Determining Source Contribution in Integration Systems

Alin Deutsch* Yannis Katsis† Yannis Papakonstantinou

Computer Science and Engineering, University of California, San Diego
{deutsch,ikatsis,yannis}@cs.ucsd.edu

ABSTRACT

Owners of sources registered in an information integration system, which provides answers to a (potentially evolving) set of client queries, need to know their contribution to the query results. We study the problem of deciding, given a client query Q and a source registration R, whether R is (i) "self-sufficient" (can contribute to the result of Q even if it is the only source in the system) or (ii) "now complementary" (can contribute, but only in cooperation with other specific existing sources), or (iii)"later complementary" (can contribute if in the future appropriate new sources join the system). We consider open-world integration systems in which registrations are expressed using source-to-target constraints, and queries are answered under "certain answer" semantics.

1. INTRODUCTION

Mediator systems allow their client applications and users to obtain information from multiple sources using a single point of access, which is an integrated view of the data of the sources. For example, consider a car shopping portal. Multiple car dealers contribute advertisements, while at the same time third parties provide reviews, such as the "Blue Book" (`http://www.kbb.com`). Then one can easily build the, say, "Great Auto Deal" web application that looks for cars of a user-provided make and type, selling for less than 10% of their Blue Book value. Similarly, many biology labs contribute to portals such as Nature magazine's SignallingGateway and the Gene Ontology (GO). Other biology labs and researchers periodically search the portals for proteins or genes related to a particular species and physiological aspects that the lab studies.

We adopt the GLAV approach to integration [13], which fits best our focus, namely mediators and corresponding integration portals that collect information from a large number of independent sources. In GLAV integration, there is (a) a number of sources (also referred to as local databases) with different *local schemas* that hold the actual data and (b) a global database over a *global schema* that hides the underlying local schemas from the user. A description of the correspondence between the local and the global schema can be specified using mappings from the local to the global schema. An intuitive tool for visual mapping specification is provided by CLIO's front-end [18, 21, 11]. For example, Figure 2 shows the mapping of the local source of schema $\mathcal{S}_1$ into the target global schema $\mathcal{G}$. Technically, mappings are expressed as constraints and the query answers are the certain answers, as defined in [20, 13, 14, 17].

A recognized key advantage of the GLAV approach is that the correspondence of a local schema to the global schema can be specified independently of the other sources, hence enabling large scale systems where the source owners themselves register their sources to community portals.

However, when a source owner wants to register a new source, he needs to understand whether his source addresses the information requirements of the queries issued by client applications and users. An overkill (if at all possible) way to ensure this is to force the source owner to map some data into every attribute of the global schema. This approach may be simply impossible because a source owner may not possess data for some parts of the global schema; for example, a car dealer owns ads but not the Blue Book. Even when it is possible, it may be economically unwise to provide data for all attributes. In particular, global schemas are typically developed by communities and are very expansive and detailed[1]. If our car dealer owns textual ads, he may not want to painfully clean them up and parse them in order to specify all attributes (including trivial ones) of an ad, such as date of battery change. Instead, he may prefer to clean up only attributes that are pertinent to the existing client queries, such as the make, price and year of the car, and leave the rest in an unstructured "additional data" field.

In order to guide the source owner, mapping interfaces must explain to him how to contribute towards the application query information requirements. The first step in developing such systems is to characterize the contribution of a given mapping.

We distinguish four qualitative categories of contributions of a source schema/mapping pair with respect to a given application query:

- *Self Sufficient*: Given the mapping, the source can contribute certain answers to the query, even in the absence of any other source. For example, consider a query that asks for cheap cars, regardless of how their value compares to the Blue Book. Then a car dealer's contribution is self-sufficient, assuming it provides the price and the other attributes of interest to the query.
- *Now Complementary*: The source can contribute certain answers but only because other sources provide complementary

*Supported by NSF/CAREER award IIS–0347968.
†Supported by NSF/ITR grant IIS–0427196.

PODS 2005 June 13-15, 2005, Baltimore, Maryland.

[1]The reader is referred to the DTDs and XML Schemas of OASIS `www.oasis-open.org` for an example.

$\Delta_{\mathcal{G}}$	none	PKs	PKs + RICs
suff.	yes	yes	no
now comp.	yes	yes	no
later comp.	yes	yes	?
unusable	yes	yes	?

Figure 1: Decidability of categorizing registrations (see Section 5 for complexity results)

data. For example, consider a query that asks for cars that are 10% cheaper than their Blue Book value and assume that the Blue Book is already registered. Then the car dealer's source is now complementary since it relies on an existing source in order to contribute.

- *Later Complementary*: The source cannot contribute certain answers currently but it is possible to contribute if an appropriate new source is registered in the future. For example, consider the case of "now complementary" above whereas a source such as the Blue Book, providing typical prices, has not yet been registered. Then the dealer's source is later complementary.

- *Unusable*: The source inherently does not contribute.

Contributions. We study the problem of deciding the category of source contributions when the mappings are given by source-to-target constraints in the language of embedded dependencies, and the client queries are unions of conjunctive queries with equalities. Figure 1 summarizes our decidability results. Each row corresponds to one of the contribution tests, while columns correspond to various assumptions on the set $\Delta_{\mathcal{G}}$ of constraints on the global schema $\mathcal{G}$. PKs stands for primary keys and RICs for referential integrity constraints, all of which are expressible in the more general class of embedded dependencies [2]. Our undecidability results therefore carry over to this class. Question marks denote open problems, which we conjecture undecidable.

Our decidability results for self-sufficiency and now-complementarity are due to a reduction to checking satisfiability and containment of a *certain rewriting* R of a client query Q. R is expressed against the local schemas and it returns the certain answers of Q. Developing a relevant containment test is not straightforward. Previous work [10] describes only algorithms which yield a certain rewriting as a recursive Datalog program when functional dependencies (FDs) are given on the global schema. These algorithms do not serve our purpose, as containment of recursive Datalog programs is undecidable [2]. However, we show (in Theorem 8) that, if each relation in the target schema allows at most one FD,[2] then there exists an equivalent, nonrecursive rewriting expressed as a union of conjunctive queries with equalities (UCQ$^{=}$). This case is quite common in practice, covering that of primary keys and BCNF schemas. Moreover, we show how the nonrecursive certain rewriting under the Open World Assumption can be constructed by reusing an algorithm for finding exact rewritings under the Closed World Assumption. Our result is therefore of independent interest as an extension on prior work in finding certain rewritings, and also *maximally-contained rewritings* [10] shown in [1] to be equivalent in our setting to certain rewritings. The result sheds additional light on the connection between exact rewritings in the closed world assumption and certain rewritings in the open world assumption.

[2]This includes the case of several FDs which can be summarized into a single one using Armstrong's axioms [2].

2. PRELIMINARIES

Constraint-based Data Integration. We consider systems which integrate a collection of n data sources, where for each $1 \leq i \leq n$, source i has a schema S_i and extent DB_i. The local sources contribute to a global, integrated database G of global schema $\mathcal{G}$, satisfying the set $\Delta_{\mathcal{G}}$ of integrity constraints expressed in terms of $\mathcal{G}$ (denoted $G \models \Delta_{\mathcal{G}}$). The contribution of source i to the global database is described by a set M_i of *mapping constraints* over the combined schemas $\mathcal{S}_i$ and $\mathcal{G}$. Specifically, the constraints are of the form $U \subseteq V$, with U and V queries against schema $\mathcal{S}_i$, respectively $\mathcal{G}$. Intuitively, these constraints specify that, given a local database DB_i and a global database G, the local data identified as $U(DB_i)$ is visible among the global data identified by $V(G)$: $U(DB_i) \subseteq V(G)$. Notice that there are no containment statements in the opposite direction because an individual local source owner cannot know what other sources contribute to the global database and therefore cannot presume to contribute all global data. For instance, a local Toyota dealership's data source may contribute its cars to a state-wide car database, but cannot claim to offer all globally accessible car offers (including other brands). This is consistent with the *open world assumption* [14, 17].

The global database is described indirectly as a database G which satisfies the integrity constraints in $\Delta_{\mathcal{G}}$. Moreover, together with each local source DB_i, G satisfies the mapping constraints in M_i. This is denoted $DB_i, G \models M_i$ and defined as:

$$DB_i, G \models M_i \Leftrightarrow \bigwedge_{(U \subseteq V) \in M_i} U(DB_i) \subseteq V(G).$$

Given DB_i, M_i does not fully identify G, as it only states that G must hold part of DB_i's data, leaving unspecified what else G may contain. There are therefore (potentially infinitely) many possible global databases which, together with DB_i, satisfy M_i. We think of M_i as inducing a mapping from each data source DB_i to its set of possible global databases. We denote the set of targets of DB_i through this mapping as:

$$targets_{M_i}(DB_i) = \{G : (G \models \Delta_{\mathcal{G}}) \wedge (DB_i, G \models M_i)\}.$$

The set of possible global databases defined by a collection of sources $\overline{DB} = DB_1, \ldots, DB_n$ and their sets of mapping constraints $\bar{M} = M_1, \ldots, M_n$ consists of the global databases which are simultaneously targets of each DB_i under M_i:

$$targets_{\bar{M}}(\overline{DB}) = \bigcap_{i=1}^{n} targets_{M_i}(DB_i).$$

Since the set of possible global databases can potentially be infinite, clients can only inspect them indirectly, by posing queries against them. Given a client query Q against the global schema $\mathcal{G}$, the system returns only the *certain* answers to Q. These are defined as tuples appearing in the result of Q on each possible global database. We denote the set of certain answers to Q with $cert^{\bar{M}}_{\overline{DB}}(Q)$ and define it as:

$$cert^{\bar{M}}_{\overline{DB}}(Q) = \begin{cases} \emptyset, & \text{if } targets_{\bar{M}}(\overline{DB}) = \emptyset \\ \bigcap_{G \in targets_{\bar{M}}(\overline{DB})} Q(G), & \text{otherwise} \end{cases}$$

For an existing open-world system in which source contributions are described using mapping constraints (called *source-to-target* constraints there) and in which client queries are answered under the "certain answers" semantics, see IBM's Clio system [18].

Queries. A *term* is a variable or constant. By $\bar{x}$ we denote a finite sequence of terms $x_1, \ldots, x_k$. The language of conjunctive

queries with equalities (CQ$^=$) consists of expressions of the form $Q(\bar{z})$:- $\ell_1(\bar{x}_1), \ldots, \ell_n(\bar{x}_n)$ where we define the *head* and *body* of Q to be the parts to the left and to the right of the :- , respectively. Each $\ell_i(\bar{x}_i)$ in Q's body is a *literal*, i.e., an atom $R(\bar{x}_i)$ or an equality $x_g = x_h$ with $\bar{x}_i = x_g, x_h$. The language UCQ$^=$ denotes all unions of CQ$^=$ queries.

Constraints. For a given query language $\mathcal{QL}$, we consider the corresponding constraint language

$$\text{IC}(\mathcal{QL}) := \{(U \subseteq V) : U, V \in \mathcal{QL}\}.$$

Such constraints express the containment of query U in query V and are equivalent in expressive power to *embedded dependencies* [2] when $\mathcal{QL} = \text{CQ}^=$.

Embedded dependencies can express standard integrity constraints such as primary keys (PKs) and referential integrity constraints (RICs). For example, the following IC(CQ$^=$) constraint on the global schema of Figure 2 states that Model and Seller provide a combined PK for ads in the ad relation: $(U \subseteq V)$, where

$V(M, S, P, P)$:- $ad(M, S, P)$ and

$U(M, S, P_1, P_2)$:- $ad(M, S, P_1), ad(M, S, P_2)$.

RICs are particular cases of IC(CQ$^=$) constraints stating inclusions between projections of relations. For example, the following RIC states that all ads refer via their Model attribute to cars whose details are published in the car relation: $(U \subseteq V)$, where $U(M)$:- $ad(M, S, P)$ and $V(M)$:- $car(M, C, D, B)$.

In this paper, we assume that all client queries belong to UCQ$^=$ and that all constraints belong to IC(CQ$^=$). This is the case in all our examples as well.

3. DEGREES OF SOURCE CONTRIBUTION

We introduce the notion of contribution of a source's extent to a given client query, as well as four qualitative degrees of potential contribution of a source registration relative to the other sources registered in the system.

Each individual source owner registers his local source i to the integration system by declaring the source schema $\mathcal{S}_i$ as well as the set of mapping constraints M_i. We call the pair $R_i = (\mathcal{S}_i, M_i)$ a *source registration.* Let $\overline{R} = R_1, ..., R_n$ be the list of registrations of all sources present within the system and $\overline{D} = D_1, ..., D_n$ the corresponding source instances. Also, let Q be a query formulated by a client application against global schema $\mathcal{G}$.

Contribution of a source to a query. If $R_{n+1} = (\mathcal{S}_{n+1}, M_{n+1})$ is the registration of a new source and D_{n+1} an instance of it, then the contribution of the new source to Q is denoted $contr^Q_{\overline{R},\overline{D}}(R_{n+1}, D_{n+1})$ and corresponds to the certain answers to Q which are not provided by the already registered n sources:

$$contr^Q_{\overline{R},\overline{D}}(R_{n+1}, D_{n+1}) = cert^{\overline{M},M_{n+1}}_{\overline{D},D_{n+1}}(Q) \setminus cert^{\overline{M}}_{\overline{D}}(Q)$$

Potential contribution of a registration. For the benefit of the source owner, we determine whether the source has any chance to contribute (now or in the future) any certain answers to some given application query Q. The definitions are data independent by existentially quantifying the source extents. Specifically, we say that source registration R_{n+1} is

- **Self Sufficient** (written as $Suf_Q(R_{n+1})$)
  iff there is an extent of source $n+1$ (not necessarily the current one) for which it can contribute certain answers to Q even in the absence of any other source. Formally, iff there is a source extent D_{n+1} such that $contr^Q_{\emptyset,\emptyset}(R_{n+1}, D_{n+1}) \neq \emptyset$.

- **Now Complementary** ($NComp^Q_{\overline{R}}(R_{n+1})$)
  iff source $n+1$ on its own brings an empty contribution regardless of its extent, but it could contribute in cooperation with the other sources, provided appropriate extents for them and for source $n+1$:
  - not $Suf_Q(R_{n+1})$ and
  - there are source extents $\{D_i\}_{1 \leq i \leq n+1}$ such that $contr^Q_{\overline{R},D_1,\ldots,D_n}(R_{n+1}, D_{n+1}) \neq \emptyset$

- **Later Complementary** ($LComp^Q_{\overline{R}}(R_{n+1})$)
  iff source $n+1$ is currently not now complementary given the registrations $\overline{R}$, but could become so given future source registrations $\overline{R'}$:
  - not $NComp^Q_{\overline{R}}(R_{n+1})$ and
  - there is a list of registrations $\overline{R'}$, s.t. $NComp^Q_{\overline{R'}}(R_{n+1})$

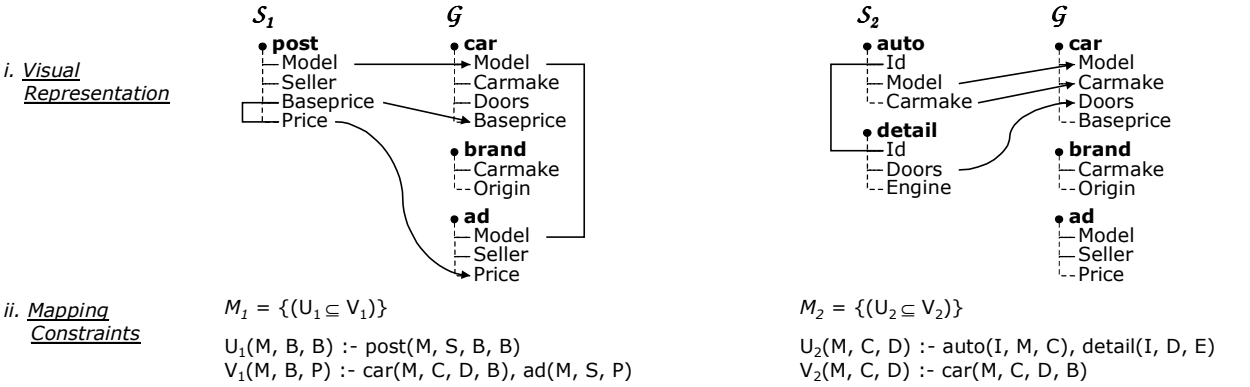
- **Unusable** ($Unusable^Q_{\overline{R}}(R_{n+1})$)
  iff source $n+1$ cannot contribute now (even if its own extent or the extents of the existing sources were to change) or in the future (if any other sources register within the system):
  - not $Suf_Q(R_{n+1})$ and
  - not $NComp^Q_{\overline{R}}(R_{n+1})$ and
  - not $LComp^Q_{\overline{R}}(R_{n+1})$

4. EXAMPLES

In this section we introduce our running example and use it to illustrate the degrees of contribution in both the absence and presence of target constraints.

Consider the creation of a car portal that integrates information about new cars for sale. The global schema $\mathcal{G}$, designed by the portal owner, is shown on the right in Figures 2a-i and 2b-i. The left side contains the schema $\mathcal{S}_1$ of the already registered source 1 and respectively, $\mathcal{S}_2$ of a new source 2, whose contribution we want to check. All schemas are shown in a tree-format. Bullets represent relations, hyphens correspond to attributes and dashed lines connect a relation with its attributes. For example, $\mathcal{G}$ consists of 3 relations: car contains the model, make, door number and base price of a car, brand provides the headquarters' location of a car manufacturer and ad stores the price at which somebody sells a model.

The registration R_i of each source i consists by a (for simplicity) *single* mapping constraint $(U_i \subseteq V_i)$ over $\mathcal{S}_i$ and $\mathcal{G}$, printed in Figures 2a-ii and 2b-ii and represented in 2a-i and 2b-i as a set of solid lines and arrows. The atoms of query U_i (respectively V_i) are all relations of $\mathcal{S}_i(\mathcal{G})$, that contain an attribute adjacent to an arrow or solid line. The equalities in U_i (V_i) are depicted as solid lines between attributes in $\mathcal{S}_i$ ($\mathcal{G}$) and the distinguished variables of U_i (V_i) correspond to attributes serving as arrow sources (targets). For example, source 1 provides ad prices, base prices and model descriptions for ads having the same ad and base price. Similarly source 2 provides the model, make and number of doors of some cars.



a. Registration $R_1 = (S_1, M_1)$ of *existing* source 1 **b. Registration $R_2 = (S_2, M_2)$ of *new* source 2**

Figure 2: Source Mappings

Note that differences between the source and global schema can be easily handled by the mapping language. For instance, by equating the values of Model in relations `car` and `ad`, $(U_1 \subseteq V_1)$ specifies that the mapped ad and base price correspond to the *same* model. Similarly $(U_2 \subseteq V_2)$ maps only source values that appear in pairs of tuples of `auto` and `detail` with the *same* Id (shown in Figure 2b-ii through a solid line between the 2 occurrences of Id).

In the absence of target constraints. Assuming that $\Delta_{\mathcal{G}} = \emptyset$, we illustrate the 4 degrees of contribution by presenting the contribution of the registration R_2 to the answer of 4 queries.

Example 1. *Self Sufficient*
Consider a query Q_1 asking for all car models manufactured in both a 2-door and a 4-door version:

$$Q_1(M) \text{ :- } car(M, C_1, '2', B_1), car(M, C_2, '4', B_2).$$

R_2 is self sufficient w.r.t. Q_1, since, if source 2 is the only registered source, we can find some extent DB_2 of it, for which the set of certain answers to Q_1 is nonempty. Such an instance is the following:

DB_2 : $auto$('1', 'M3 98', 'BMW'), $detail$('1', '2', 'V6'), $detail$('1', '4', 'V6').

In this case all global instances satisfying mapping constraints M_2 contain tuples of the form
car('M3 98', 'BMW', '2', X_1) and car('M3 98', 'BMW', '4', X_2) where X_1, X_2 are constants (potentially different among global instances). Thus the tuple ('M3 98') is an answer to Q_1 against each global instance consistent with M_2 (i.e. it is a certain answer). □

Example 2. *Now Complementary*
Consider now another query Q_2 asking for the make and corresponding ad prices of cars:

$$Q_2(C, P) \text{ :- } car(M, C, D, B), ad(M, S, P).$$

R_2 is now complementary w.r.t. Q_2 and R_1 for the following two reasons: *First*, R_2 is not self sufficient w.r.t. Q_2, since if only source 2 is present in the system, it does not provide any certain answers to Q_2 regardless of its extent. This follows from the fact that always one of the global instances satisfying M_2 will have an empty relation `ad` (since M_2 does not place any restrictions on the contents of `ad`) and thus will give the empty answer to Q_2. *Second*, if both sources 1 and 2 are registered, R_2 contributes certain answers to Q_2 for some extents DB_1, DB_2 of the sources 1 and 2, respectively:

DB_1 : $post$('M3 98', 'A. Brown', '25K', '25K')
DB_2 : $auto$('1', 'M3 98', 'BMW'), $detail$('1', '2', 'V6').

In particular, the set of certain answers to Q_2 in presence of both R_1 and R_2 equals {('BMW', '25K')}, whereas in presence of R_1 alone it is empty (since M_1 does not restrict the values of Carmake). □

Example 3. *Later Complementary*
Assume now that we add an atom to Q_2. The new query Q_3 returns the make and ad prices of models but *only if there exist data about their manufacturer*:

$$Q_3(C, P) \text{ :- } car(M, C, D, B), ad(M, S, P), brand(C, O).$$

Since none of M_1, M_2 restricts the instances of target relation `brand`, the set of certain answers is empty both in the presence of R_1 alone and R_1, R_2 together. Thus R_2 is neither self sufficient, nor now complementary w.r.t. Q_3 and R_1.

However R_2 is later complementary, since there exists a new registration R_3 s.t. R_2 is now complementary w.r.t. Q_3 and $\overline{R'} = R_1, R_3$. R_3 registers a source with schema $\mathcal{S}_3 = \{manufacturer(Carmake, Origin)\}$, which provides the headquarter's location for car manufacturers: $R_3 = (\mathcal{S}_3, M_3)$ with $M_3 = \{(U_3 \subseteq V_3)\}$ and $U_3(C, O)$:- $manufacturer(C, O)$,
$V_3(C, O)$:- $brand(C, O)$. Indeed for some extents DB_1, DB_2, DB_3 of sources 1 through 3, respectively, R_2 contributes to the certain answers of Q:

DB_1 : $post$('M3 98', 'A. Brown', '25K', '25K').
DB_2 : $auto$('1', 'M3 98', 'BMW'), $detail$('1', '2', 'V6')
DB_3 : $manufacturer$('BMW', 'Germany').

In this case the set of certain answers to Q in presence of all sources is {('M3 98', '25K')}, while in the absence of source 2, it is empty (since R_1 and R_3 do not restrict the value of Carmake in the `car` tuples). □

Example 4. *Unusable*
Consider using again Q_2 but projecting also the base price (i.e. return the make, base and ad price of cars):

$$Q_4(C, B, P) \text{ :- } car(M, C, D, B), ad(M, S, P).$$

R_2 is unusable w.r.t. Q_4, because, regardless of any sources registered in the future and of the extents for them and for source 1, R_2 cannot contribute certain answers to Q_4. Indeed, consider arbitrary future registrations $\overline{R'}$ and extents for both the new sources and source 1. By definition of certain answers, for any tuple (C, B, P)

that is a certain answer to Q_4, there will exist a pair of tuples of the form $car(X_1, C, X_2, B), ad(X_1, X_3, P)$ in *all* global instances satisfying R_1 and $\overline{R'}$ (where X_i, $1 \leq i \leq 3$ are constants possibly different among global instances). However the existence of these tuples is not imposed by M_2, since it does not restrict, either the ad tuples, or the value of the Baseprice attribute of the car tuples contained in the global instances. Thus by removing the registration R_2 from the system, we will get the exact same set of certain answers to Q_4. This means that R_2 does not contribute to this set and so it is unusable w.r.t. Q_4. □

In the presence of target constraints. In the presence of integrity constraints on $\mathcal{G}$ (i.e. if $\Delta_\mathcal{G} \neq \emptyset$), the contribution of an individual source is harder to determine, because of the interference via integrity constraints with data provided from other sources. Since each constraint restricts the set of possible global instances, it may change the set of certain answers to a query and consequently the contribution of a source to it. We illustrate this fact by presenting the effect of adding target constraints on some of the previous examples.

By adding constraints on $\mathcal{G}$, a client query Q may become unsatisfiable. In this case, the set of certain answers to Q is always empty and hence every registration becomes unusable w.r.t. it. Such a case is described in Example 5.

Example 5.
Consider again query Q_1 from Example 1, asking for models produced both with 2 and 4 doors. If we add the target constraint:

$$\delta_1 : (U_1' \subseteq V_1') \text{ with}$$
$$U_1'(M, C_1, D_1, B_1, C_2, D_2, B_2) \text{ :-}$$
$$car(M, C_1, D_1, B_1), car(M, C_2, D_2, B_2)$$
$$V_1'(M, C, D, B, C, D, B) \text{ :- } car(M, C, D, B)$$

stating that Model is the primary key (PK) of car, then Q_1 becomes unsatisfiable and thus R_2 (which is self sufficient w.r.t. Q_1 when $\Delta_\mathcal{G} = \emptyset$) becomes unusable. □

However in other cases the addition of a target constraint may instead increase the contribution of a registration. Example 6 illustrates how by adding a primary key on the global schema an otherwise unusable registration becomes now complementary.

Example 6.
Look again at Q_4 asking for the make, base and ad price of models. R_2 (unusable w.r.t. Q_4 when $\Delta_\mathcal{G} = \emptyset$) advances to now complementary w.r.t. Q_4 and R_1 if Model is a PK of car (i.e. if $\Delta_\mathcal{G} = \{\delta_1\}$). Indeed, R_2 contributes certain answers to Q_4 for the source extents shown in Example 2. In particular, all global instances consistent with R_1 and R_2 contain tuples car('M3 98', X_1, X_2, '25K'), ad('M3 98', X_3, '25K') and car('M3 98', 'BMW', '2', X_4) (X_i may be different across instances) and in those satisfying *also* $\Delta_\mathcal{G}$, the two partially specified car tuples are merged into car('M3 98', 'BMW', '2', '25K'). Thus in presence of R_1 and R_2, the set of certain answers to Q_4 contains ('BMW', '25K', '25K'), while if R_1 is registered alone within the system it is empty. Intuitively, a PK on $\mathcal{G}$ allows a source to contribute to a query Q by providing only *part* of a tuple required by Q (the values of the other required attributes of the tuple can be obtained by another source via the PK). □

Other types of target constraints may also have the same effect. Example 7 shows how the addition of referential integrity constraints on $\mathcal{G}$ can lead to the contribution increase of some registration.

Example 7.
Consider again Q_3 asking for the make and ad prices of models for which there exists info about their manufacturer. If we declare the referential integrity constraint:

$$\delta_2 : U_2' \subseteq V_2' \text{ with } U_2'(C) \text{ :- } car(M, C, D, B)$$
$$V_2'(C) \text{ :- } brand(C, O)$$

which states that for each car tuple, a brand tuple with the same make also exists, then each global instance consistent with M_1, M_2 and δ_2 contains for each car tuple provided by M_2 a corresponding brand tuple. Hence R_2 (which is later complementary in the absence of target constraints) becomes now complementary w.r.t. Q_3 and R_1 when $\Delta_\mathcal{G} = \{\delta_2\}$. □

5. MAIN RESULTS

We study only self-sufficiency, now-complementarity and later-complementarity, as unusability is by definition reducible to them.

The decidability of the self-sufficient and now-complementary tests is based on reducing them to reasoning about the *certain rewriting* of the client query Q, in particular to checking satisfiability, respectively containment of certain rewritings.

Certain Rewritings. Given registrations $\{R_i = (\mathcal{S}_i, M_i)\}_{1 \leq i \leq n}$, a certain rewriting of a client query Q against $\mathcal{G}$ is a query against the combined schemas $\mathcal{S}_i$, which returns precisely the certain answers to Q for any source extents $\overline{DB}$. Specifically, denoting a certain rewriting with $rewr_{\overline{R}}(Q)$, we have:

$$rewr_{\overline{R}}(Q)(\overline{DB}) = cert^{\bar{M}}_{\overline{DB}}(Q).$$

Example 8. Recall Q_1 from Example 1, asking for models with both 2 and 4-door versions. Its certain rewriting in R_2's presence is:

$$Q_1'(M) \text{ :- } auto(I_1, M, C_1), detail(I_1, '2', E_1),$$
$$auto(I_2, M, C_2), detail(I_2, '4', E_2).$$

Note that it is not a priori clear that any certain rewriting exists, and even if it does, it is not given that it is expressible in a language in which containment and satisfiability are decidable. Indeed, even for pure LAV data integration scenarios (which are subsumed by the GLAV systems considered here), in the presence of functional dependencies (FDs) on the global schema, the solutions proposed in the literature [10] yield rewritings expressed as recursive Datalog programs, for which containment is undecidable [2].

However, in Theorem 8 in Section 5.4, we show that when there is only at most one FD per relation in $\mathcal{G}$, the certain rewriting can be obtained as a union of conjunctive queries, for which containment is in NP. For the sake of presentation, we present this result last. Notice that the case of allowing only primary key constraints is covered by this result, since PKs are particular cases of FDs, and there can be at most one PK per relation.

When not specified otherwise, in the following results all registrations are given by sets of mapping constraints from IC($CQ^=$), and all queries belong to $UCQ^=$.

5.1 Self-Sufficient Registrations

Our first result concerns the decidability of testing whether a registration R is self-sufficient when the global schema contains only primary keys.

This result is based on the following:

LEMMA 1. *Assume that Q has a certain rewriting $rewr_R(Q)$ expressed in some query language. Then $Suf_Q(R)$ holds if and only if $rewr_R(Q)$ is satisfiable.*

Lemma 1 and Theorem 8 imply:

THEOREM 1. *If $\Delta_\mathcal{G}$ contains only primary keys, then for any client query $Q \in UCQ^=$ it can be decided whether Suf_Q(R) holds in* PTIME *in the size of $rewr_R(Q)$.*

Example 9. The result presented in Example 1 that R_2 is self-sufficient w.r.t. Q_1 can be inferred by checking that $rewr_{R_2}(Q_1)$, shown in Example 8, is satisfiable. □

Theorem 1 does not apply in the presence of RICs because the results in [10] do not provide a certain rewriting in that case. We actually obtain the following result:

THEOREM 2. *It is undecidable to check that, given $\Delta_\mathcal{G}$ containing PKs and RICs, registration R and $Q \in CQ^=$, $Suf_Q(R)$ holds.*

The proof is by a reduction from the Post Correspondence Problem [2].

Since both PKs and RICs are expressible in IC($CQ^=$), we immediately obtain the following corollary:

COROLLARY 1. *It is undecidable to check that, given $\Delta_\mathcal{G} \subseteq IC(CQ^=)$, registration R and $Q \in CQ^=$, $Suf_Q(R)$ holds.*

5.2 Now-Complementary Registrations

The basis for our decidability result for checking now-complementarity consists in a reduction to query containment. We first state the reduction for the most general case it applies in, although for the purpose of decidability we will have to consider only primary keys on the global schema.

The reduction holds when *full dependencies* are allowed on the global schema. Full dependencies are IC($CQ^=$) queries of the form $(U \subseteq V)$, where all of V's variables appear in its head. Full dependencies include all primary keys, functional dependencies, and restricted referential integrity constraints in which V does not project away any attributes.

THEOREM 3. *Given $\Delta_\mathcal{G}$ containing full dependencies and given registrations $\overline{R}, R_{n+1}$, we have that $NComp^Q_{\overline{R}}(R_{n+1})$ holds if and only if*

(i) $rewr_{R_{n+1}}(Q)$ is not satisfiable and
(ii) $rewr_{\overline{R},R_{n+1}}(Q)$ is not contained in $rewr_{\overline{R}}(Q)$.

Theorem 3 and Theorem 8 yield the following:

COROLLARY 2. *If $\Delta_\mathcal{G}$ contains only PKs, then $NComp^Q_{\overline{R}}(R_{n+1})$ can be checked in* NP *in the size of $rewr_{\overline{R}}(Q)$ and in* PTIME *in the size of $rewr_{\overline{R},R_{n+1}}(Q)$.*

We next expose an interesting connection between the problem of deciding self-sufficiency and that of deciding now-complementarity. This will help us transfer undecidability results from the former to the latter.

THEOREM 4. *The problem of deciding self-sufficiency of a registration is reducible to that of deciding now-complementarity.*

Theorem 2 and Theorem 4 yield:

COROLLARY 3. *It is undecidable to check, given $\Delta_\mathcal{G}$ containing PKs and RICs, query $Q \in CQ^=$ and registrations $\overline{R}, R_{n+1}$, whether $NComp^Q_{\overline{R}}(R_{n+1})$ holds.*

Corollary 3 implies undecidability in the presence of IC($CQ^=$) integrity constraints on $\mathcal{G}$.

5.3 Later-Complementary Registrations

The inherent difficulty in deciding later-complementarity lies in having to check now-complementarity with any one of an infinite set of possible registrations. We first show that, in the absence of target constraints ($\Delta_\mathcal{G} = \emptyset$), we can confine the test to a canonically chosen registration, defined below:

Identity Registration. Given $\mathcal{G}$, the registration consisting of schema $\mathcal{G}$ and mappings *copying* its extent to the global database is called the *identity registration* R_{id}:

$R_{id} = (\mathcal{G}, M)$ with $M = \{(U_i \subseteq V_i) : 1 \leq i \leq n\}$ and
$U_i(\bar{x_i})$:- $r_i(\bar{x_i})$, $V_i(\bar{x_i})$:- $r_i(\bar{x_i})$, where
r_i the relations of $\mathcal{G}$ and $\bar{x_i}$ their attributes ($1 \leq i \leq n$).

If there are no constraints on $\mathcal{G}$, later-complementarity reduces to now-complementarity w.r.t. R_{id}:

THEOREM 5. *Assume that $\Delta_\mathcal{G} = \emptyset$ and R_{n+1} is not now complementary w.r.t. Q and $\overline{R}$. Then $LComp^Q_{\overline{R}}(R_{n+1})$ holds iff $NComp^Q_{R_{id}}(R_{n+1})$ holds.*

Corollary 2 and Theorem 5 yield the following:

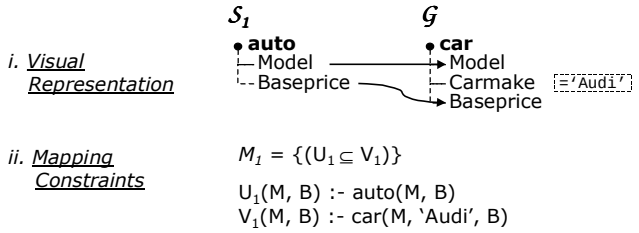
COROLLARY 4. *If $\Delta_\mathcal{G} = \emptyset$, whether $LComp^Q_{\overline{R}}(R_{n+1})$ holds can be decided in* NP *in the size of $rewr_{\overline{R}}(Q)$ and of $rewr_{R_{id}}(Q)$ and in* PTIME *in the size of $rewr_{\overline{R},R_{n+1}}(Q)$ and of $rewr_{R_{id},R_{n+1}}(Q)$.*

Example 10 below shows that the test based on the identity registration fails in the presence of integrity constraints and Theorem 5 cannot be relaxed to allow even PKs in $\Delta_\mathcal{G}$.

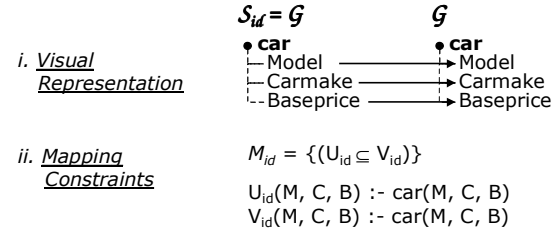
Example 10. Consider a simplified version of our running example. In an initially empty system we are adding source 1 via registration R_1 shown in Figure 3a. The source provides Audi models and their base prices (the equality atom *Carmake = 'Audi'* is depicted as a box with the value 'Audi' next to Carmake). Assuming that Model is the PK of `car`, we want to check the contribution of R_1 to a query asking for the base price of BMW M3 98: $Q(B)$:- $car('M3\ 98', 'BMW', B)$. R_1 is obviously neither self sufficient, nor now complementary w.r.t. Q, since it provides only data about Audis, while the query asks for the price of a BMW.

We check whether R_1 is now complementary w.r.t. Q and R_{id} (shown in Figure 3b). In order to get a certain answer to Q in presence of R_{id} and R_1, a tuple for M3 98 has to be provided by the source with registration R_{id} (it cannot be given by source 1, because it would imply that the make of M3 98 is Audi and thus we would not get any certain answer to Q). But R_{id} by definition, when giving the tuple of M3 98, will also specify all other attributes (i.e. make and price) of the tuple for which Q is asking and thus it will not allow the first source to contribute anything to Q. Hence $NComp^Q_{R_{id}}(R_1)$ does not hold.

However R_1 can contribute to Q if source 2 with registration $R' = (\mathcal{S}', M')$ joins the system, where its schema $\mathcal{S}'$ consists of a single relation `pair`(Model$_1$,Model$_2$) and its set of mappings is $M' = \{(U' \subseteq V')\}$ with
$U'(M_1, M_2)$:- $pair(M_1, M_2)$ and
$V'(M_1, M_2)$:- $car(M_1, 'BMW', B), car(M_2, C, B)$,
which means that source 2 provides pairs of BMW models with other models of the same price. Indeed, assuming the extents: DB_1 : $auto$('S4 97', '25K') and
DB_2 : $pair$('M3 98', 'S4 97') for sources 1 and 2, respectively, R' alone does not provide any certain answers to Q but R_1 and R' together do. Intuitively R_1 contributes the price of BMW M3



a. Registration $R_1 = (S_1, M_1)$ of *new* source 1



b. Identity Registration $R_{id} = (S_{id}, M_{id})$

Figure 3: Registrations for Example 10

98 by providing the price of Audi S4 97 (which according to the data given by source 2 is equal to the base price of BMW M3 98). Hence R_1 is later complementary but not now complementary w.r.t. Q and R_{id}. □

Example 10 also shows that the search for the future registration R' has to consider mappings whose queries perform self-joins. This poses a problem, since even if we picked one of the infinitely many schemas for $\mathcal{S}'$, we'd still be left with an infinite search space obtained by considering all CQ$^=$ queries over $\mathcal{G}$ for the conclusion of the mapping constraints in M' (not to mention the infinite search space for the premise). Our decidability result is based on bounding these search spaces as follows:

Let $R = (\mathcal{S}, M)$ be the registration of our source, Q the client query, and let *size*(Q) be the sum of the arities of all relational atoms in Q.

THEOREM 6. *Assume that $NComp^Q_{\overline{R}}(R)$ does not hold, and $\Delta_\mathcal{G}$ consists only of PKs, with m denoting the maximum size of a key.*

Then $LComp^Q_{\overline{R}}(R)$ holds iff $NComp^Q_{\overline{R'}}(R)$ holds, where $R' = (\mathcal{S}', M')$ and $M' = \{(U' \subseteq V')\}$ such that for some $1 \leq k \leq$ size(Q), we have

- *$\mathcal{S}'$ contains a single relation D of arity $2 \times m \times k$, and*
- *U' is a projection of D, and*
- *V' is a query over $\mathcal{G}$, of arity at most $2 \times m \times k$, with at most $2k$ atoms, and whose constants (if any) appear in M or Q.*

There are therefore exponentially many (in the size of Q) candidate registrations to consider. However, each is of size polynomial in that of Q.

Example 11. For the registration R_1 from Example 10, we have $m = 1$ and *size*$(Q) = 3$. Registration R' is found for $k = 1$: D is the relation `pair` of arity $2 \times 1 \times 1$, V' has arity 2 and has 2 atoms, while U' is a projection of `pair`. □

5.4 Non-recursive Certain Rewritings

Previous work [1, 10] describes only algorithms which yield a certain rewriting as a recursive Datalog program even in local-as-view scenarios, as soon as there are functional dependencies (FDs) in the target schema. This is justified by an example integration scenario in [10] where the FDs force the certain rewriting to be recursive. However, the example uses several FDs per relation. We show here (Theorem 8 below) that, if each relation in the target schema allows at most one FD, then there exists an equivalent, nonrecursive rewriting expressed as a union of conjunctive queries with equalities (UCQ$^=$). Moreover, we show that such a rewriting can be constructed by reusing an algorithm developed in previous work for *exact* rewritings under the *closed world assumption* [8, 9]. These results are of independent interest as they extend previous work on finding certain rewritings, and they shed additional light on the connection between exact rewritings in the closed world assumption and certain rewritings in the open world assumption.

Exact Rewritings. Given registrations $\{R_i = (\mathcal{S}_i, M_i)\}_{1 \leq i \leq n}$, an exact rewriting of a client query Q against $\mathcal{G}$ is a query *xrewr*$_{\overline{R}}(Q)$ against the combined schemas $\mathcal{S}_i$, which returns precisely the answer to Q for any source extents $\overline{DB}$:

$$\forall \overline{DB}\ \forall (G \in targets_{\overline{M}}(\overline{DB}))\ \ xrewr_{\overline{R}}(Q)(\overline{DB}) = Q(G).$$

Notice that any exact rewriting is also a certain rewriting. Clearly, in the classical GLAV scenario in which all mappings are given by source-to-target inclusion constraints [11] (this is the case in all our examples as well), there is no exact rewriting as the inclusions between source data and target data are not violated by adding tuples to the target. This changes $Q(G)$ but does not affect any query over the sources, so its answer cannot be the same with $Q(G)$ for all targets G. Exact rewritings are therefore meaningful only under the Closed World Assumption.

Closed World Assumption. Traditionally, the Open and Closed World Assumptions (OWA, respectively CWA) are defined only when the registration mappings are given by views [1]. We extend the definitions to constraints. Given source extents $\overline{DB}$, registration mappings $\overline{M}$ and target $G \in targets_{\overline{M}}(\overline{DB})$, we say that G is a *minimal target* of $\overline{DB}$ under $\overline{M}$ if no proper sub-instance G' of G is also a target ($G' \notin targets_{\overline{M}}(\overline{DB})$). We say that the registration mappings $\overline{M}$ satisfy the Closed World Assumption (in short, $\overline{M}$ are CWA mappings) if for any source extents, all targets under $\overline{M}$ are minimal. Otherwise, $\overline{M}$ are OWA mappings.

When the registration is given only by source-to-target inclusion constraints, the mappings are OWA.

Previous work [8, 9] has shown that the Chase&Backchase algorithm introduced in [7] is sound and complete for finding all minimized exact rewritings when the query is expressed in UCQ$^=$ and the constraints in IC(UCQ$^=$). Ignoring minimization, the results in [8, 9] imply the existence of the following algorithm, which is guaranteed to construct a canonical exact rewriting whenever it exists (according to Theorem 7 below).

Procedure CANREWR$_{\overline{\mathcal{S}},\mathcal{G},\Delta_\mathcal{G},\overline{M}}(Q)$

1. chase Q with $\overline{M} \cup \Delta_\mathcal{G}$ to obtain a query $U \in$ UCQ$^=$ formulated against the combined schemas $\overline{\mathcal{S}}, \mathcal{G}$. For an extension of the chase to UCQ$^=$ queries and IC(UCQ$^=$) constraints, see [9, 6].
2. construct a query $U|_{\overline{\mathcal{S}}}$ by dropping from all of U's rules all atoms over $\mathcal{G}$. Drop all the rules that have become unsafe (this might result in the empty set of rules, corresponding to the unsatisfiable query).
3. return $U|_{\overline{\mathcal{S}}}$.

A common scenario requiring exact rewriting under the closed world assumption is that of rewriting using materialized views.

Example 12. *Exact rewriting using views.*
Consider the query $Q(y, z)$:- $R(x, y, z)$ and the two materialized views $V_1(x, y)$:- $R(x, y, z)$ and $V_2(x, z)$:- $R(x, y, z)$. Also, let the first component of R be a key. Rewriting Q using only the

views can be seen as finding an exact rewriting of Q using the following CWA mapping. The global schema is $\mathcal{G} = \{R\}$, the source schema is $\mathcal{S} = \{V_1, V_2\}$ and the set of global target constraints $\Delta_\mathcal{G}$ contains only the key on R. The mapping M consists of the constraints

$\{V_1(x,y) \subseteq \exists z\, R(x,y,z), V_2(x,z) \subseteq \exists y\, R(x,y,z),$
$\exists z\, R(x,y,z) \subseteq V_1(x,y), \exists y\, R(x,y,z) \subseteq V_2(x,z)\}$.

Notice that the first and third constraints capture view V_1, stating the inclusions between its materialized extent and the defining query. Similarly, the second and fourth constraint capture view V_2. The first two constraints in M are source-to-target constraints (the queries are expressed as formulae with free variables), whereas the last two state inclusions from the target to the source database. It is easy to see that M satisfies the CWA, and that the query $Q_r(y,z)$:- $V_1(x,y), V_2(x,z)$ is an exact rewriting of Q. Q_r is constructed by algorithm CANREWR as follows. Chasing Q with $M \cup \Delta_\mathcal{G}$ yields

$$U(y,z) \text{ :- } R(x,y,z), V_l(x,y), V_2(x,z)$$

(chase steps apply only with the last two constraints in M) and $U|_\mathcal{S}$ yields Q_r. Q_r turns out to be an exact rewriting, i.e. equivalent to Q, as can be checked by chasing Q_r with the first two constraints in M and with the key constraint, to find a containment mapping from Q into the latter chase result. □

If no exact rewriting exists, then algorithm CANREWR returns a *minimally-containing* UCQ$^=$ rewriting of Q.

Minimally-Containing Rewritings. Given registrations $\{R_i = (\mathcal{S}_i, M_i)\}_{1 \le i \le n}$, a containing rewriting of a client query Q against $\mathcal{G}$ is a query Q_c against the combined schemas $\mathcal{S}_i$, such that:

$$\forall \overline{DB}\, \forall (G \in targets_{\overline{M}}(\overline{DB}))\ \ Q(G) \subseteq Q_c(\overline{DB}).$$

Q_m is a minimally-containing UCQ$^=$ rewriting of Q if it is a containing rewriting and for any other containing UCQ$^=$ rewriting Q_c of Q, we have

$$\forall \overline{DB}\, \forall (G \in targets_{\overline{M}}(\overline{DB}))\ \ Q_m(\overline{DB}) \subseteq Q_c(\overline{DB}).$$

THEOREM 7. *Assume that the chase of Q with $\overline{M} \cup \Delta_\mathcal{G}$ terminates, and that Q has a containing UCQ$^=$ rewriting.*

Then CANREWR$_{\overline{\mathcal{S}},\mathcal{G},\Delta_\mathcal{G},\overline{M}}(Q)$ *is guaranteed to return a minimally containing UCQ$^=$ rewriting Q_m of Q. In particular, if Q admits an exact UCQ$^=$ rewriting, then Q_m is guaranteed to be exact.*

The proof was given in [6]. It follows from the results in [8, 9].

Finding certain rewritings. The algorithm for finding certain rewritings in the OWA is based on modifying the original registration mappings $\overline{M}$ to obtain the CWA mappings $\overline{M_f}$, and then applying algorithm CANREWR on the new mappings. Its result will be the certain rewriting.

To obtain $\overline{M_f}$, we start from the idea introduced in the "inverse-rule algorithm" presented in [10]. This algorithm turns a mapping given by a set $\overline{M}$ of IC(UCQ$^=$) constraints into one given by a *logic program* LP with function symbols, called Skolem functions. LP has two properties: (i) it satisfies the CWA since its output is defined by the minimal model semantics, and (ii) $cert^{\overline{M}}_{\overline{DB}}(Q)$ can be computed by running Q on $LP(\overline{DB})$ and removing from the result all tuples containing Skolem function terms.

Example 13. *Inverse rules (adapted from [10])*
Consider the source schema $\mathcal{S} = \{s\}$, the global schema $\mathcal{G} = \{g\}$, and the mapping given by $M = \{(U \subseteq V)\}$ with $U(P,C)$:- $s(P,C)$ and $V(P,C)$:- $g(A,P,C)$, mapping source relation s into the projection of global relation g. M is turned into the logic program rule

$$g(Sk_F(P,C), P, C) \quad \text{:-} \quad s(P,C) \tag{1}$$

where Sk_F is a fresh Skolem function symbol. By running this program on the extent of s, we get an extent for g whose first column contains fresh invented values computed using Sk_F. If we fix Sk_F, the extent of g is uniquely determined by that of s. An important requirement of Sk_F (coming from [10]) is that it never assign the same value to distinct arguments, unless forced to do so by an FD in $\Delta_\mathcal{G}$. For instance, if we assume the FD $fd_1 = C \to A$ on g, then $Sk_F(P,C) = Sk_F(P',C)$ for any values of P, P', in order not to violate fd_1. However, $Sk_F(P,C) \neq Sk_F(P,C')$ for all $C \neq C'$. In the absence of any FDs, Sk_F must be injective. We refer to this property of Sk_F as *injectivity modulo FDs*. □

[10] shows how LP can be turned into a Datalog program DP without function symbols, which is a certain rewriting. However, in the presence of FDs in $\Delta_\mathcal{G}$, DP is recursive and thus doesn't serve our purpose. We take an alternate approach, turning LP into a new set of mapping constraints $\overline{M_f}$, together with a set of new target constraints $\Delta^f_\mathcal{G}$. For simplicity of presentation, we only illustrate the construction of $\overline{M_f}$ and $\Delta^f_\mathcal{G}$ on the above example.

Example 14. *Capturing inverse rules with constraints*
We first obtain the new global schema $\mathcal{G}_f$ by extending schema $\mathcal{G}$ with a relation F modeling the Skolem function (its intended meaning is that $F(A,P,C)$ iff $Sk_F(P,C) = A$). Then we eliminate the Skolem terms from the logic program rules, substituting F for Sk_F. (1) thus becomes $(U_0 \subseteq V_0) \in M_f$ where

$U_0(P,C)$:- $s(P,C)$
$V_0(P,C)$:- $g(A,P,C), F(A,P,C)$.

To ensure that F models a function, we add to $\Delta^f_\mathcal{G}$ the corresponding FD $PC \to A$ on F, expressible as $U_1 \subseteq V_1$:

$U_1(A_1,A_2)$:- $F(A_1,P,C), F(A_2,P,C)$,
$V_1(A,A)$:- $F(A,P,C)$.

We specify F's injectivity modulo FDs as follows. Recall from Example 13 that in the presence of fd_1, Sk_F will return the same A value on arguments agreeing on the C value. To express that this is the only case when the results of Sk_F coincide, we add to $\Delta^f_\mathcal{G}$ the constraint $(U_2 \subseteq V_2)$:

$$\begin{aligned} U_2(A,C,C') \quad &\text{:-} \quad g(A,P,C), F(A,P,C), \\ &\qquad\ g(A,P',C'), F(A,P',C') \\ V_2(A,C,C) \quad &\text{:-} \quad g(A,P,C), F(A,P,C) \end{aligned}$$

We next turn the mapping into one satisfying the CWA. Consider the decomposition $\rho = \{AC, PC\}$ of g. This is a lossless join decomposition, which cannot be further decomposed without compromising the lossless join property [2]. For each $X \in \rho$, we introduce a corresponding constraint into M_f, stating that all tuples in $\pi_X(g)$ must stem from the source via the evaluation of the logic program rule on s (this is the minimality requirement). Specifically, we add $\{U_3 \subseteq V_3, U_4 \subseteq V_4\}$ to M_f, where

$U_3(A,C)$:- $g(A,P,C)$,
$V_3(A,C)$:- $s(P',C), F(A,P',C)$, and
$U_4(P,C)$:- $g(A,P,C)$,
$V_4(P,C)$:- $s(P,C)$.

Notice that these two constraints state inclusions from the target to the source.

Now consider the query $Q(P_1,P_2)$:- $g(A,P_1,C_1), g(A,P_2,C_2)$. CANREWR$_{\mathcal{S},\mathcal{G}_f,\Delta^f_\mathcal{G},M_f}(Q)$ proceeds as follows. It first chases Q with $(U_4 \subseteq V_4)$ (two steps), then with $(U_0 \subseteq V_0)$ (two steps) and

$(U_3 \subseteq V_3)$ (another two), followed by two steps with the PK on g, and finally with $(U_2 \subseteq V_2)$ (one step) to obtain

$$\begin{array}{lll} U(P_1, P_2) & \text{:-} & g(A, P_1, C_1), g(A, P_2, C_2), \\ & & s(P_1, C_1), s(P_2, C_2), \\ & & g(A', P_1, C_1), F(A', P_1, C_1) \\ & & g(A'', P_2, C_2), F(A'', P_2, C_2) \\ & & s(P_1', C_1), F(A, P_1', C_1), \\ & & s(P_2', C_2), F(A, P_2', C_2), \\ & & A = A', A = A'', C_1 = C_2 \end{array}$$

$U|_{\mathcal{S}}$ yields

$$Q_m(P_1, P_2) \text{ :- } s(P_1, C_1), s(P_2, C_1), s(P_1', C_1), s(P_2', C_1),$$

which minimizes trivially to $Q_m(P_1, P_2)$:- $s(P_1, C_1), s(P_2, C_1)$. This is the certain rewriting of Q. Intuitively, this is because Q requires P_1, P_2 to be associated with the same A. Since the value of A is not provided by the source, we use the next best condition, namely we require P_1, P_2 to coincide on C. Since the latter determines A, all tuples returned by Q_m are valid. □

It turns out that, for several FDs per relation, the constraints for injectivity modulo FDs become more complex, leading to guaranteed non-termination of the chase. Interestingly, the successive queries obtained during the infinite chase sequence enumerate the unfoldings of the recursive Datalog program constructed in [10] from LP.

Algorithm CREWR$(\overline{\mathcal{S}}, \mathcal{G}, \Delta_{\mathcal{G}}, \overline{R}, Q)$

1. Construct the "inverse-rules" logic program LP corresponding to mapping $\overline{M}$.
2. Construct a new global schema $\mathcal{G}_f$, a new set of target constraints $\Delta_{\mathcal{G}}^f$, and new mapping constraints $\overline{M_f}$ which induce the same mapping as LP.
3. Return CANREWR$_{\overline{\mathcal{S}}, \mathcal{G}_f, \Delta_{\mathcal{G}}^f, \overline{M_f}}(Q)$.

THEOREM 8. *If $\Delta_{\mathcal{G}}$ contains at most one functional dependency per relation, then for any list of registrations $\overline{R}$ and any client query $Q \in$ UCQ$^=$, Q has a nonrecursive certain UCQ$^=$ rewriting and algorithm* CREWR *finds such a rewriting.*

The crux of the proof is that (i) $\overline{M}_f$ induces the same mapping as the inverse-rule logic program LP, and (ii) the chase with the constraints in $\overline{M}_f \cup \Delta_{\mathcal{G}}^f$ is guaranteed to terminate for any client query Q, so that Theorem 7 applies.

6. CONCLUSIONS AND RELATED WORK

We believe that a first step towards making data integration accessible to non-expert users consists in providing visual interfaces which guide users through the process. We are currently developing a tool whose front-end uses the Clio system [18] to help the user specify registrations. Determining whether the registration of a new source is at all relevant to an existing application is non-trivial and requires automated assistance. We have formulated a novel classification of registrations and studied the problem of deciding on registration categories.

Our classification is independent of the underlying source instances. We are contemplating extensions of our notions of contribution degrees to take into account source extents. Technically, we can incorporate knowledge about the data values as constraints added to the mapping constraints. This leads us to explore the integration setting with constraints on the source schemas.

To extend applicability of our tools, we will consider extending the class of queries and mapping constraints supported, both in the presence of data instances and in their absence. We expect an interesting trade-off between feasibility and expressivity.

Related Work. [10] provides the "inverse-rule" algorithm for finding maximally contained rewritings for LAV integration, and [1] shows that maximally contained rewritings coincide with certain rewritings. The result was extended in [13] to the GLAV case, but in the absence of integrity constraints on the global schema. A further extension to the nested relational and XML data models is provided in [21]. [15] shows how to find the certain rewritings in a GLAV-like setting used in peer mediation. [16] settles the complexity and decidability of checking the existence of this rewriting (the GLAV mappings are called "symmetric constraints" there), considering both cyclic and acyclic sets of mapping constraints. In the presence of functional dependencies on the global schema, the inverse-rule algorithm returns a recursive Datalog program, defeating our goal of reasoning about containment of certain rewritings. [19] decides containment of certain rewritings in LAV scenarios without integrity constraints. The authors consider limited access capabilities, which also result in certain rewritings expressed as recursive Datalog programs. [3] reduces a GLAV mapping into a pure GAV mapping, exploiting integrity constraints on the global schema. [4] extends the ideas in [3] to allow for inclusion dependencies on the global schema. This work is extended in [5], which performs a comprehensive study of the complexity of answering client queries for certain answers, under the Open World Assumption, but also for various relaxations of the query semantics which are appropriate for inconsistent and incomplete information. [5] shows that query answering is undecidable in the presence of arbitrary PKs and inclusion dependencies and that, for restrictions on the inclusion dependencies, the problem becomes decidable. Various complexities are obtained for various classes of constraints and queries. Despite the fact that the work in [5] assumes the source instance given while our definitions are instance-independent, we plan to investigate whether the restrictions in [5] lead to improved complexities in our setting.

7. REFERENCES

[1] Serge Abiteboul and Oliver Duschka. Complexity of answering queries using materialized views. In *PODS*, 1998.

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] Andrea Cali, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. In *CAiSE*, 2002.

[4] Andrea Cali, Giuseppe De Giacomo, and Maurizio Lenzerini. Models of information integration: Turning local-as-view into global-as-view. In *FMII*, 2001.

[5] Andrea Cali, Domenico Lembo and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *PODS*, 2003.

[6] Alin Deutsch, Bertram Ludäscher and Alan Nash. Rewriting Queries Using Views with Access Patterns under Integrity Constraints. In *ICDT*, 2005.

[7] Alin Deutsch and Lucian Popa and Val Tannen. Physical Data Independence, Constraints and Optimization using Universal Plans. In *VLDB*, 1999.

[8] Alin Deutsch and Val Tannen. Reformulation of xml queries and constraints. In *ICDT*, 2003.

[9] Alin Deutsch and Val Tannen. XML Queries and Constraints, Containment and Reformulation. *JTCS (invited paper)*, to appear in May 2005.

[10] Oliver Duschka, Michael Genesereth, and Alon Levy. Recursive query plans for data integration. *Journal of Logic Programming*, 43(1):49–73, 2000.

[11] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, 2003.

[12] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. In *PODS*, 2003.

[13] Marc Friedman, Alon Levy, and Todd Millstein. Navigational plans for data integration. In *AAAI*, 1999.
[14] Alon Halevy. Logic-based techniques in data integration. In *Logic Based Artificial Intelligence*, 2000.
[15] Alon Halevy, Zachary Ives, Jayant Madhavan, Peter Mork, Dan Suciu, and Igor Tatarinov. The Piazza peer data management system. *IEEE Transactions on Data Engineering (to appear)*, 2004.
[16] Christoph Koch. Query rewriting with symmetric constraints. In *FoIKS*, 2002.
[17] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, 2002.
[18] Renee Miller, Mauricio Hernandez, Laura Haas, Ling-Ling Yan, C. T. Howard Ho, Ronald Fagin, and Lucian Popa. The Clio project: Managing heterogeneity. *SIGMOD Record*, 1(30), 2001.
[19] Todd Millstein, Alon Y. Halevy, and Marc Friedman. Query containment for data integration systems. *JCSS*, 2002.
[20] Jeffrey D. Ullman. Information integration using logical views. In *ICDT*, 1997.
[21] Cong Yu and Lucian Popa. Constraint-based XML query rewriting for data integration. In *SIGMOD*, 2004.

APPENDIX

A. SOME PROOFS

PROOF. (**of Lemma 1**) By definition, $Suf_Q(R)$ holds if and only if the certain answer to Q is non-empty on some database DB, if and only if $rewr_R(Q)$ returns a non-empty answer on DB, if and only if $rewr_R(Q)$ is satisfiable. □

PROOF. (**of Theorem 1**) By Theorem 1, it suffices to check satisfiability of $rewr_R(Q)$. Since by Theorem 8, we have that $rewr_R(Q) \in \text{UCQ}^{=}$, this amounts to finding at least one satisfiable individual conjunct C in the union. C is satisfiable if and only if there is no equality between distinct constants which follows by transitivity, symmetry and reflexivity from the equality atoms of C. □

PROOF. (**of Theorem 2**) The proof is a reduction from the Post Correspondence Problem (PCP). Let $L_1 = \{u_i\}_{1\leq i\leq n}$, $L_2 = \{v_i\}_{1\leq i\leq n}$ be lists of words over an alphabet Σ (i.e. $u_i \in \Sigma^*$, $v_i \in \Sigma^*$, $1 \leq i \leq n$). A *solution* to PCP is a sequence of indexes $i_1, ..., i_m$ s.t. $u_{i_1}u_{i_2}...u_{i_m} = v_{i_1}v_{i_2}...v_{i_m}$. The string $u_{i_1}u_{i_2}...u_{i_m}$ is called the *expansion of this solution*. In order to prove the theorem, for any PCP instance we create an instance of the Self Sufficiency Problem (SSP) (i.e. a global schema $\mathcal{G}$, a set of target constraints $\Delta_\mathcal{G}$, a source registration $R = (\mathcal{S}, M)$ and a query Q) s.t. PCP has a solution iff $Suf_Q(R)$ holds.

For ease of exposition we first present a reduction using one target constraint that is neither a PK nor a RIC and then change the SSP instance to include only PKs and RICs as target constraints.

First Cut. The constructed SSP instance that contains a non-PK and non-RIC target constraint δ_3 is shown in Figure 4. Source relation $E_\mathcal{S}$ is the edge relation of a labeled directed graph with $E_\mathcal{S}(s, l, t)$ describing the edge from node s to t with label l. The intention is to represent a word $w = a_1...a_p$ (where a_i are letters) by a chain of the form $E_\mathcal{S}(x_1, a_1, x_2), ..., E_\mathcal{S}(x_p, a_p, x_{p+1})$. On the global schema, target relation C is intended to contain tuples $C(s_u, s_v, i, t_u, t_v)$ if from pair of nodes s_u, s_v we can reach nodes t_u, t_v following paths representing u_i, v_i, respectively. Additionally, target relation R should contain a tuple $R(t_u, t_v)$ if nodes t_u, t_v are reachable from the *same* node s by paths representing $u_{i_1}...u_{i_k}$ and $v_{i_1}...v_{i_k}$, respectively for some indexes $i_1, ..., i_k$. Finally, query Q asks for the existence of a tuple $R(x, x)$, i.e. of a node x reachable by some node s both by a path representing $u_{i_1}...u_{i_k}$ and one representing $v_{i_1}...v_{i_k}$. Since however the graph contains only chains, these paths will coincide and represent the expansion of a solution to the PCP. Thus Q has nonempty certain answers for some source instance (i.e. $Suf_Q(R)$ holds) iff PCP has a solution.

- **Local schema** $\mathcal{S}$ = $\{E_\mathcal{S}$: 3-ary $\}$
- **Global schema** $\mathcal{G}$ = $\{E^1_\mathcal{G}$: 3-ary, $E^2_\mathcal{G}$: 3-ary, C: 5-ary, R: 2-ary $\}$
- **Set of target constraints** $\Delta_\mathcal{G}$ = $\{(U^1_\delta \subseteq V^1_\delta), (U^2_\delta \subseteq V^2_\delta), \delta_3\}$
  $U^1_\delta(s, l_1, t_1, l_2, t_2)$:- $E^1_\mathcal{G}(s, l_1, t_1), E^1_\mathcal{G}(s, l_2, t_2)$
  $V^1_\delta(s, l, t, l, t)$:- $E^1_\mathcal{G}(s, l, t)$

  $U^2_\delta(t, s_1, l_1, s_2, l_2)$:- $E^2_\mathcal{G}(s_1, l_1, t), E^2_\mathcal{G}(s_2, l_2, t)$
  $V^2_\delta(t, s, l, s, l)$:- $E^2_\mathcal{G}(s, l, t)$

  $\delta_3 = (\forall x, y, i[R(x, y) \wedge C(x, y, i, x', y') \rightarrow R(x', y')])$
- **Set of mappings** M = $\{(U^i_\mathcal{G} \subseteq V^i_\mathcal{G}) | 1 \leq i \leq 2n + 2\}$
  $U^1_\mathcal{G}(s, l, t)$:- $E_\mathcal{S}(s, l, t)$ and $V^1_\mathcal{G}(s, l, t)$:- $E^1_\mathcal{G}(s, l, t)$
  $U^2_\mathcal{G}(s, l, t)$:- $E_\mathcal{S}(s, l, t)$ and $V^2_\mathcal{G}(s, l, t)$:- $E^2_\mathcal{G}(s, l, t)$
  ---
  foreach $1 \leq i \leq n$, let $u_i = a_1...a_k$ and $v_i = b_1...b_l$
  $U^{i+2}_\mathcal{G}(x_1, y_1, i, x_{k+1}, y_{l+1})$:-
  $E_\mathcal{S}(x_1, a_1, x_2), E_\mathcal{S}(x_2, a_2, x_3), ..., E_\mathcal{S}(x_k, a_k, x_{k+1}),$
  $E_\mathcal{S}(y_1, b_1, y_2), E_\mathcal{S}(y_2, b_2, y_3), ..., E_\mathcal{S}(y_l, b_l, y_{l+1})$
  $V^{i+2}_\mathcal{G}(x_1, y_1, i, x_2, y_2)$:- $C(x_1, y_1, i, x_2, y_2)$
  ---
  let l_0 be a letter not in Σ
  foreach $1 \leq i \leq n$, let $u_i = a_1...a_k$ and $v_i = b_1...b_l$
  if one of u_i, v_i is a prefix of the other, then
  $U^{i+n+2}_\mathcal{G}(x_k, y_l)$:-
  $E_\mathcal{S}(s_0, l_0, s),$
  $E_\mathcal{S}(s, a_1, x_1), E_\mathcal{S}(x_1, a_2, x_2), ..., E_\mathcal{S}(x_{k-1}, a_k, x_k),$
  $E_\mathcal{S}(s, b_1, y_1), E_\mathcal{S}(y_1, b_2, y_2), ..., E_\mathcal{S}(y_{l-1}, b_l, y_l)$
  $V^{i+n+2}_\mathcal{G}(x_1, y_1)$:- $R(x_1, y_1)$
  else remove the constraint $(U^{i+n+2}_\mathcal{G} \subseteq V^{i+n+2}_\mathcal{G})$
- **Client Query** $Q()$:- $R(x, x)$

Figure 4: Self Sufficiency Problem Instance for First Cut

The above semantics are specified as follows: The mapping constraints $(U^i_\mathcal{G} \subseteq V^i_\mathcal{G}), 1 \leq i \leq 2$ and target constraints $(U^i_\delta \subseteq V^i_\delta), 1 \leq i \leq 2$ restrict the source instances to graphs consisting of a set of disjoint chains and cycles. Note that the source instances cannot be restricted to graphs containing only chains, since chains and cycles are indistinguishable by first order formulas. Furthermore, the mapping constraints $(U^i_\mathcal{G} \subseteq V^i_\mathcal{G}), 3 \leq i \leq n+2$ are used to capture the intended meaning for relation C.

Finally, target constraint δ_3 and mapping constraints $(U^i_\mathcal{G} \subseteq V^i_\mathcal{G}), n + 3 \leq i \leq 2n + 2$ implement the semantics of relation R, which, according to its definition, should contain the transitive closure of C. The recursive step to obtain the transitive closure is described by δ_3 and the base case of the recursion is captured by constraints $(U^i_\mathcal{G} \subseteq V^i_\mathcal{G}), n + 3 \leq i \leq 2n + 2$. The base case consists of pairs of nodes t_u, t_v s.t. they are reachable from a node s (with an incoming edge label $l_0 \notin \Sigma$) both through a path representing u_i and through one representing v_i, where u_i is a prefix of v_i or vice versa. The prefix requirement is due to the fact that if $i_1, ..., i_m$ is a solution to the PCP, then one of u_{i_1}, v_{i_1} will be a prefix of the other. Moreover the requirement that the start node s has an incoming edge labeled $l_0 \notin \Sigma$ avoids considering a path from s to t as the expansion of a solution to the PCP when there is a path from s to t through $u_{i_1}...u_{i_m}$ and one through $v_{i_1}...v_{i_m}$ but one of these paths goes around a cycle on which s, t are located more times than the other.

Let us now modify the SSP instance, so that $\Delta_\mathcal{G}$ consists of PKs

- **Local schema** $\mathcal{S}$ = $\{E_{\mathcal{S}}$: 3-ary $\}$
- **Global schema** $\mathcal{G}$ = $\{E^1_{\mathcal{G}}$: 3-ary, $E^2_{\mathcal{G}}$: 3-ary, C': 7-ary, C'': 3-ary, C''': 1-ary $\}$
- **Set of target constraints** $\Delta_{\mathcal{G}} = \{(U^i_\delta \subseteq V^i_\delta) | 1 \leq i \leq 6\}$

  $U^1_\delta(s, l_1, t_1, l_2, t_2)$:- $E^1_{\mathcal{G}}(s, l_1, t_1), E^1_{\mathcal{G}}(s, l_2, t_2)$
  $V^1_\delta(s, l, t, l, t)$:- $E^1_{\mathcal{G}}(s, l, t)$

  $U^2_\delta(t, s_1, l_1, s_2, l_2)$:- $E^2_{\mathcal{G}}(s_1, l_1, t), E^2_{\mathcal{G}}(s_2, l_2, t)$
  $V^2_\delta(t, s, l, s, l)$:- $E^2_{\mathcal{G}}(s, l, t)$

  $U^3_\delta(s_0, t_u, t_v)$:- $C'(s_0, s_u, s_v, i, t_0, t_u, t_v)$
  $V^3_\delta(t_0, t_u, t_v)$:- $C'(s_0, s_u, s_v, i, t_0, t_u, t_v)$

  $U^4_\delta(s_0, s_u, s_v)$:- $C'(s_0, s_u, s_v, i, t_0, t_u, t_v)$
  $V^4_\delta(o, t_1, t_2)$:- $C''(o, t_1, t_2)$

  $U^5_\delta(t_0, t_u, t_v)$:- $C'(s_0, s_u, s_v, i, t_0, t_u, t_v)$
  $V^5_\delta(o, t_1, t_2)$:- $C''(o, t_1, t_2)$

  $U^6_\delta(t_1, t_2, o_1, o_2)$:- $C''(o_1, t_1, t_2), C''(o_2, t_1, t_2)$
  $V^6_\delta(t_1, t_2, o, o)$:- $C''(o, t_1, t_2)$
- **Set of mappings** $M = \{(U^i_{\mathcal{G}} \subseteq V^i_{\mathcal{G}}) | 1 \leq i \leq 2n+3\}$

  $U^1_{\mathcal{G}}(s, l, t)$:- $E_{\mathcal{S}}(s, l, t)$ and $V^1_{\mathcal{G}}(s, l, t)$:- $E^1_{\mathcal{G}}(s, l, t)$
  $U^2_{\mathcal{G}}(s, l, t)$:- $E_{\mathcal{S}}(s, l, t)$ and $V^2_{\mathcal{G}}(s, l, t)$:- $E^2_{\mathcal{G}}(s, l, t)$

  foreach $1 \leq i \leq n$, let $u_i = a_1 ... a_k$ and $v_i = b_1 ... b_l$
  $U^{i+2}_{\mathcal{G}}(x_1, y_1, i, x_{k+1}, y_{l+1})$:-
  $E_{\mathcal{S}}(x_1, a_1, x_2), E_{\mathcal{S}}(x_2, a_2, x_3), ..., E_{\mathcal{S}}(x_k, a_k, x_{k+1}),$
  $E_{\mathcal{S}}(y_1, b_1, y_2), E_{\mathcal{S}}(y_2, b_2, y_3), ..., E_{\mathcal{S}}(y_l, b_l, y_{l+1})$
  $V^{i+2}_{\mathcal{G}}(x_1, y_1, i, x_2, y_2)$:- $C'(z_1, x_1, y_1, i, z_2, x_2, y_2)$

  let l_0 be a letter not in Σ
  foreach $1 \leq i \leq n$, let $u_i = a_1 ... a_k$ and $v_i = b_1 ... b_l$
  if one of u_i, v_i is a prefix of the other, then
  $U^{i+n+2}_{\mathcal{G}}(s, x_k, y_l)$:-
  $E_{\mathcal{S}}(s_0, l_0, s),$
  $E_{\mathcal{S}}(s, a_1, x_1), E_{\mathcal{S}}(x_1, a_2, x_2), ..., E_{\mathcal{S}}(x_{k-1}, a_k, x_k),$
  $E_{\mathcal{S}}(s, b_1, y_1), E_{\mathcal{S}}(y_1, b_2, y_2), ..., E_{\mathcal{S}}(y_{l-1}, b_l, y_l)$
  $V^{i+n+2}_{\mathcal{G}}(s, x_1, y_1)$:- $C'(s, x_1, y_1, z_1, z_2, z_3, z_4)$
  else remove the constraint $(U^{i+n+2}_{\mathcal{G}} \subseteq V^{i+n+2}_{\mathcal{G}})$

  $U^{2n+3}_{\mathcal{G}}(x)$:- $E_{\mathcal{S}}(s_0, l_0, x)$
  $V^{2n+3}_{\mathcal{G}}(x)$:- $C'''(x)$
  where l_0 the same letter used above
- **Client Query** $Q()$:- $C'(s_0, s_u, s_v, i, t_0, t, t), C'''(t_0)$

Figure 5: Self Sufficiency Problem Instance for Second Cut

and RICs only.

Second Cut. The new instance is shown in Figure 5. While using the same source schema, we are now capturing the semantics of both the relations C and R used in the first cut by a *single* target relation C'. The intention is that if from a node s_0 (which has an incoming l_0 edge), we can reach nodes s_u, s_v through chains representing $u_{i_1} ... u_{i_k}$ and $v_{i_1} ... v_{i_k}$, respectively for some indexes $i_1, ..., i_k$ and additionally from nodes s_u, s_v we can reach t_u, t_v through chains describing u_i, v_i, respectively for some index i, then there should exist a tuple $C'(s_0, s_u, s_v, i, s_0, t_u, t_v)$. Relation C''' is a unary relation containing the nodes that can serve as starts of a chain (i.e. nodes with an incoming edge labeled with l_0).

Thus the query Q is now asking for the existence of a node t, which is reachable from some node t_0 that can serve as the start of a chain both via a path labeled $u_{i_1} ... u_{i_k}$ and one labeled $v_{i_1} ... v_{i_k}$. For the reasons outlined in the first cut, $Suf_Q(R)$ holds iff PCP has a solution.

The intended semantics for the relations are again implemented through constraints. As in the first cut, mapping constraints $(U^i_{\mathcal{G}} \subseteq V^i_{\mathcal{G}})$, $1 \leq i \leq 2$ and target constraints $(U^i_\delta \subseteq V^i_\delta)$, $1 \leq i \leq 2$ disallow the existence of anything different from disjoint chains and cycles in the graph described by $E_{\mathcal{S}}$.

Additionally, mapping constraints $(U^i_{\mathcal{G}} \subseteq V^i_{\mathcal{G}})$, $3 \leq i \leq n+2$ map into C' what was mapped into C in the first cut. Similarly constraints $(U^i_{\mathcal{G}} \subseteq V^i_{\mathcal{G}})$, $n+3 \leq i \leq 2n+2$ bring into C' what was mapped into R in the first cut to form the base case for the recursion. The semantics of the recursive step are the following: If there exists a tuple $C'(s_0, s_u, s_v, i, t_0, t_u, t_v)$ then t_0 should be equal to s_0 and moreover, whenever an additional tuple $C'(s'_0, t_u, t_v, i', t'_u, t'_v)$ appears in C', s'_0 should be equal to t_0. These semantics are ensured through constraints $(U^i_\delta \subseteq V^i_\delta)$, $3 \leq i \leq 6$. Finally the intended semantics of C''' are captured by mapping constraint $(U^{2n+3}_{\mathcal{G}} \subseteq V^{2n+3}_{\mathcal{G}})$. □

PROOF. **(of Theorem 3)** By definition, registration R_{n+1} is now complementary iff it is not self-sufficient and there are source extents $\overline{DB}, DB_{n+1}$ s.t. $cert^{\overline{M}, M_{n+1}}_{\overline{DB}, DB_{n+1}}(Q) \setminus cert^{\overline{M}}_{\overline{DB}}(Q) \neq \emptyset$.

(i) The proof of Theorem 1 shows that R_{n+1} is self-sufficient iff $rewr_{R_{n+1}}(Q)$ is satisfiable.

(ii) There are source extents $\overline{DB}, DB_{n+1}$ s.t. $cert^{\overline{M}, M_{n+1}}_{\overline{DB}, DB_{n+1}}(Q) \setminus cert^{\overline{M}}_{\overline{DB}}(Q) \neq \emptyset$
iff
there are $\overline{DB}, DB_{n+1}$ with
$rewr_{\overline{R}, R_{n+1}}(Q)(\overline{DB}, DB_{n+1}) \setminus rewr_{\overline{R}}(Q)(\overline{DB}) \neq \emptyset$
iff
there are $\overline{DB}, DB_{n+1}$ with

$rewr_{\overline{R}, R_{n+1}}(Q)(\overline{DB}, DB_{n+1}) \setminus rewr_{\overline{R}}(Q)(\overline{DB}, DB_{n+1}) \neq \emptyset$
(since $rewr_{\overline{R}}(Q)$ does not mention the schema of DB_{n+1})
iff
$rewr_{\overline{R}, R_{n+1}}(Q)$ is not contained in $rewr_{\overline{R}}(Q)$. □

PROOF. **(of Theorem 4)** Given global schema $\mathcal{G}$, local schema $\mathcal{S}$, registration $R = (\mathcal{S}, M)$ and query Q formulated against $\mathcal{G}$, we construct the new local schema $\mathcal{G}'$, two local schemas $\mathcal{S}_1, \mathcal{S}_2$ with registrations R_1, R_2, and query Q' against $\mathcal{G}'$ such that R is self-sufficient w.r.t. Q if and only if R_2 is now-complementary w.r.t. R_1 and Q': $Suf_R(Q)$ iff $NComp^Q_{R_1}(R_2)$.

We obtain $\mathcal{G}'$ by adding to $\mathcal{G}$ a fresh, zero-ary relation N. Q' is obtained from Q by adding the atom $N()$ to the body. $\mathcal{S}_1$ consists of a single, zero-ary relation O, and the registration mapping M_1 is given by $M_1 = \{(U \subseteq V)\}$ with $U()$:- $O()$ and $V()$:- $N()$. $\mathcal{S}_2$ and R_2 coincide with $\mathcal{S}$ and R, respectively.

Notice that given the same extent to the sources of schema $\mathcal{S}$, query Q' returns the same answer as Q if the extent of O is not empty, and returns the empty answer otherwise.

We observe that $Suf_{Q'}(R_2)$ does not hold, as M_2 does not map into N and therefore regardless of the extent of source 2, there is always a target database in which N is empty, and one in which it is not (contains the empty tuple). The certain answers to Q' are therefore always empty if source 2 is alone in the system. That is, R_2 is not self-sufficient.

Similar reasoning shows that source 1 alone contributes no certain answers to Q' regardless of its extent, as it does not map into the other relations appearing in Q'.

Therefore, in order for source 2 to contribute additional certain answers to Q' in the presence of source 1, it suffices to contribute any one tuple. Any such answer would be an answer to query Q as well. We conclude that $Suf_R(Q)$ iff $NComp^Q_{R_1}(R_2)$. □

Proof of Theorem 5 We use the concept of *canonical target instances* found in the literature [11, 12]. We give its definition for $\Delta_{\mathcal{G}} = \emptyset$.

Canonical Target Instances. Given source registrations $\overline{R}$ and corresponding extents $\overline{DB}$, a canonical target instance of $\overline{R}$ and $\overline{DB}$ is a *minimal* global instance that satisfies the source mappings and contains specially constructed Skolems for all attributes whose values are not restricted by $\overline{R}$.

Note that due to different naming of the Skolems, many canonical target instances may exist. We denote by $can(\overline{R}; \overline{DB})$ any one of them.

Such an instance can be easily created by treating the mapping constraints as rules that, starting from tuples of the sources, generate tuples in G. For a detailed explanation of the construction of a canonical target instance, see [11, 12, 21].

This work also shows that by running Q on any canonical target instance and removing from the result all tuples containing Skolems (denoted by $Q(can(\overline{M}; \overline{DB}))\downarrow$) we can compute the set of certain answers to Q:

LEMMA 2. *For $\Delta_{\mathcal{G}} = \emptyset$ and $Q \in$ UCQ$^{=}$ the following holds:* $cert^{\overline{M}}_{\overline{DB}}(Q) = Q(can(\overline{R}; \overline{DB}))\downarrow$

PROOF. (**of Theorem 5**) *(if)* This direction immediately follows by the definition of Later Complementary.
(only if) Assume that $LComp^{Q}_{\bar{R}}(R_{n+1})$ holds. Then by definition there exist registrations $\overline{R'}$ and source extents $\overline{DB'}$ and DB_{n+1} s.t. $cert^{\overline{M'}, M_{n+1}}_{\overline{DB'}, DB_{n+1}}(Q) \setminus cert^{\overline{M'}}_{\overline{DB'}}(Q) \neq \emptyset$. We will, based on the existence of $\overline{R'}$, $\overline{DB'}$, show that there also exists an extent DB_{id} of a source with the identity registration such that
$cert^{M_{id}, M_{n+1}}_{DB_{id}, DB_{n+1}}(Q) \setminus cert^{M_{id}}_{DB_{id}}(Q) \neq \emptyset$.

We obtain DB_{id} by building $can(\overline{R'}; \overline{DB'})$ and replacing any distinct Skolem value in it by a distinct fresh constant (i.e. a constant that appears neither in $\overline{DB}$, DB_{n+1}, nor in Q). Since R_{id} copies the source extent to the target without leaving any value unspecified, it is easy to see that $can(R_{id}; DB_{id}) = DB_{id}$. Thus $cert^{M_{id}}_{DB_{id}}(Q) = Q(can(R_{id}; DB_{id}))\downarrow = Q(DB_{id})$. However by construction, DB_{id} contains all information provided by $can(\overline{R'}, \overline{DB'})$ plus specific new values instead of Skolems. Thus $Q(DB_{id})$ (which has been shown equal to $cert^{M_{id}}_{DB_{id}}(Q)$) contains all tuples of $Q(can(\overline{R'}; \overline{DB'}))\downarrow$ plus some new ones, each of which contains at least one of the fresh constants introduced in DB_{id}. Thus if we denote by A this set of new tuples, the following holds:

$$cert^{M_{id}}_{DB_{id}}(Q) \setminus Q(can(\overline{R'}; \overline{DB'}))\downarrow = A.$$

Using Lemma 2, this transforms to:

$$cert^{M_{id}}_{DB_{id}}(Q) \setminus cert^{\overline{M'}}_{\overline{DB'}}(Q) = A$$

and combined with the fact that

$$cert^{\overline{M'}, M_{n+1}}_{\overline{DB'}, DB_{n+1}}(Q) \setminus cert^{\overline{M'}}_{\overline{DB'}}(Q) \neq \emptyset$$

yields

$$cert^{\overline{M'}, M_{n+1}}_{\overline{DB'}, DB_{n+1}}(Q) \setminus cert^{M_{id}}_{DB_{id}}(Q) \neq \emptyset \qquad (2)$$

(since all tuples in A contain fresh constants and thus they can not appear in $cert^{\overline{M'}, M_{n+1}}_{\overline{DB'}, DB_{n+1}}(Q)$).

We will now compare $cert^{\overline{M'}, M_{n+1}}_{\overline{DB'}, DB_{n+1}}(Q)$ and $cert^{M_{id}, M_{n+1}}_{DB_{id}, DB_{n+1}}(Q)$, which by Lemma 2 equal to $Q(can(\overline{R'}, R_{n+1}; \overline{DB'}, DB_{n+1}))\downarrow$ and $Q(can(\overline{R_{id}}, R_{n+1}; \overline{DB_{id}}, DB_{n+1}))\downarrow$, respectively. As previously, by the construction of DB_{id} from $\overline{DB}$ it follows that

$$cert^{M_{id}, M_{n+1}}_{DB_{id}, DB_{n+1}}(Q) \supseteq cert^{\overline{M'}, M_{n+1}}_{\overline{DB'}, DB_{n+1}}(Q). \qquad (3)$$

Equations 2 and 3 yield:

$$cert^{M_{id}, M_{n+1}}_{DB_{id}, DB_{n+1}}(Q) \setminus cert^{M_{id}}_{DB_{id}}(Q) \neq \emptyset.$$

Hence $NComp^{Q}_{R_{id}}(R_{n+1})$ holds. □

PROOF. (**of Theorem 6**) Consider the registration R and a future registration R_f. In the presence of both registrations, Q must have at least one certain answer, therefore it must have a match against the canonical instance as defined in the proof of Theorem 5. The image of Q's atoms consists of tuples in the canonical instance corresponding to R, R_f (can_{R,R_f}). These tuples couldn't all have existed in the canonical instance of R alone, can_R, (since Q would have been self-sufficient), or in that of R_f, can_{R_f}, (since R wouldn't be now complementary with R_f). There must therefore exist a nonempty set of tuples in can_R such that some k attributes in them do not have the value needed to serve as image of Q. These attributes are turned into the desired values once the registration R_f is added. The only way these values can be changed by R_f is if they are Skolems in can_R. It is easy to see that, since each registration introduces its own Skolems, the only way R_f can affect Skolems introduced by R is by interference through PKs. That is, R_f can change some Skolem Sk into some value v only if can_R contains some tuple t in some relation P whose attribute set X is a key, such that none of $t[X]$ are Skolems, and such that for some attribute $A \notin X$, $t[A] = Sk$. R_f must insert into can_{R,R_f} a new P-tuple t' such that $t[X] = t'[X]$ and $t'[A] = v$. Note that P does not have to be mentioned by Q, since Skolems can occur in several places in can_R. Sk may occur in a tuple which will serve as image for Q once R_f arrives, and also in some (potentially distinct) P-tuple.

The value v may come from (i) the source f, or (ii) it may be moved from another tuple in can_R.

Case (i) requires source f to contribute data via some mapping of form $D[Y] \subseteq P[X, A]$, where $D[Y]$ is the query projecting D on its attribute set Y and $|Y| = |X| + 1$. Case (ii) requires some mapping of form $D[X, Y] \subseteq \sigma_{A=B}(N[Y, B] \times P[X, A])$, where relation N is not necessarily distinct from P, Y is the key for N, and B is some non-key attribute of N. The mapping "loads" the value $v = t''[B]$ from some N-tuple $t'' \in can_R$ by providing an N-tuple t''' which coincides with t'' on the key value and which holds a Skolem in the B attribute. Due to the PK on N, $t''[B] = t'''[B] = v$. v is then "stored" into $t[A]$, because the mapping provides a P-tuple t' agreeing with t on the key, and holding v in its A attribute.

The set of all k mappings as above can be summarized into a single mapping stating the inclusion of the Cartesian product of the left-hand queries into the Cartesian product of the right-hand queries. The size of D and of the mapping queries follows. □