut for both $s \subseteq q$ and $s \supseteq q$,

where q is a conjunctive query on $\mathcal{G}$ with the same arity as s.

Let $(\mathcal{D}, \mathcal{D}')$ be a source database and global instance pair for $\mathcal{I}$. A LAV mapping l is *true in* $(\mathcal{D}, \mathcal{D}')$, noted $(\mathcal{D}, \mathcal{D}') \models l$, if:

- if l has the form $s \subseteq q$ then $s^{\mathcal{D}'} \subseteq q^{\mathcal{D}'}$,
- if l has the form $s \supseteq q$ then $s^{\mathcal{D}} = s^{\mathcal{D}'} \supseteq q^{\mathcal{D}'}$ or
- if l has the form $s \equiv q$ then $s^{\mathcal{D}} = s^{\mathcal{D}'} = q^{\mathcal{D}'}$.

Generally in data integration, mappings are assumed to be sound. Since LAV mappings are defined using a query (called q in the previous definition) on the global schema $\mathcal{G}$, then we often talk about sound LAV mapping as "sound views" on $\mathcal{G}$. Intuitively, if s is a sound view it means that each tuple in the interpretation of s in the current source database is a tuple in the evaluation of q over any global instance. Moreover, s is a sound view means that there may be tuples in the evaluation of q that are not present in the interpretation of s. This means that some tuples that are not in s may be still true in q. This is why, by analogy with the OWA, we say that the OWA (on the views w.r.t. the global schema) is made whenever we deal with sound LAV mappings.

The opposite case happens when sources are still assumed to be sound but when every other unknown fact is considered to be false (w.r.t. to the interpretation of q). This means that sources are considered to be complete (in addition to be sound). So this is equivalent to the assumption that views are exact. By analogy with the CWA, we say that the CWA (on the views w.r.t. the global schema) is made whenever we deal with sound and complete (ie exact) LAV mappings.

CWA is the classical assumption when studying relational databases. On the contrary, OWA is considered the best assumption when dealing with data integration. Indeed, when trying to integrate data coming from different heterogeneous sources, it seems more realistic to consider that an unknown fact is not necessarily false. That is why, from now on, we will focus on the LAV mappings under the OWA.

The following proposition explicits the inclusion contained in a sound LAV mapping: it is basically a logical implication in $\mathcal{D}'$.

Proposition 4.11. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system. Let $(\mathcal{D}, \mathcal{D}')$ be a source database and global instance pair for $\mathcal{I}$. Let $s \subseteq q$ be a sound LAV mapping, with $Q(\overline{X}; \overline{Z})$ the definite query associated to q. We have: $(\mathcal{D}, \mathcal{D}') \models s \subseteq q$ iff*
$\mathcal{D}' \models \forall \overline{X}\, (s(\overline{X}) \rightarrow \exists \overline{Z}\, \mathrm{Body}(Q(\overline{X}; \overline{Z}))$

Proof. By definition 4.10 and lemma 3.17. □

4.3 Query answering in a DIS

Query answering is an important issue in data integration. It is the problem of computing the answer to a query posed on the global schema of a source instance of an integration system. The only data usable to generate the answers are in the sources. However we can derive information about the global schema database by reasoning from sources data and the LAV mappings. The idea is to consider only the global schema databases that are coherent with the mappings knowing the source database. These are called the valid instances of the global schema. Then answers to a query can be defined as common answers to all valid instances. These are called certain answers. Certain answers define the semantics of a data integration query answering problem [1] (with LAV mappings and under the OWA).

Definition 4.12. Valid global instance
Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system, with LAV mappings in $\mathcal{M}_{\mathcal{G},\mathcal{S}}$, assuming the OWA. Let $(\mathcal{D}, \mathcal{D}')$ a be a source database and global instance pair, for $\mathcal{I}$. $\mathcal{D}'$ is a *valid global instance* for $(\mathcal{I}, \mathcal{D})$ if $\forall l \in \mathcal{M}_{\mathcal{G},\mathcal{S}}$, $(\mathcal{D}, \mathcal{D}') \models l$; we say also that $(\mathcal{D}, \mathcal{D}')$ is a *source database and valid global instance pair* for $\mathcal{I}$.

Example 4.13. Going on with the running example with $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be the studied integration system. Let's assume we have $\mathcal{L} = (\{v4, v5, v6\}, \{1, 2, 3, 4, 5\})$ as the initial first order relational language. Then the source database $\mathcal{D}$ defined by
$v4^{\mathcal{D}} = \{\langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle\}$,
$v5^{\mathcal{D}} = \{\langle 1, 4 \rangle, \langle 2, 3 \rangle, \langle 5, 3 \rangle\}$, and
$v6^{\mathcal{D}} = \{\langle 5, 5 \rangle, \langle 2, 5 \rangle\}$
can be viewed as a $\mathcal{L}^H$-Herbrand with, for instance, $\mathcal{L}^H = (\{v4, v5, v6\},$

$\{1,2,3,4,5\} \cup \{$ even number $\geq 6\}$). $\mathcal{D}$ can also be viewed as the name extension of a relational structure $\mathcal{E}$ which language is $\mathcal{L}_E = (\{v4, v5, v6\}, \emptyset)$ and which universe is $|\mathcal{E}| = \{1,2,3,4,5\}$.
Now, let $\mathcal{K}$ be the following language:
$\mathcal{K} = (\{v4, v5, v6\} \cup \{cites, sameTopic\}, \{1,2,3,4,5\})$.
$\mathcal{D}'_1$, $\mathcal{D}'_2$ and $\mathcal{D}''$ are global instances and databases for $(\mathcal{I}, \mathcal{D})$. The database $\mathcal{D}'_1$ is defined by
$v4^{\mathcal{D}'_1} = \{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle\}$,
$v5^{\mathcal{D}'_1} = \{\langle 1,4\rangle, \langle 2,3\rangle, \langle 5,3\rangle\}$,
$v6^{\mathcal{D}'_1} = \{\langle 5,5\rangle, \langle 2,5\rangle\}$,
$cites^{\mathcal{D}'_1} = \{\langle 1,6\rangle, \langle 6,1\rangle, \langle 2,3\rangle, \langle 3,2\rangle, \langle 3,5\rangle, \langle 5,3\rangle\}$,
$sameTopic^{\mathcal{D}'_1} = \{\langle 1,4\rangle, \langle 2,3\rangle, \langle 9,12\rangle\}$,
and thus we have:
$q4^{\mathcal{D}'_1} = \{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle, \langle 5\rangle, \langle 6\rangle\}$,
$q5^{\mathcal{D}'_1} = \{\langle 1,4\rangle, \langle 2,3\rangle, \langle 9,12\rangle\}$, and
$q6^{\mathcal{D}'_1} = \{\langle 2,2\rangle, \langle 2,5\rangle\}$.
$\mathcal{D}'_1$ can be viewed as a $\mathcal{K}^H$-Herbrand model with $\mathcal{K}^H = (\{v4, v5, v6\} \cup \{cites, sameTopic\}, \mathbb{N})$. $(\mathcal{D}, \mathcal{D}'_1)$ is a source database and conservative global instance pair for $\mathcal{I}$. $\mathcal{D}'_1$ is not valid since $\langle 5,3\rangle \notin sameTopic^{\mathcal{D}'_1}$.
The database $\mathcal{D}'_2$ is defined by
$v4^{\mathcal{D}'_2} = \{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle\}$,
$v5^{\mathcal{D}'_2} = \{\langle 1,4\rangle, \langle 2,3\rangle, \langle 5,3\rangle\}$,
$v6^{\mathcal{D}'_2} = \{\langle 5,5\rangle, \langle 2,5\rangle\}$,
$cites^{\mathcal{D}'_2} = \{\langle 1,6\rangle, \langle 6,1\rangle, \langle 2,3\rangle, \langle 3,2\rangle, \langle 3,5\rangle, \langle 5,3\rangle\}$,
$sameTopic^{\mathcal{D}'_2} = \{\langle 1,4\rangle, \langle 2,3\rangle, \langle 5,3\rangle, \langle 9,12\rangle\}$,
and thus we have:
$q4^{\mathcal{D}'_2} = \{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle, \langle 5\rangle, \langle 6\rangle\}$,
$q5^{\mathcal{D}'_2} = \{\langle 1,4\rangle, \langle 2,3\rangle, \langle 5,3\rangle, \langle 9,12\rangle\}$, and
$q6^{\mathcal{D}'_2} = \{\langle 5,5\rangle, \langle 5,2\rangle, \langle 2,2\rangle, \langle 2,5\rangle\}$.
$\mathcal{D}'_2$ can be viewed as a $\mathcal{K}^H$-Herbrand model with $\mathcal{K}^H = (\{v4, v5, v6\} \cup \{cites, sameTopic\}, \mathbb{N})$. $(\mathcal{D}, \mathcal{D}'_2)$ is a source database and valid conservative global instance pair for $\mathcal{I}$.
The instance $\mathcal{D}''$ is defined by
$v4^{\mathcal{D}''} = \{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle, \langle 6\rangle\}$,
$v5^{\mathcal{D}''} = \{\langle 1,4\rangle, \langle 2,3\rangle, \langle 5,3\rangle, \langle 9,12\rangle\}$,
$v6^{\mathcal{D}''} = \{\langle 5,5\rangle, \langle 2,5\rangle\}$,
$cites^{\mathcal{D}''} = \{\langle 1,6\rangle, \langle 6,1\rangle, \langle 2,3\rangle, \langle 3,2\rangle, \langle 3,5\rangle, \langle 5,3\rangle\} \cup \{\langle i, i+1\rangle \mid i \geq 11\}$,
$sameTopic^{\mathcal{D}''} = \{\langle 1,4\rangle, \langle 2,3\rangle, \langle 5,3\rangle, \langle 9,12\rangle\}$,
and thus we have:
$q4^{\mathcal{D}''} = \{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle, \langle 5\rangle, \langle 6\rangle\}$,
$q5^{\mathcal{D}''} = \{\langle 1,4\rangle, \langle 2,3\rangle, \langle 5,3\rangle, \langle 9,12\rangle\}$, and
$q6^{\mathcal{D}''} = \{\langle 5,5\rangle, \langle 5,2\rangle, \langle 2,2\rangle, \langle 2,5\rangle\}$.
$\mathcal{D}''$ can be viewed as a $\mathcal{K}^H$-Herbrand model with $\mathcal{K}^H = (\{v4, v5, v6\} \cup \{cites, sameTopic\}, \mathbb{N})$. $(\mathcal{D}, \mathcal{D}'')$ is a source database and valid global instance pair for $\mathcal{I}$. $\mathcal{D}''$ is not conservative since $\langle 9,12\rangle \in v5^{\mathcal{D}''}$.

In the sequel, we may use the notations *vgi* for valid global instance and *vcgd* for valid conservative global database. Using the next definition, lemma 4.15 gives a characterization of valid global instances for a DIS.

Definition 4.14. Associated sentence
Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system. Let $m : s \subseteq q$ be a sound LAV mapping, with $Q(\overline{X}; \overline{Z})$ the definite query associated to q. The sentence $m' : \left(\forall \overline{X}\,(s(\overline{X}) \rightarrow \exists \overline{Z}\,\mathrm{Body}(Q(\overline{X}; \overline{Z}))\right)$ is the *associated sentence to the mapping m*. Moreover we use the notation: $\mathcal{M}'_{\mathcal{G},\mathcal{S}} = \{m' \mid \text{s.t. } m' \text{ is the associated sentence to a mapping } m \text{ with } m \in \mathcal{M}_{\mathcal{G},\mathcal{S}}\}$.

Lemma 4.15. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system such that the mappings in $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ are LAV mappings assuming OWA. Let $(\mathcal{D}, \mathcal{D}')$ a be a source database and global instance pair $(\mathcal{D}, \mathcal{D}')$ for $\mathcal{I}$. The following statements are equivalent:*

- *$(\mathcal{D}, \mathcal{D}')$ is a source database and valid global instance pair for $\mathcal{I}$.*
- *$\mathcal{D}' \models \left(\bigwedge \mathcal{D}_{def}\right) \wedge \left(\bigwedge \mathcal{M}'_{\mathcal{G},\mathcal{S}}\right)$.*

Proof. By Definition 4.7 and Proposition 4.11. □

Now, let's give the definitions of the certain answers of a query, and then of the query answering problem.

Definition 4.16. Certain answers of a query in a LAV mediation system [1]
Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system, with LAV mappings in $\mathcal{M}_{\mathcal{G},\mathcal{S}}$, assuming the OWA. Let $\mathcal{D}$ be a source database for $\mathcal{I}$. Let q be a conjunctive query on $\mathcal{G}$. The certain answers of q w.r.t. $(\mathcal{I}, \mathcal{D})$, noted $\mathrm{Cert}(q, \mathcal{I}, \mathcal{D})$, is defined as follows:
$$\mathrm{Cert}(q, \mathcal{I}, \mathcal{D}) = \bigcap_{\substack{\mathcal{D}' \text{ valid conservative} \\ \text{global database for } (\mathcal{I}, \mathcal{D})}} q^{\mathcal{D}'}$$

Definition 4.17. Query answering problem
Using the assumptions from definition 4.16, the query answering problem of q w.r.t. $(\mathcal{I}, \mathcal{D})$ is the problem of computing $\mathrm{Cert}(q, \mathcal{I}, \mathcal{D})$.

The previous definition of certain answers is restricted to databases, i.e. finite instances, that are conservative. These two restrictions are abandoned in the following extended definition of certain answers. Despite this relaxation, theorem 4.20 (via lemma 4.19), shows they amount to the same set of tuples.

Definition 4.18. Herbrand certain answers of a query in a LAV mediation system
The Herbrand certain answers of q w.r.t. $(\mathcal{I}, \mathcal{D})$, noted $\mathrm{Cert}^H(q, \mathcal{I}, \mathcal{D})$, is defined as follows:
$$\mathrm{Cert}^H(q, \mathcal{I}, \mathcal{D}) = \bigcap_{\substack{\mathcal{D}' \text{ valid global} \\ \text{instance for } (\mathcal{I}, \mathcal{D})}} q^{\mathcal{D}'}$$

By definitions 4.16 and 4.18 and by proposition 4.9, it is easy to show the following lemma.

Lemma 4.19. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an DIS, with LAV mappings in $\mathcal{M}_{\mathcal{G},\mathcal{S}}$, assuming the OWA. Let $\mathcal{D}$ be a source database for $\mathcal{I}$. Let q be a conjunctive query on $\mathcal{G}$. Let n be the arity of q. There is:* $\mathrm{Cert}^H(q, \mathcal{I}, \mathcal{D}) \subseteq \mathrm{Cert}(q, \mathcal{I}, \mathcal{D}) \subseteq |\mathcal{D}|^n$

Theorem 4.20. Certain answers compactness for conjunctive query data integration
Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system, with LAV mappings in $\mathcal{M}_{\mathcal{G},\mathcal{S}}$, assuming the OWA. Let $\mathcal{D}$ be a source database for $\mathcal{I}$. Let q be a conjunctive query on $\mathcal{G}$. We have:
$\mathrm{Cert}^H(q, \mathcal{I}, \mathcal{D}) = \mathrm{Cert}(q, \mathcal{I}, \mathcal{D})$

Proof. Let $\mathcal{D}'$ be a valid global instance for $(\mathcal{I}, \mathcal{D})$. Let $q : q(\overline{X}) \leftarrow \exists \overline{Z}, A_1 \wedge \ldots \wedge A_m$ be a query on $\mathcal{G}$. Let n be the arity of q.

For each answer $\overline{c} \in q^{\mathcal{D}'} \cap |\mathcal{D}|^n$ we choose $\overline{t}_{\overline{c}}$ such that:
$\mathcal{D}' \models (A_1 \wedge \ldots \wedge A_m)\{\overline{X}/\overline{c}, \overline{Z}/\overline{t}_{\overline{c}}\}$.

We define the set of atomic formulas:
$\mathcal{D}'_q = \bigcup_{\overline{c} \in q^{\mathcal{D}'} \cap |\mathcal{D}|^n} \left\{A_1\{\overline{X}/\overline{c}, \overline{Z}/\overline{t}_{\overline{c}}\}, \ldots, A_m\{\overline{X}/\overline{c}, \overline{Z}/\overline{t}_{\overline{c}}\}\right\}$ We notice that $\mathcal{D}'_q \subseteq \mathcal{D}'_{def}$.

Given $\mathcal{S} = \{s_j, j \in \{1, \ldots, r\}\}$ and $\mathcal{M}_{\mathcal{G},\mathcal{S}} = \{m_j, j \in \{1, \ldots, r\}\}$. Given the mapping $m_j : s_j \subseteq q_j$, with $j \in \{1, \ldots, r\}$, we define in a similar way $\mathcal{D}'_{q_j}$; we notice that $\mathcal{D}'_{q_j} \subseteq \mathcal{D}'_{def}$.

Let $\widehat{\mathcal{D}'}$ be the valid conservative global database for $(\mathcal{I}, \mathcal{D})$ defined by: $\widehat{\mathcal{D}'}_{def} = \mathcal{D}_{def} \cup \mathcal{D}'_q \cup \mathcal{D}'_{q_1} \cup \cdots \cup \mathcal{D}'_{q_r}$. We notice that $\widehat{\mathcal{D}'}_{def} \subseteq \mathcal{D}'_{def}$, hence by monotonicity $q^{\widehat{\mathcal{D}'}} \cap |\mathcal{D}|^n \subseteq q^{\mathcal{D}'} \cap |\mathcal{D}|^n$. On the other hand, by construction, $q^{\mathcal{D}'} \cap |\mathcal{D}|^n \subseteq q^{\widehat{\mathcal{D}'}} \cap |\mathcal{D}|^n$. Hence $q^{\mathcal{D}'} \cap |\mathcal{D}|^n = q^{\widehat{\mathcal{D}'}} \cap |\mathcal{D}|^n$. So:

$$
\begin{aligned}
&\mathrm{Cert}^H(q, \mathcal{I}, \mathcal{D}) \\
=\ & \mathrm{Cert}^H(q, \mathcal{I}, \mathcal{D}) \cap |\mathcal{D}|^n \\
=\ & \left(\bigcap_{\mathcal{D}' \text{ vgi for } (\mathcal{I},\mathcal{D})} q^{\mathcal{D}'}\right) \cap |\mathcal{D}|^n \\
=\ & \bigcap_{\mathcal{D}' \text{ vgi for } (\mathcal{I},\mathcal{D})} \left(q^{\mathcal{D}'} \cap |\mathcal{D}|^n\right) \\
=\ & \bigcap_{\mathcal{D}' \text{ vgi for } (\mathcal{I},\mathcal{D})} \left(q^{\widehat{\mathcal{D}'}} \cap |\mathcal{D}|^n\right) \\
\supseteq\ & \bigcap_{\mathcal{D}'' \text{ vcgd for } (\mathcal{I},\mathcal{D})} \left(q^{\mathcal{D}''} \cap |\mathcal{D}|^n\right) \\
& \text{(since } \widehat{\mathcal{D}'} \text{ is a vcgd)} \\
=\ & \left(\bigcap_{\mathcal{D}'' \text{ vcgd for } (\mathcal{I},\mathcal{D})} q^{\mathcal{D}''}\right) \cap |\mathcal{D}|^n \\
=\ & (\mathrm{Cert}(q, \mathcal{I}, \mathcal{D})) \cap |\mathcal{D}|^n \\
=\ & \mathrm{Cert}(q, \mathcal{I}, \mathcal{D})
\end{aligned}
$$

With lemma 4.19, we obtain the wanted result. □

In order to characterize certain answers, we need to introduce the clausal form and the clausal form theorem. We assume clauses (cf. def. 2.1) are viewed as the finite set of their literals and thus can be noted $\{A_1, \ldots, A_n\}$ or even $A_1, \ldots, A_n$. This notation thus represents the universal closure of the disjunction of its elements.

Definition 4.21. Tensor product [7]
Let $\mathcal{A}$ and $\mathcal{A}'$ two sets of clauses. We note $\mathcal{A} \otimes \mathcal{A}'$ the set $\{A \cup A' \mid A \in \mathcal{A} \text{ and } A' \in \mathcal{A}'\}$.

Definition 4.22. Clausal form [7]
We define, for each first order formula φ, and each 0, 1-word w a clause set $S_w(\varphi)$ as follows (with α and β two first order formulas):

(1) Negations are put beside atomic formulas:
- $\mathrm{Cl}_w(\neg(\alpha \wedge \beta)) = \mathrm{Cl}_w(\neg\alpha \vee \neg\beta)$
- $\mathrm{Cl}_w(\neg(\alpha \vee \beta)) = \mathrm{Cl}_w(\neg\alpha \wedge \neg\beta)$
- $\mathrm{Cl}_w(\neg(\alpha \rightarrow \beta)) = \mathrm{Cl}_w(\alpha \wedge \neg\beta)$
- $\mathrm{Cl}_w(\neg\forall X\alpha) = \mathrm{Cl}_w(\exists X\neg\alpha)$
- $\mathrm{Cl}_w(\neg\exists X\alpha) = \mathrm{Cl}_w(\forall X\neg\alpha)$

(2) The clause set is computed by:
- $\mathrm{Cl}_w(\alpha) = \alpha$ if α is a literal.
- $\mathrm{Cl}_w(\alpha \wedge \beta) = \mathrm{Cl}_{w0}(\alpha) \cup \mathrm{Cl}_{w1}(\beta)$.
- $\mathrm{Cl}_w(\alpha \vee \beta) = \mathrm{Cl}_{w0}(\alpha) \otimes \mathrm{Cl}_{w1}(\beta)$.
- $\mathrm{Cl}_w(\alpha \rightarrow \beta) = \mathrm{Cl}_w(\neg\alpha \vee \beta)$.
- $\mathrm{Cl}_w(\forall X\alpha) = \mathrm{Cl}_w(\alpha)$
- $\mathrm{Cl}_w(\exists X\alpha) = \mathrm{Cl}_{w0}(\alpha\{X/f_w(X_1, \ldots, X_n)\})$
  where $X_1, \ldots, X_n$ are the free variables of α different from X, and f_w is a new function symbol.

The clauses set $\mathrm{Cl}_\epsilon(\varphi)$ is called the clausal form of φ and is denoted by $\mathrm{Cl}(\varphi)$.

Example 4.23. We keep on studying the running example. According to proposition 4.11, the mapping $v4 \subseteq q4$, with $q4$ defined as $q4(X) \leftarrow cites(X, Y), cites(Y, X)$, corresponds to the following first order sentence φ_1:
$\forall X\ v4(X) \rightarrow (\exists Y(cites(X, Y) \wedge cites(Y, X)))$
Let's compute the clausal form of this sentence. The negation processing step is useless here since there is no negation in this sentence. The computation of the clause set is the following:
$\mathrm{Cl}_\epsilon(\varphi_1)$
$= \mathrm{Cl}_\epsilon(\forall X\ v4(X) \rightarrow (\exists Y(cites(X, Y) \wedge cites(Y, X))))$
$= \mathrm{Cl}_\epsilon(v4(X) \rightarrow (\exists Y(cites(X, Y) \wedge cites(Y, X))))$
$= \mathrm{Cl}_\epsilon((\neg v4(X)) \vee (\exists Y(cites(X, Y) \wedge cites(Y, X))))$
$= \mathrm{Cl}_0(\neg v4(X)) \otimes \mathrm{Cl}_1(\exists Y(cites(X, Y) \wedge cites(Y, X)))$
$= \{\neg v4(X)\} \otimes$
$\mathrm{Cl}_{10}((cites(X, Y) \wedge cites(Y, X))\{Y/f_1(X)\})$
$= \{\neg v4(X)\} \otimes \mathrm{Cl}_{10}(cites(X, f_1(X)) \wedge cites(f_1(X), X))$
$= \{\neg v4(X)\} \otimes$
$(\mathrm{Cl}_{100}(cites(X, f_1(X))) \cup \mathrm{Cl}_{101}(cites(f_1(X), X)))$
$= \{\neg v4(X)\} \otimes \{cites(X, f_1(X)), cites(f_1(X), X)\}$
$= \{\{\neg v4(X), cites(X, f_1(X))\},$
$\{\neg v4(X), cites(f_1(X), X)\}\}$
This set of clauses is quivalent to the following couple of definite clauses:
$\{cites(X, f_1(X)) \leftarrow v4(X)\ ,\ cites(f_1(X), X) \leftarrow v4(X)\}$.

Theorem 4.24. Clausal Form Theorem [7]
Let Γ be a set of $\mathcal{L}$-formulas. Let φ be an $\mathcal{L}$-formula. Then $\Gamma \models \varphi \Leftrightarrow \mathrm{Cl}(\bigwedge \Gamma) \models \varphi$

We can now give the certain answers characterization, expressed in terms of the definite program that models the DIS.

Definition 4.25. Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system such that the mappings in $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ are LAV mappings assuming OWA. Let $\mathcal{D}$ a be a source database for $\mathcal{I}$. The *definite positive program associated to* $(\mathcal{I}, \mathcal{D})$, denoted by $P(\mathcal{I}, \mathcal{D})$, is defined by:
$P(\mathcal{I}, \mathcal{D}) = \mathrm{Cl}\left(\left(\bigwedge \mathcal{D}_{def}\right) \wedge \left(\bigwedge \mathcal{M}'_{\mathcal{G},\mathcal{S}}\right)\right)$

Theorem 4.26. Certain answers characterization
Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system such that the mappings in $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ are LAV mappings assuming OWA. Let $\mathcal{D}$ be a source database for $\mathcal{I}$. Let q be a conjunctive query on $\mathcal{G}$ associated to a definite query $Q\left(\overline{X}; \overline{Z}\right)$. We have:

$$\mathrm{GQAT}\left(Q\left(\overline{X}; \overline{Z}\right), P(\mathcal{I}, \mathcal{D})\right) = \mathrm{Cert}(q, \mathcal{I}, \mathcal{D})\,.$$

Proof. We assume the notations given in definitions 4.7 and 4.12. We suppose that n is the arity of q, $\mathcal{M}_{\mathcal{G},\mathcal{S}} = \{m_1, \ldots, m_r\}$ and thus $\mathcal{M}'_{\mathcal{G},\mathcal{S}} = \{m'_1, \ldots, m'_r\}$, and $\mathcal{L}^H$ is a Herbrand language for $(\mathcal{I}, \mathcal{D})$. We have:
$\mathrm{Cert}(q, \mathcal{I}, \mathcal{D})$
$\overset{Theo.4.20}{=} \mathrm{Cert}^H(q, \mathcal{I}, \mathcal{D})$
$\overset{Lem.4.3\ and\ 4.19}{=} \mathcal{N}(\mathrm{Cert}^H(q, \mathcal{I}, \mathcal{D}))$
$\overset{Def.4.18}{=} \bigcap_{\mathcal{D}' \text{ vgi for } (\mathcal{I},\mathcal{D})} \mathcal{N}(q^{\mathcal{D}'})$

$$\overset{Theo.3.15}{=} \bigcap_{\mathcal{D}' \text{ vgi for } (\mathcal{I},\mathcal{D})} \text{GQAT}\left(\mathcal{Q}\left(\overline{X};\overline{Z}\right),(\mathcal{D}')_{def}\right)$$

$$\overset{Def.2.6}{=} \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid \text{for each } \mathcal{D}' \text{ vgi for } (\mathcal{I},\mathcal{D}), \text{there exists a ground substitution } \sigma \text{ s.t. } \overline{\mathcal{T}} = \overline{X}\sigma \text{ and } \mathcal{D}'_{def} \models (Body(\mathcal{Q}\left(\overline{X};\overline{Z}\right))\sigma\}$$

$$= \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid \text{for each } \mathcal{D}' \text{ vgi for } (\mathcal{I},\mathcal{D}), \text{there exists a ground substitution } \sigma \text{ s.t. } \overline{\mathcal{T}} = \overline{X}\sigma \text{ and } \mathcal{D}'_{def} \models (\text{Body}(\mathcal{Q}\left(\overline{\mathcal{T}};\overline{Z}\sigma\right)))\}$$

$$\overset{Corol.3.11}{=} \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid \text{for each } \mathcal{D}' \text{ vgi for } (\mathcal{I},\mathcal{D}), \text{there exists a ground substitution } \sigma \text{ s.t. } \overline{\mathcal{T}} = \overline{X}\sigma \text{ and } \mathcal{D}' \models (\text{Body}(\mathcal{Q}\left(\overline{\mathcal{T}};\overline{Z}\sigma\right)))\}$$

$$= \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid \text{for each } \mathcal{D}' \text{ vgi for } (\mathcal{I},\mathcal{D}), \mathcal{D}' \models \exists\overline{Z}(\text{Body}(\mathcal{Q}\left(\overline{\mathcal{T}};\overline{Z}\right)))\}$$

$$= \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid \text{for all } \mathcal{L}^H\text{-Herbrand model } \mathcal{D}' \text{ if } \mathcal{D}' \text{ is a vgi for } (\mathcal{I},\mathcal{D}), \text{then } \mathcal{D}' \models \exists\overline{Z}(\text{Body}(\mathcal{Q}\left(\overline{\mathcal{T}};\overline{Z}\right)))\}$$

$$\overset{Lem.4.15}{=} \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid \text{for all } \mathcal{L}^H\text{-Herbrand model } \mathcal{D}' \text{ if } \mathcal{D}' \models \wedge\left(\mathcal{D}_{def} \cup \{m'_1,\ldots,m'_r\}\right) \text{ then } \mathcal{D}' \models \exists\overline{Z}(\text{Body}(\mathcal{Q}\left(\overline{\mathcal{T}};\overline{Z}\right)))\}$$

$$= \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid \text{for all } \mathcal{L}^H\text{-Herbrand model } \mathcal{D}' \; \mathcal{D}' \models \wedge\left(\mathcal{D}_{def} \cup \{m'_1,\ldots,m'_r\}\right) \rightarrow \exists\overline{Z}\,\text{Body}(\mathcal{Q}(\overline{\mathcal{T}};\overline{Z}))\}$$

$$\overset{Theo.4.5}{=} \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid \models \wedge\left(\mathcal{D}_{def} \cup \{m'_1,\ldots,m'_r\}\right) \rightarrow \exists\overline{Z}\,\text{Body}(\mathcal{Q}(\overline{\mathcal{T}};\overline{Z}))\}$$

$$= \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid \mathcal{D}_{def} \cup \{m'_1,\ldots,m'_r\} \models \exists\overline{Z}\,\text{Body}(\mathcal{Q}(\overline{\mathcal{T}};\overline{Z}))\}$$

$$\overset{Theo.4.24}{=} \{\overline{\mathcal{T}} \in \mathcal{U}^n \mid P(\mathcal{I},\mathcal{D}) \models \exists\overline{Z}\,\text{Body}(\mathcal{Q}(\overline{\mathcal{T}};\overline{Z}))\}$$

$$\overset{Def.2.6}{=} \text{GQAT}(\mathcal{Q}\left(\overline{X};\overline{Z}\right),P(\mathcal{I},\mathcal{D}))$$

□

5 CONCLUSION AND FUTURE WORKS

This papers revisits some classical notions and results from logic programming, relational databases and data integration, from the answer tuples point of view. This enables to give a new compactness theorem concerning certain answers and then a new characterization of these certain answers in terms of the answers of a definite query w.r.t a definite program.

This work is a preliminary work for an on-going study about query rewriting algorithms in data integration. Indeed, since it justifies the use of SLD-resolution in this area, then other classical algorithms, like for instance the bucket [15], the inverse-rules [8] or the minicon [17] algorithms, might be valuably compared to SLD-resolution. The deeper and synthetic understanding of these algorithms that could emerge from this study may also be applied thereafter on other query rewriting algorithms, like those in the context of ontological query answering [11].

REFERENCES

[1] S. Abiteboul and O. Duschka. 1998. Complexity of answering queries using materialized views. *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98)* (1998), 254–265.

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

[3] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, and Pierre Senellart. 2012. *Web Data Management.* Cambridge University Press. 432 pages. http://hal.inria.fr/hal-00847933 Open access of the full text on the Web.

[4] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 2002. Data Integration under Integrity Constraints. In *Proc. of the 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2002) (Lecture Notes in Computer Science)*, Vol. 2348. Springer, 262–279.

[5] Andrea Cali, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 2002. Data Integration under Integrity Constraints. In *Advanced Information Systems Engineering*, AnneBanks Pidduck, M.Tamer Ozsu, John Mylopoulos, and CarsonC. Woo (Eds.). Lecture Notes in Computer Science, Vol. 2348. Springer Berlin Heidelberg, 262–279. https://doi.org/10.1007/3-540-47961-9_20

[6] Andrea Calì, Diego Calvanese, and Maurizio Lenzerini. 2013. Data Integration under Integrity Constraints. In *Seminal Contributions to Information Systems Engineering*, Janis A. Bubenko Jr., John Krogstie, Oscar Pastor, Barbara Pernici, Colette Rolland, and Arne Solvberg (Eds.). Springer, 335–352.

[7] R. David, K. Nour, and C. Raffalli. 2004. *Introduction à la logique: théorie de la démonstration : cours et exercices corrigés.* Dunod.

[8] O.M. Duschka. 1997. Query Optimization Using Local Completeness. In *Proceedings of the Fourteenth AAAI National Conference on Artificial Intelligence, AAAI-97.*

[9] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: semantics and query answering. *Theoretical Computer Science* 336, 1 (2005), 89 – 124. Database Theory.

[10] M. Fitting. 1990. *First-order Logic and Automated Theorem Proving.* Springer Verlag, Berlin, Germany. 326 pages.

[11] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. 2011. Ontological Query Answering via Rewriting. In *Proc. of 15th East-European Conf. on Advances in Databases and Information Systems (ADBIS).* 1–18.

[12] John Grant and Jack Minker. 2002. A Logic-based Approach to Data Integration. *Theory Pract. Log. Program.* 2, 3 (May 2002), 323–368. https://doi.org/10.1017/S1471068401001375

[13] A.Y. Halevy. 2001. Answering Queries Using Views: A Survey. *VLDB Journal* (2001).

[14] Christoph Koch. 2004. Query rewriting with symmetric constraints. *AI Commun.* 17, 2 (2004), 41–55.

[15] A. Levy, A. Rajaraman, and J. Ordille. 1996. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proceedings of the 22nd VLDB Conference, Bombay, India*, T. M. Vijayaraman, Alejandro P. Buchmann, C. Mohan, and Nandlal L. Sarda (Eds.). Morgan Kaufmann, 251–262.

[16] Ulf Nilsson and Jan Maluszynski. 1995. *Logic, Programming, and PROLOG* (2nd ed.). John Wiley & Sons, Inc., New York, NY, USA.

[17] R. Pottinger and A. Halevy. 2001. MiniCon: A scalable algorithm for answering queries using views. *The VLDB Journal* 10, 2-3 (2001), 182–198.

A PROOFS

OF LEMMA 3.9.

(1) This comes directly from definition 3.2 and FOL semantics.
(2) Let $\overline{X}$ be the tuple of free variables of $\exists\overline{Z}(\psi_1 \wedge \ldots \wedge \psi_m)$. There is : $\left(\mathcal{D} \models_v \exists\overline{Z}(\psi_1 \wedge \ldots \wedge \psi_m)\right)$

$\Leftrightarrow$ there is a valuation v' which coincides with v on $\overline{X}$ s.t. $(\mathcal{D} \models_{v'} (\psi_1 \wedge \ldots \wedge \psi_m))$;
$\Leftrightarrow$ there is a valuation v' which coincides with v on $\overline{X}$ s.t. $(\mathcal{D}^v \models_{v'} (\psi_1 \wedge \ldots \wedge \psi_m))$;
$\Leftrightarrow \left(\mathcal{D}^v \models_v \exists\overline{Z}(\psi_1 \wedge \ldots \wedge \psi_m)\right)$.

□

of lemma 3.10.

(1) This equivalence is direct since a new ground atomic formula cannot be inferred from a set of ground atomic formulas.
(2) Here we suppose, w.l.o.g. concerning the order of variables and constants, that φ is the following atomic formula: $r(X_1, \ldots, X_{p_1}, c'_1, \ldots, c'_{p_2})$ with $r \in \mathcal{R}, \forall i \in \{1, \ldots, p_1\}$, $X_i \in Var$ and $\forall j \in \{1, \ldots, p_2\}$, $c'_j \in C^v$. For every $c' \in C^v$ let $c \in |\mathcal{D}|$ be the interpretation ${c'}^{\mathcal{D}^v}$ of c'; notice that $N(c', c)$. Moreover, $\forall i \in \{1, \ldots, p_1\}$, $N(X_i v', v(X_i))$. Hence
$< v(X_1), \ldots, v(X_{p_1}), c_1, \ldots, c_{p_2} > \in r^{\mathcal{D}^v}$
$\Leftrightarrow r(X_1 v', \ldots, X_{p_1} v', c'_1, \ldots, c'_{p_2}) \in \mathcal{D}_{def}$
(3) By the semantics of conjunction.
(4) Let $\overline{X}$ be the free variables tuple of $(\exists\overline{Z}\,(\varphi_1 \wedge \ldots \wedge \varphi_n))$. We have:
$\mathcal{D}^v \models_v (\exists\overline{Z}\,(\varphi_1 \wedge \ldots \wedge \varphi_n))$
  - $\Leftrightarrow$ There is a valuation v_1 such that $\mathcal{D}^v \models_{v_1} \varphi_1 \wedge \ldots \wedge \varphi_n$ and such that v and v_1 coincide for variables in $\overline{X}$.
  - $\Leftrightarrow$ There is a name substitution v'_1 for v_1 such that $\mathcal{D}_{def} \models (\varphi_1 \wedge \ldots \wedge \varphi_n)\, v'_1$ and such that v' and v'_1 coincide for variables in $\overline{X}$;
  - $\Leftrightarrow$ There is a ground substitution v'_1 such that $\mathcal{D}_{def} \models (\varphi_1 \wedge \ldots \wedge \varphi_n)\, v'_1$ and such that v' and v'_1 coincide for variables in $\overline{X}$.
  - $\Leftrightarrow \mathcal{D}_{def} \models \left(\exists\overline{Z}\,(\varphi_1 \wedge \ldots \wedge \varphi_n)\right) v'$

We give some details for one direction of the last equivalence. If $\left(\mathcal{D}_{def} \models (\exists\overline{Z}(\varphi_1 \wedge \ldots \wedge \varphi_n))v'\right)$ then, by compactness Theorem, there is a finite subset $\Phi \subseteq \mathcal{D}_{def}$ such that $\left(\Phi \models (\exists\overline{Z}(\varphi_1 \wedge \ldots \wedge \varphi_n))v'\right)$. Hence, by Herbrand Theorem, there is a ground substitution v'_1 such that $\Phi \models (\varphi_1 \wedge \ldots \wedge \varphi_n)\, v'_1$ and such that v' and v'_1 coincide for variables in $\overline{X}$. □

of theorem 3.15.

Let $Body\left(Q\left(\overline{X}; \overline{Z}\right)\right) = A_1 \wedge \ldots \wedge A_l$, with $\overline{X} = \langle X_1, \ldots, X_n \rangle$. We have:
$\mathrm{GQAT}(Q\left(\overline{X}; \overline{Z}\right), \mathcal{D}_{def})$
by def. 2.6
$= \{\overline{\mathcal{T}} \in \mathrm{QAT}(Q\left(\overline{X}; \overline{Z}\right), \mathcal{D}_{def}) \mid \overline{\mathcal{T}} \text{ is ground}\}$
by def. 2.5
$= \{\overline{X}\sigma | \sigma \in \mathrm{AS}(Q\left(\overline{X}; \overline{Z}\right), \mathcal{D}_{def}) \text{ and } \overline{X}\sigma \text{ is ground}\}$
by def. 2.3
$= \{\overline{X}\sigma \mid \sigma \in \mathrm{TS} \text{ and } \mathcal{D}_{def} \models (A_1 \wedge \ldots \wedge A_l)\sigma \text{ and } \overline{X}\sigma \text{ is ground}\}$
Since the FO language used to build $\mathcal{D}_{def}$ is $(\mathcal{R}, C^v)$ and lemma 3.10 says $(\mathcal{D}_{def} \models A_i\sigma) \Leftrightarrow (A_i\sigma \in \mathcal{D}_{def})$ for any A_i and σ, we have:
$= \{\overline{X}\sigma \in (C^v)^n \mid \sigma \in \mathrm{GTS} \text{ and } \mathcal{D}_{def} \models (A_1 \wedge \ldots \wedge A_l)\sigma\}$
$= \{\overline{X}\sigma = \langle c'_1, \ldots, c'_n \rangle \in (C^v)^n \mid \sigma \in \mathrm{GTS} \text{ and } \sigma{\restriction}_{\overline{X}} = \{X_1/c'_1, \ldots, X_n/c'_n\} = \sigma' \text{ and } \mathcal{D}_{def} \models (A_1 \wedge \ldots \wedge A_l)\sigma\}$
$= \{\langle c'_1, \ldots, c'_n \rangle \in (C^v)^n \mid (\exists\sigma \in \mathrm{GTS} \text{ such that } \sigma{\restriction}_{\overline{X}} = \{X_1/c'_1, \ldots, X_n/c'_n\} = \sigma' \text{ and } \mathcal{D}_{def} \models \left(\exists\overline{Z}(A_1 \wedge \ldots \wedge A_l)\right)\sigma')$
Since, by construction of $\mathcal{D}^v$, $\forall c'_i \in C^v\ \exists c_i \in |\mathcal{D}^v| \mid N(c'_i, c_i)$, then we have:
$= \{\langle c'_1, \ldots, c'_n \rangle \in (C^v)^n \mid (\exists\sigma \in \mathrm{GTS} \text{ such that } \sigma{\restriction}_{\overline{X}} = \{X_1/c'_1, \ldots, X_n/c'_n\} = \sigma' \text{ and } \mathcal{D}_{def} \models \left(\exists\overline{Z}(A_1 \wedge \ldots \wedge A_l)\right)\sigma') \text{ and } \exists \langle c_1, \ldots, c_n \rangle \in (|\mathcal{D}^v|)^n \text{ s.t. } N(c'_1, c_1), \ldots, N(c'_n, c_n)\}$
since $|\mathcal{D}^v| = |\mathcal{D}|$.
$= \{\langle c'_1, \ldots, c'_n \rangle \in (C^v)^n \mid (\exists\sigma \in \mathrm{GTS} \text{ such that } \sigma{\restriction}_{\overline{X}} = \{X_1/c'_1, \ldots, X_n/c'_n\} = \sigma' \text{ and } \mathcal{D}_{def} \models \left(\exists\overline{Z}(A_1 \wedge \ldots \wedge A_l)\right)\sigma') \text{ and } \exists \langle c_1, \ldots, c_n \rangle \in (|\mathcal{D}|)^n \text{ s.t. } N(c'_1, c_1), \ldots, N(c'_n, c_n)\}$
by lem. 3.10
$= \{\langle c'_1, \ldots, c'_n \rangle \in (C^v)^n \mid \exists \langle c_1, \ldots, c_n \rangle \in (|\mathcal{D}|)^n \text{ s.t. } \mathcal{D} \models_{\{X_1/c_1, \ldots, X_n/c_n\}} \exists\overline{Z}(A_1 \wedge \ldots \wedge A_l) \text{ and } N(c'_1, c_1), \ldots, N(c'_n, c_n)\}$
by def. 3.14
$= \mathcal{N}\left(q^{\mathcal{D}}\right)$

□

of lemma 3.17. Let n be the arity of q_1 and q_2.
$q_1^{\mathcal{D}} \subseteq q_2^{\mathcal{D}}$
by def. 3.14
$\Leftrightarrow \{\langle c_1, \ldots, c_n \rangle \in |\mathcal{D}|^n \mid \mathcal{D} \models_{\{X_1/c_1, \ldots, X_n/c_n\}} \exists\overline{Z}\left(Body(Q_1\left(\overline{X}; \overline{Z}\right))\right)\} \subseteq \{\langle d_1, \ldots, d_n \rangle \in |\mathcal{D}|^n \mid \mathcal{D} \models_{\{W_1/d_1, \ldots, W_n/d_n\}} \exists\overline{V}\left(Body(Q_2\left(\overline{W}; \overline{V}\right))\right)\}$

$\Leftrightarrow$ for all $\langle c_1, \ldots, c_n \rangle \in |\mathcal{D}|^n$
if $\mathcal{D} \models_{\{X_1/c_1,\ldots,X_n/c_n\}} \exists \overline{Z} \left(Body(Q_1 \left(\overline{X}; \overline{Z} \right)) \right)$
then $\mathcal{D} \models_{\{X_1/c_1,\ldots,X_n/c_n\}} \exists \overline{V} \left(Body(Q_2 \left(\overline{X}; \overline{V} \right)) \right)$

$\Leftrightarrow$ for all $\langle c_1, \ldots, c_n \rangle \in |\mathcal{D}|^n$
$\mathcal{D} \models_{\{X_1/c_1,\ldots,X_n/c_n\}} \exists \overline{Z} \left(Body(Q_1 \left(\overline{X}; \overline{Z} \right)) \right) \rightarrow$
$\exists \overline{V} \left(Body(Q_2 \left(\overline{X}; \overline{V} \right)) \right)$

$\Leftrightarrow$ $\mathcal{D} \models \forall \overline{X} (\exists \overline{Z} Body \left(Q_1 \left(\overline{X}; \overline{Z} \right) \right)) \rightarrow$
$(\exists \overline{V} Body \left(Q_2 \left(\overline{X}; \overline{V} \right) \right))$

□