An ID-Logic Formalization of the Composition of Autonomous Databases

Bert Van Nuffelen[1], Ofer Arieli[2], Alvaro Cortés-Calabuig[1], and Maurice Bruynooghe[1]

[1] Department of Computer Science, Katholieke Universiteit Leuven, Belgium
{bertv,alvaro,maurice}@cs.kuleuven.ac.be
[2] Department of Computer Science, The Academic College of Tel-Aviv, Israel
oarieli@mta.ac.il

Abstract. We introduce a declarative approach for a coherent composition of autonomous databases. For this we use ID-logic, a formalism that extends classical logic with inductive definitions. We consider ID-logic theories that express, at the same time, the two basic challenges in database composition problems: relating different schemas of the local databases to one global schema (*schema integration*) and amalgamating the distributed and possibly contradictory data to one consistent database (*data integration*). We show that our framework supports different methods for schema integration (as well as their combinations) and that it provides a straightforward way of dealing with inconsistent data. Moreover, this framework facilitates the implementation of database repair and consistent query answering by means of a variety of reasoning systems.

1 Introduction and Motivation

Composition of information that arrives from different data-sources is a major challenge of information systems and its importance has been recognized by many researches. The works on this subject may be divided to two types according to their objectives:

- Systems for *schema integration*, in which the main goal is to provide a uniform vocabulary to which the various vocabularies of the data-sources can be mapped (see, for instance, [6, 10, 21, 26, 30]).
- Systems for *data integration*, in which the major concern is to resolve contradictions that may occur when the distributed data is amalgamated (see, e.g., [1–5, 8]).

In this work we consider a framework that handles both these tasks *at the same time*. To illustrate (and motivate) this, consider the following situation:

Example 1. Given two data sources. One source stores information about all the students that were enrolled for the first time during 2004, and the other source contains information about all the students whose first year of enrollment is 2005. The encoding of such databases might be the following:

$$\mathcal{DB}_1 = (\{st04(\cdot)\}, \{st04(john)\}),$$
$$\mathcal{DB}_2 = (\{st05(\cdot)\}, \{st05(mary), st05(john)\}).$$

Here, $\mathcal{DB}_1$ encodes the fact that John is a student first enrolled in 2004, and $\mathcal{DB}_2$ encodes the facts that John and Mary are students first enrolled in 2005.

Assume further that there is a set of (global) integrity constraints, stating that the first year of enrollment is unique for every student, and that someone must have been enrolled already in 2003. That is,

$$\mathcal{IC} = \left\{ \begin{array}{l} \forall XYZ\big(enr(X,Y) \wedge enr(X,Z) \rightarrow Y = Z\big) \\ \exists X enr(X, 2003) \end{array} \right\}.$$

A proper integration system is expected to take the following actions in this case:

a) remove the information that John was first enrolled in 2004 or the information that he was first enrolled in 2005 (*but not both*!)
b) insert a fact that some student *other than John or Mary* was first enrolled in 2003.

Note that in order to accomplish this task, the mediator system should be able to cope with the following challenges:

1. relate the different terminologies of the local databases and that of the global integrity constraints,
2. identify inconsistencies (i.e., violations of integrity constraints) and resolve them by making some (minimal amount of) changes in the unified database,
3. search for solutions that may lie outside the active domain of the databases (in order, e.g., to satisfy the second constraint above).

We call the whole process described above *database composition*. In this paper we present a composition system that generalizes the schema integration process introduced in [30] and at the same time enhances the abductive system for data integration introduced in [3], by using *the same logical formalism*. The outcome is a uniform solution for database composition, which to the best of our knowledge is more comprehensive (with respect to the expressive power of the underlying language, the amount of integration methods that are supported, and the mediation capabilities mentioned in Example 1) than any other approach implemented by similar systems.

2 Preliminaries

2.1 ID-logic

ID-logic [12, 13] is a knowledge representation formalism extending classical first-order logic with non-monotone inductive definitions. It is motivated by the realization that (inductive) definitions are a distinctive form of human knowledge and are often encountered in mathematical practice. At the same time, inductive definitions cannot easily be expressed in classical logic (for instance, the transitive closure of a graph is one of the simplest concepts typically defined by induction but it is well-known that this concept cannot be defined in first-order logic).

The language of ID-logic uses the well-founded semantics [28] to extend classical logic with a new 'inductive definition' primitive, and as such it allows even non-monotone inductive definitions to be correctly formalized in an intuitive way. It has also been shown that ID-logic is able to capture the basic ideas behind different concepts and approaches in common-sense reasoning, such as the semantical foundations of situation calculus [15] and description logic [27]. As our goal here is to define composed databases in terms of the distributed ones, together with a description of the merging process, ID-logic is a natural candidate for being the underlying formalism behind such an axiomatization. Below we give the formal definition of this logic.

Definition 1. An ID-logic *theory* $\mathcal{T}$, based on a first-order language $\mathcal{L}$, is a pair ($\mathcal{D}$,$\mathcal{F}$), where $\mathcal{D}$ is a set of definitions D_i $(i = 1, \ldots, n)$ and $\mathcal{F}$ is a set of first-order formulas. A *definition* D is a set of rules of the form $p(\bar{t}) \leftarrow B$, where $p(\bar{t})$ is an atom and B is a first-order formula.

Example 2. The transitive closure of a graph can be defined by the following ID-logic definition: $TransCl(x, y) \leftarrow Edge(x, y) \vee \exists z(TransCl(x, z) \wedge TransCl(z, y))$.

In what follows we refer to $\mathcal{D}$ and $\mathcal{F}$ as the definitions and the constraints (respectively) of $\mathcal{T}$. The predicates occurring in the heads of the rules in the definitions are the *defined* predicates of $\mathcal{D}$. All the other predicates belong to $\mathsf{Open}(\mathcal{D})$, the set of the *open* (*abducible*) predicates of $\mathcal{D}$.

Definition 2. A structure M is a *model* of a definition D iff there exists an interpretation I of $\mathsf{Open}(D)$ such that M is the two-valued well-founded model [28] of D that extends I. A structure M is a model of $\mathcal{D}$ iff M is a model of each $D \in \mathcal{D}$.

Definition 3. A structure M is a *model* of an ID-logic theory $\mathcal{T}$=($\mathcal{D}$,$\mathcal{F}$) iff M is a model of $\mathcal{D}$ and satisfies all formulas of $\mathcal{F}$. The collection of all models of $\mathcal{T}$ is denoted by $mod(\mathcal{T})$.

We say that a formula ψ is *satisfied* by an ID-logic theory $\mathcal{T}$ if there is a model of $\mathcal{T}$ that satisfies ψ. A formula ψ *follows* from $\mathcal{T}$ if every model of $\mathcal{T}$ satisfies ψ.

The following notation will be useful in what follows:

Definition 4. For two ID-logic theories $\mathcal{T}_1, \mathcal{T}_2$ over the same language $\mathcal{L}$, the composed theory $\mathcal{T}_1 \circ \mathcal{T}_2$ is an ID-logic theory $\mathcal{T}$ over $\mathcal{L}$, obtained by the pairwise union of both theories: $\mathcal{T} = \mathcal{T}_1 \circ \mathcal{T}_2 = (\mathcal{D}_1, \mathcal{F}_1) \circ (\mathcal{D}_2, \mathcal{F}_2) = (\mathcal{D}_1 \cup \mathcal{D}_2, \mathcal{F}_1 \cup \mathcal{F}_2)$.

Proposition 1. $mod(\mathcal{T}_1 \circ \mathcal{T}_2) = mod(\mathcal{T}_1) \cap mod(\mathcal{T}_2)$.

2.2 Database Composition

In this section we formally define the problem under consideration and its solutions.

Definition 5. A *database* is a pair $\mathcal{DB} = (\mathcal{L}, \mathcal{D})$, where the *database language* $\mathcal{L}$ is a first-order language based on a vocabulary consisting of the predicate symbols in a fixed database schema S and a finite set Dom of constants representing the elements of the domain of discourse. The *database instance* $\mathcal{D}$ is a finite set of ground atoms in the language $\mathcal{L}$.

The semantics of a database instance is given by the conjunction of the atoms in $\mathcal{D}$ augmented with the *Unique Name Assumption* ($\mathsf{UNA}(\mathsf{Dom})$), i.e., different constants represent different objects, and the *Closed World Assumption* ($\mathsf{CWA}(\mathcal{D})$) that assures that each atom which is not explicitly mentioned in $\mathcal{D}$ is false. Often, the *Domain Closure Assumption* ($\mathsf{DCA}(\mathsf{Dom})$) is also imposed, meaning that all elements of the domain of discourse are named by constants in Dom.[3] The meaning of a database instance under these assumptions is formalized in a model theoretical way by the *least Herbrand model* semantics.

Definition 6. A *composition problem* is a pair $(\mathfrak{D}, \mathfrak{C})$, where the *set of resources* $\mathfrak{D}$ is a non-empty set of (local) databases $\mathcal{DB}_i = (\mathcal{L}_i, \mathcal{D}_i)$, $i = 1, \ldots, n$, and the (global) *integrity constraints* $\mathfrak{C} = (\mathcal{L}_\mathcal{G}, \mathcal{IC})$ consists of a (possibly empty) set $\mathcal{IC}$ of first-order formulae in a first-order (global) language $\mathcal{L}_\mathcal{G}$.

Given a composition problem $(\mathfrak{D}, \mathfrak{C})$, our goal is therefore to construct a composed (global) database $\mathcal{DB}_\mathcal{G} = (\mathcal{L}_\mathcal{G}, \mathcal{D}_\mathcal{G})$ such that its database instance $\mathcal{D}_\mathcal{G}$ contains the translation to $\mathcal{L}_\mathcal{G}$ of data-facts that appear in database instances of elements in $\mathfrak{D}$, provided that these data facts do not violate the integrity constraints in $\mathcal{IC}$. Clearly, this database instance should 'gather' from the local databases as much information as consistently possible, that is, it should be 'as close as possible' to $\bigcup_{i=1,\ldots,n} \mathcal{D}_i$, without violating $\mathcal{IC}$.

Example 3. Consider again Example 1. The composition problem in this case consists of two databases, $\mathcal{DB}_1$ and $\mathcal{DB}_2$, that should be composed under the integrity constraints in $\mathcal{IC}$. When $\mathcal{L}_\mathcal{G} = \{enr(X, Y)\}$ is the language of the composed database (meaning that a student X was first enrolled in a year Y), the two best composed database instances are the following:

$$\mathcal{D}^1_\mathcal{G} = \{\, enr(mary, 2005), enr(john, 2005), enr(u, 2003) \,\}$$
$$\mathcal{D}^2_\mathcal{G} = \{\, enr(mary, 2005), enr(john, 2004), enr(u, 2003) \,\}$$

where u is a (Skolem) constant, different from $mary$ and $john$.

When Dom consists only of $mary$ and $john$, $\mathsf{DCA}(\mathsf{Dom})$ excludes the two solutions above, and the composed database instances are the following (see also Example 5):

$$\mathcal{D}^3_\mathcal{G} = \{\, enr(mary, 2005), enr(john, 2003) \,\}$$
$$\mathcal{D}^4_\mathcal{G} = \{\, enr(john, 2005), enr(mary, 2003) \,\}$$
$$\mathcal{D}^5_\mathcal{G} = \{\, enr(john, 2004), enr(mary, 2003) \,\}$$

Note that other compositions of $\mathcal{DB}_1$ and $\mathcal{DB}_2$ are less intuitive in this case. For instance, $\mathcal{D}^* = \{enr(john, 2003)\}$ requires more revisions in the original assumptions than $\mathcal{D}^3_\mathcal{G}$, and so $\mathcal{D}^3_\mathcal{G}$ is 'closer' to $\mathcal{D}_1 \cup \mathcal{D}_2$ than $\mathcal{D}^*$.

In what follows we shall describe how to define ID-logic theories that *represent* the composition problems under consideration (Section 3) and how to *reason* with these theories in order to *compute* composed databases for the given composition problems (Section 4).

[3] This assumption is sometimes lifted when constants outside Dom should be introduced; see e.g. Examples 3 and 5 below.

3 The Composition Theory

In what follows we assume that all the databases share the same domain Dom (this will simplify the presentation, as the implicit assumptions $\mathsf{UNA}(\mathsf{Dom})$ and $\mathsf{DCA}(\mathsf{Dom})$ can be imposed globally) and that all the languages are mutually distinct (to assure this, one can annotate the predicate names with the source identities).

Definition 7. A *mediator system* for a composition problem $(\mathfrak{D}, \mathfrak{C})$ is a quadruple $\mathcal{M} = \langle \mathcal{L}, \mathsf{CP}, \mathsf{SI}, \mathsf{DI} \rangle$, where:

- $\mathcal{L}$ is the union of the source languages $\mathcal{L}_i$, the global language $\mathcal{L}_\mathcal{G}$, and the auxiliary predicates defined below.
- $\mathsf{CP} = \{\mathsf{S}_1, \ldots, \mathsf{S}_n, \mathsf{IC}\}$, *the composition problem description*, is a set of ID-logic theories S_i encoding the source databases (i.e., the database instances $\mathcal{D}_i$ of the databases in $\mathfrak{D}$), and an ID-logic theory IC encoding the global integrity constraints $\mathcal{IC}$ in $\mathfrak{C}$ (see Section 3.1).
- $\mathsf{SI} = \{\mathsf{M}_1, \ldots, \mathsf{M}_n, \mathsf{K}\}$, *the schema integration specification*, is a set of ID-logic theories M_i encoding the relations between the source languages $\mathcal{L}_i$ and the global language $\mathcal{L}_\mathcal{G}$, and an ID-logic theory K encoding additional information about the relations among the schemas (see Section 3.2).
- $\mathsf{DI} = \{\mathsf{Comp}, \mathsf{Trans}\}$, *the data integration specification*, is a set of ID-logic theories that specify how to make the combined database consistent with the set $\mathcal{IC}$ of integrity constraints in $\mathfrak{C}$ (see Section 3.3).

A *composition theory* for a mediator system $\mathcal{M}$ is an ID-logic theory $\mathcal{T}_\mathcal{M} = \mathcal{T}_\mathsf{SI} \circ \mathcal{T}_\mathsf{DI}$ in $\mathcal{L}$, where $\mathcal{T}_\mathsf{SI} = \mathsf{S}_1 \circ \cdots \circ \mathsf{S}_n \circ \mathsf{M}_1 \circ \cdots \circ \mathsf{M}_n \circ \mathsf{K}$ and $\mathcal{T}_\mathsf{DI} = \mathsf{IC} \circ \mathsf{Comp} \circ \mathsf{Trans}$.

In the rest of this section we consider in further details the components of a mediator.

3.1 Representation of the Composition Problem

Let $(\{(\mathcal{L}_1, \mathcal{D}_1), \ldots, (\mathcal{L}_n, \mathcal{D}_n)\}, (\mathcal{L}_\mathcal{G}, \mathcal{IC}))$ be a composition problem.

- The encoding of a source database instance $\mathcal{D}_i$ $(1 \leq i \leq n)$ in CP is the ID-logic theory $\mathsf{S}_i = (\{\{D_i\}\}, \emptyset)$, where D_i is the enumeration of the facts in $\mathcal{D}_i$.
- The encoding of the global integrity constraints $\mathcal{IC}$ in CP is the ID-logic theory $\mathsf{IC} = (\emptyset, \{IC\})$, where IC is obtained by interpreting the integrity constraints $\mathcal{IC}$ as an enumeration of formulae in $\mathcal{L}_\mathcal{G}$, and substituting every occurrence of a predicate $p(\bar{t})$ by $fact(p(\bar{t}))$.[4]

3.2 Representation of Schema Integration

The component SI of a mediator system describes the relationships between the source languages $\mathcal{L}_i$ and the global language $\mathcal{L}_\mathcal{G}$. These relationships are expressed in the form of (inductive) definitions, taking into account the ontological relationships between the predicates and the actual knowledge of the source. For a proper description of such relations, one has to take into consideration the following cases:

[4] The need of this substitution will become apparent in Section 3.3.

1. Suppose that the set of rules $\{p(\overline{t}) \leftarrow B_i \mid i = 1 \ldots k\}$ only partially defines a predicate p. A complete definition can be obtained by adding a rule $p(\overline{s}) \leftarrow p^*(\overline{s})$, in which the auxiliary open predicate p^* represents all the tuples in p that are not defined by any of the bodies B_i.
2. Sometimes a body of a rule $p(\overline{t}) \leftarrow B$ is too general, i.e., it includes tuples not intended to be in the predicate p. In this case it is possible to add to the body B an auxiliary open predicate p^s that filters the extraneous tuples. The completed rule in this case is $p(\overline{t}) \leftarrow B \wedge p^s(\overline{t})$.

Auxiliary predicates help to relate different languages and to complete partial information. As such, they serve as open (abducible) predicates.

Definition 8. A *language mapping* from a language $\mathcal{L}_1$ to a language $\mathcal{L}_2$ is an ID-logic theory $(\mathsf{R}_{1\to 2}, \mathsf{IC}_{1,2})$, where $\mathsf{R}_{1\to 2}$ defines the predicates of $\mathcal{L}_2$ in terms of the predicates of $\mathcal{L}_1$ and the necessary auxiliary predicates, and $\mathsf{IC}_{1,2}$ formulates generic integrity constraints on the auxiliary predicates.

In terms of the last definition, the elements $\mathsf{M}_1, \ldots, \mathsf{M}_n$ of SI are the language mappings between the source languages and the global language. The first component of each one of these mappings M_i may be of one of the following two forms:

- $\mathsf{R}_{i\to\mathcal{G}}$. In this case the predicates of global language $\mathcal{L}_\mathcal{G}$ are defined in terms of the predicates of a local language ($\mathcal{L}_i$ in this case). This mapping corresponds to the *Global as View* (GAV) approach in schema integration (see, e.g., [26]).
- $\mathsf{R}_{\mathcal{G}\to i}$. In this case the language $\mathcal{L}_i$ of a local source is defined in terms of the global language $\mathcal{L}_\mathcal{G}$. This mapping corresponds to the *Local as View* (LAV) approach in schema integration (see [21]).

Most of the systems that are introduced in the literature support only one type of the schema integration methods described above. In our case it is clear that one may use both kinds of language mappings in the same theory, and so imitate both GAV and LAV by the same mediator system.

Example 4. Consider the source languages $\mathcal{L}_1 = \{st05(\cdot)\}$ and the global language $\mathcal{L}_\mathcal{G} = \{enr(\cdot,\cdot)\}$, where $st05(\cdot)$ represents (some) students that are first enrolled in 2005 and $enr(\cdot,\cdot)$ represents all students with their first year of enrollment. According to the LAV approach, a mapping $\mathsf{M}_1 = (\mathsf{R}_{\mathcal{G}\to 1}, \mathsf{IC}_{\mathcal{G},1})$ between these languages could be the following ID-logic theory:

$$\begin{pmatrix} \{\{st05(X) \leftarrow enr(X, 2005) \wedge st05^s(X).\}\}, \\ \{\forall X(st05^s(X) \rightarrow enr(X, 2005))\} \end{pmatrix},$$

where $st05^s(\cdot)$ represents the students known by the source and the integrity constraint expresses that this is a subset of all students enrolled in 2005. Similarly, a GAV mapping $\mathsf{M}_1 = (\mathsf{R}_{1\to\mathcal{G}}, \mathsf{IC}_{1,\mathcal{G}})$ could be, e.g., the following ID-logic theory:

$$\begin{pmatrix} \{\{enr(X,Y) \leftarrow (st05(X) \wedge Y = 2005) \vee enr^*(X,Y).\}\}, \\ \forall XY(enr^*(X,Y) \rightarrow \neg(st05(X) \wedge Y = 2005)) \end{pmatrix},$$

where $enr^*(\cdot,\cdot)$ represents the students not known by the source, and the integrity constraint imposes that $enr^*(\cdot,\cdot)$ does not duplicate the information from the source.

The remaining element in SI, denoted by K, contains some additional information about the predicates and the interrelations. In the above example, for instance, one may know that the source has complete information about all students enrolled in 2005, so now $st05(\cdot)$ should represent *all* the students first enrolled in 2005. This can be expressed by the constraint $\forall X enr(X, 2005) \leftrightarrow st05(X)$. In the LAV approach it implies that the relation $st05^s(\cdot)$ is empty (and could be omitted in the language mapping); in the GAV approach, it implies that $enr^*(\cdot,\cdot)$ cannot have tuples with the year 2005.

3.3 Representation of Data Integration

The data integration specification DI is a specification of how the database instances should be integrated such that no integrity constraint will be violated. It makes sure that if the set $\mathcal{D} = \bigcup \mathcal{D}_i$ of all the local databases (translated to the global schema) preserves all the integrity constraints in $\mathcal{IC}$ (notation: $\mathcal{D} \models \mathcal{IC}$),[5] then the global database instance $\mathcal{D}_\mathcal{G}$ will be equal to this set. Otherwise, some (minimal amount of) data-facts should be inserted to- or retracted from this union in order to restore its consistency with respect to $\mathcal{IC}$. In other words, $\bigcup \mathcal{D}_i$ should be 'repaired':

Definition 9. [1] A *repair* of a database instance $\mathcal{D}$ with respect to a set $\mathcal{IC}$ of integrity constraints is a pair $(\mathsf{Insert}, \mathsf{Retract})$, such that: (1) $\mathsf{Insert} \cap \mathcal{D} = \emptyset$, (2) $\mathsf{Retract} \subseteq \mathcal{D}$,[6] and (3) $((\mathcal{D} \cup \mathsf{Insert}) \setminus \mathsf{Retract}) \models \mathcal{IC}$.

Intuitively, Insert is a set of elements that should be inserted to $\mathcal{D}$ and Retract is a set of elements that should be removed from $\mathcal{D}$ in order to assure that $\mathcal{D}$ is consistent with $\mathcal{IC}$. This is represented by the following ID-logic theory (the *composer*):

$$\mathsf{Comp} = \left(\begin{array}{l} \left\{ \left\{ \begin{array}{l} fact(X) \leftarrow db(X) \wedge \neg retract(X). \\ fact(X) \leftarrow insert(X). \end{array} \right\} \right\}, \\ \left\{ \begin{array}{l} \forall X \neg(insert(X) \wedge db(X)) \\ \forall X\ db(X) \leftarrow retract(X) \end{array} \right\} \end{array} \right),$$

where $db(X)$ denotes in the global language that X is a data-fact, and $fact(X)$ denotes that the data-fact X should appear in the global database. Here, $insert$ and $retract$ are open (abducible) predicates that describe repairs. The last two formulas of Comp assure that conditions (1) and (2) in Definition 9 will hold.[7]

As there are usually many ways to repair a given database, it is often convenient to make preferences among the possible repairs, and consider only the most preferred ones. Below are two common preference criteria for preferring a repair $(\mathsf{Insert}, \mathsf{Retract})$ over a repair $(\mathsf{Insert}', \mathsf{Retract}')$:

Definition 10. Let $(\mathsf{Insert}, \mathsf{Retract})$ and $(\mathsf{Insert}', \mathsf{Retract}')$ be two repairs of a database.

- *set inclusion preference criterion*:
  $(\mathsf{Insert}, \mathsf{Retract}) \leq_i (\mathsf{Insert}', \mathsf{Retract}')$, if $\mathsf{Insert} \subseteq \mathsf{Insert}'$ and $\mathsf{Retract} \subseteq \mathsf{Retract}'$

[5] That is, every formula in $\mathcal{IC}$ is satisfied in the least Herbrand model of $\mathcal{D}$.
[6] Note that by conditions (1) and (2), $\mathsf{Insert} \cap \mathsf{Retract} = \emptyset$.
[7] The third condition of Definition 9 is imposed by the theory IC as defined in Section 3.1.

– *minimal cardinality preference criterion*:
$(\mathsf{Insert}, \mathsf{Retract}) \leq_c (\mathsf{Insert}', \mathsf{Retract}')$, if $|\mathsf{Insert}|+|\mathsf{Retract}| \leq |\mathsf{Insert}'|+|\mathsf{Retract}'|$

If $\mathcal{D} \models \mathcal{IC}$, then $(\emptyset, \emptyset)$ is the *only* $\leq_i$- and $\leq_c$-preferred repair of $\mathcal{D}$, as expected.

The second component of DI is the *translator*, Trans. It represents all the translated data-facts in terms of one (global) language:

$$\mathsf{Trans} = (\{\{db(p_1(\overline{t})) \leftarrow p_1(\overline{t}), \ldots, db(p_{\mathcal{G}}(\overline{t})) \leftarrow p_{\mathcal{G}}(\overline{t})\}\}, \emptyset)$$

where $p_1, \ldots, p_{\mathcal{G}}$ are the predicates of the global language $\mathcal{L}_{\mathcal{G}}$.

The translator reifies the database predicates, i.e., it converts the database facts into terms of the predicate db.[8] For reducing notational complexity, we use the same symbols for the predicates and their reifications (e.g., in Trans above, the p_i appearing on the left-hand side of the implications are the reified symbols of the predicates p_i on the right-hand side of the same implications). Note that the predicate fact of the composer represents which one on these new 'fact'-terms appears in the composed database.

3.4 Back to the Canonical Example

A composition theory for Example 1 may be the following:

• The schema integration theory $\mathcal{T}_{\mathsf{SI}}$ is a composition of S_1, S_2,M_1, and M_2 (in this case K is assumed to be empty):

$$\left(\begin{array}{l} \left\{\begin{array}{l} \{\, st04(john). \,\} \\ \left\{\begin{array}{l} st05(mary). \\ st05(john). \end{array}\right\} \\ \left\{\begin{array}{l} enr(X,Y) \leftarrow st04(X) \wedge Y = 2004. \\ enr(X,Y) \leftarrow enr_1^*(X,Y). \end{array}\right\} \\ \left\{\begin{array}{l} enr(X,Y) \leftarrow st05(X) \wedge Y = 2005. \\ enr(X,Y) \leftarrow enr_2^*(X,Y). \end{array}\right\} \end{array}\right\}, \\ \left\{\begin{array}{l} \forall XY(enr_1^*(X,Y) \rightarrow \neg(st04(X) \wedge Y = 2004)). \\ \forall XY(enr_2^*(X,Y) \rightarrow \neg(st05(X) \wedge Y = 2005)). \end{array}\right\} \end{array}\right)$$

The definitions of $\mathcal{T}_{\mathsf{SI}}$ are from the theories S_1, S_2, M_1, and M_2; the constraints are from M_1 and M_2.

• The data integration theory $\mathcal{T}_{\mathsf{DI}}$ is the following composition of IC, Comp and Trans:

$$\left(\begin{array}{l} \left\{\begin{array}{l} \left\{\begin{array}{l} fact(X) \leftarrow db(X) \wedge \neg retract(X). \\ fact(X) \leftarrow insert(X). \end{array}\right\} \\ \{\, db(enr(X,Y)) \leftarrow enr(X,Y). \,\} \end{array}\right\}, \\ \left\{\begin{array}{l} \forall X \neg(insert(X) \wedge db(X)). \\ \forall X db(X) \leftarrow retract(X). \\ \forall XYZ\ (fact(enr(X,Y)) \wedge fact(enr(X,Z)) \rightarrow Y = Z). \\ \exists X\, fact(enr(X, 2003)). \end{array}\right\} \end{array}\right)$$

The definitions of $\mathcal{T}_{\mathsf{DI}}$ are from Comp (the first two) and Trans (the third one); the constraints of $\mathcal{T}_{\mathsf{DI}}$ are from Comp (the first two) and IC (the last two).

[8] See [7] for a description of reifications in the context of knowledge representation.

4 Reasoning with Composed Databases

Query answering is probably the main task of a mediator system. In order to compute answers from a composition theory in our context, the underlying ID-logic theory should be converted to an equivalent theory in answer set programming (ASP) or abductive logic programming (ALP), which are the two available methods of reasoning with ID-logic theories. By this, corresponding off-the-shelf solvers (such as the ASP systems dlv [17] and sModels [25], or the ALP solver $\mathcal{A}$system [3, 20, 29]) can be utilized for the query answering. Below we consider both options.

Abductive Logic Programming

An ID-logic theory can be converted to an equivalent abductive normal logic program (see [29] for a detailed description of this process), and then processed by solvers for reasoning with abductive theories. We have implemented our approach for database composition by such a solver, called $\mathcal{A}$system [3, 20, 29].[9] The $\mathcal{A}$system computes interpretation of the abducible predicates of a given ID-logic theory[10] by executing the abductive refutation procedure SLDNFA (an extension of **SLD**-resolution for programs with **N**egation as **F**ailure operators and **A**bducible predicates; see [14]). It therefore constructs an explanation formula $\mathcal{E}$, in terms of the open predicates of $\mathcal{T}$, that entails a query $\mathcal{Q}$. Formally:

Definition 11. An *abductive solution* for an ID-logic theory $\mathcal{T}$ and a query $\mathcal{Q}$ is a pair $(\Delta, \mathcal{E})$, where Δ is a set of abducible atoms and $\mathcal{E}$ is the conjunction of the elements in Δ, such that $\mathcal{T} \models \exists \overline{x}\, \mathcal{E}(\overline{x})$ and $\mathcal{T} \models \forall \overline{x}\, (\mathcal{E} \rightarrow \mathcal{Q})(\overline{x})$.

In our case, the computed explanation formula $\mathcal{E}$ describes a class of models of $\mathcal{T}$. When $\mathcal{E}$ is *true*, the query is satisfiable with respect to all the models. When the $\mathcal{A}$system is unable to find an abductive solution for $\mathcal{Q}$, then $\mathcal{T} \models \forall(\neg\mathcal{Q})$.

SLDNFA is a sound proof procedure for abductive normal logic programs under the (three-valued) completion semantics. Under certain conditions it is also complete (see [14]) and always terminates (see [31]).[11] These properties are inherited by the $\mathcal{A}$system, which is also equipped with a component that discards non-optimal solutions, called the *optimizer*. Given a preference criterion on the solution space, the optimizer computes only the most-preferred (abductive) solutions by pruning 'on the fly' those branches of the search tree that lead to solutions that are worse than others that have already been computed. This is actually a branch and bound 'filter' that speeds-up execution and makes sure that only the desired solutions will be obtained. If the preference criterion is a pre-order (as those of Definition 10), the optimizer is *complete*, that is, it can compute all the optimal solutions (as illustrated in Example 5 below). We refer to [3, 29] for a detailed description of the abductive inference process implemented by the $\mathcal{A}$system.

[9] See also `http://www.cs.kuleuven.ac.be/~dtai/kt/systems-E.shtml`.

[10] These interpretations uniquely determine the models of the theory; see Definitions 2 and 3.

[11] This is the case, for instance, when the underlying logic programs are hierarchical, or abductive non-recursive

Example 5. Consider again the composition theory $\mathcal{T}_{\mathcal{M}} = \mathcal{T}_{\mathsf{SI}} \circ \mathcal{T}_{\mathsf{DI}}$ given in Section 3.4 for the running example. By $\mathcal{T}_{\mathsf{SI}}$ we derive the atoms $enr(john, 2004)$, $enr(john, 2005)$, $enr(mary, 2005)$, which is the translation of the local data in terms of the global language. Now, both integrity constraints in $\mathcal{IC}$ are violated, so the data should be repaired. Indeed, by $\mathcal{T}_{\mathsf{DI}}$ and $\leq_c$-optimizer, the following repairs are obtained:

- $retract(enr(john, 2004))$, $insert(enr(u, 2003))$ for $u \notin \{john, mary\}$,
- $retract(enr(john, 2005))$, $insert(enr(u, 2003))$ for $u \notin \{john, mary\}$.

With an $\leq_i$-optimizer, three more solutions are obtained:

- $retract(enr(john, 2004)), retract(enr(john, 2005))$, $insert(enr(john, 2003))$
- $retract(enr(mary, 2005)), retract(enr(john, 2004))$, $insert(enr(mary, 2003))$
- $retract(enr(mary, 2005)), retract(enr(john, 2005))$, $insert(enr(mary, 2003))$

The global database instances in this case are, respectively,

- $\{enr(u, 2003)), enr(john, 2005)), enr(mary, 2005))\}$,
- $\{enr(u, 2003)), enr(john, 2004)), enr(mary, 2005))\}$.
- $\{enr(john, 2003)), enr(mary, 2005))\}$.
- $\{enr(mary, 2003)), enr(john, 2005))\}$.
- $\{enr(mary, 2003)), enr(john, 2004))\}$.

The first two solutions are obtained since the $\mathcal{A}$system does not impose the domain closure assumption $\mathsf{DCA}(\mathsf{Dom})$. This allows to compute solutions outside the least Herbrand model of the problem, and so to suggest explanations for database inconsistency, which could not be captured otherwise.

Answer Set Programming

An ID-logic theory in which each variable occurring in a formula is delimited by a range (domain) relation is called a strongly range-restricted theory. In [23] it is shown that strongly range-restricted ID-logic theories can be transformed to equivalent logic programs under the stable model semantics. This implies that ASP solvers may also be incorporated for reasoning with ID-logic-based mediator systems.

ALP and ASP have a lot in common, and they are often viewed as different variations of the same paradigm. In particular, both approaches compute (minimal) models of the theory. Still, in opposed to the ALP approach, which is a local inference procedure that selects only the information which is relevant for the query, ASP is a global reasoning tool for processing ground theories. ASP requires finite domains and imposes the domain closure axiom. As a consequence, this method is conceptually less suitable for reasoning about tasks which need to go outside the Herbrand space, and it is inherently less scalable (in terms of the size of the databases) than ALP.[12] Note, however, that grounding to finite theories ensures the termination of the ASP computations.

[12] In Example 5, for instance, ASP solvers will not produce the two repairs that contain the Skolem constant u.

A simple work around that allows to lift the domain closure assumption posed by the ASP approach is to iteratively add new Skolem constants to the Herbrand domain and check for solutions in the new domains. However, a problem with this naive generate-and-test approach is that one needs a criterion to know whether all the solutions have been found. In case of a preference condition, the number of the Skolems constants used in a solution allows to derive a lower bound on its cost, which could likely be the basis for a terminating condition. Note that in certain cases (e.g., when no insertions are allowed to restore consistency), all the solutions are already inside the Herbrand domain, and so ASP and ALP solvers will terminate.

5 Concluding Remarks

In this paper we have developed a formal declarative foundation for representing and reasoning with independent databases that contain information about a common domain, but may have different schemas and may contradict each other. This problem, known as database composition, is represented by ID-logic theories that mediate among the ontologies of the sources, and resolve contradictions between local information and global constraints.

It is important to note that this paper is mainly concerned with the *representation* aspects of this problem, showing that different ingredients of it can be expressed in a natural and intuitive way by a single logical formalism. In this context, we have elaborated on the following advantages of our approach:

- The underlying logic extends classical logic with inductive definitions, and as such it can be viewed as an expressive form of a description logic.[13] In particular, our approach is more expressive than similar approaches that are based on description logics.
- Unlike some other approaches of data integration, no syntactical restriction is imposed on the integrity constraints (which can be *any* set of first-order formulas).
- The inherent modularity of ID-logic allows to represent different aspects of the same problem (that is, schema and data integration) in different modules. In other formalisms (e.g., ALP, ASP, or description logics) these aspects are mixed in one complex theory.
- The representation methodology is tolerant to the structure of the autonomous databases. For instance, the composition theory described in Section 3 may be easily modified in case that a certain source of information is added or dropped.
- Different types of schema integration are supported (e.g., GAV and LAV), as well as their combinations and corresponding extensions, such as the generalized LAV approach (GLAV) [19] and Both-as-View approach (BAV) [24].

Other benefits of our framework, which are related to *computation* aspects of data integration, are hinted in Section 4. Below we list two of them: [14]

[13] See [27] for more information about the relation between ID-logic and description logics.

[14] The full details are beyond the scope (and the space limitations) of this paper. Still, as noted in Section 4, the properties below are obtained by straightforward generalizations or adaptations to our context of the techniques described in [3] (for ALP) and [23] (for ASP).

– Different types of query answering are supported. I.e., *skeptical query answering* (also called *certain answering*), in which a query is true iff it is entailed by *every* composed database, or *credulous query answering*, in which a query is true iff it is entailed by *some* composed database.
– Different notions of optimal repairs (e.g., set inclusion, minimal cardinality, minimization of the amount of inserted data-facts, and so forth) are dealt with through preferential semantics.

As noted above, the mediator systems considered here may be implemented by a variety of off-the-shelf solvers. Several other implementations have been introduced for the kind of problems we are dealing with here. Among the implementations of schema integration are the abductive GAV-based system of [9] and the LAV-based information manifold system of [21]. Systems for data integration are, e.g., BReLS [22] and the data repair system of [18]. We provide here a uniform framework for *both* kinds of integrations. Recently, some other implementations of schema and data integration have been introduced, e.g., [8] and [10]. These approaches are based on representation platforms that are more restricted than ours, as they implement only particular kinds of schema mapping styles, limit the syntactic structure of the integrity constraints, and impose the domain closure assumption.

A detailed investigation of the properties of particular computational models for our framework is beyond the scope of the current paper. We refer to [11, 16] for a discussion on some computational aspects (e.g, complexity and decidability) of the kinds of problems considered here. Other topics for future elaboration include incorporation of temporal information in the databases, handling of conflicts among integrity constraints, and a study of other merging policies (such as merging by majority vote).

References

1. M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. PODS'99*, pages 68–79, 1999.
2. M. Arenas, L. Bertossi, and J. Chomicki. Answer sets for consistent query answering in inconsistent databases. *Theory and Practice of Logic Programming*, 3(4–5):393–424, 2003.
3. O. Arieli, M. Denecker, B. Van Nuffelen, and M. Bruynooghe. Coherent integration of databases by abductive logic programming. *Artif. Intelligence Research*, 21:245–286, 2004.
4. O. Arieli, M. Denecker, B. Van Nuffelen, and M. Bruynooghe. Database repair by signed formulae. In *Proc. FoIKS'04*, LNCS 2942, pages 14–30, 2004.
5. C. Baral, S. Kraus, and J. Minker. Combining multiple knowledge bases. *IEEE Trans. on Knowledge and Data Enginnering*, 3(2):208–220, 1991.
6. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
7. R. Brachman and H. Levesque. *Knowledge representation and Reasoning*. Morgan Kaufmann, 2004.
8. L. Bravo and L. Bertossi. Logic programming for consistently querying data integration systems. In *Proc. IJCAI'03*, pages 10–15, 2003.
9. S. Bressan, C. H. Goh, K. Fynn, M. Jakobisiak, K. Hussein, H. Kon, T. Lee, S. Madnick, T. Pena, J. Qu, A. Shum, and M. Siegel. The context interchange mediator prototype. In *Proc. PODS'97*, pages 525–527. ACM, 1997.

10. A. Calì, D. Calvanese, G. De Giacomo, and M. Lanzerini. Data integration under integrity constraints. *Information Systems*, 29(2):147–163, 2004.
11. A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. PODS'03*, pages 260–271. ACM, 2003.
12. M. Denecker. Extending classical logic with inductive definitions. In *Proc. CL'2000*, LNCS 1861, pages 703–717. Springer, 2000.
13. M. Denecker, M. Bruynooghe, and V. Marek. Logic programming revisited: logic programs as inductive definitions. *ACM Trans. on Computational Logic*, 2(4):623–654, 2001.
14. M. Denecker and D. De Schreye. SLDNFA an abductive procedure for abductive logic programs. *Journal of Logic Programming*, 34(2):111–167, 1998.
15. M. Denecker and E. Ternovska. Inductive situation calculus. In *Proc. KR'04*, pages 545–553. Morgan Kaufmann Publishers, 2004.
16. T. Eiter, M. Fink, G. Greco, and D. Lembo. Efficient evaluation of logic programs for querying data integration systems. In *Proc. ICLP'03*, LNCS 2916, pages 163–177. Springer, 2003.
17. T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The KR system dlv: Progress report, comparisons and benchmarks. In *Proc. KR'98*, pages 406–417, 1998.
18. E. Franconi, A. Palma, N. Leone, D. Perri, and F. Scarcello. Census data repair: A challenging application of disjunctive logic programming. In *Proc. LPAR'01*, LNCS 2250, pages 561–578. Springer, 2001.
19. M. Friedman, A. Y. Levy, and T. D. Millstein. Navigational plans for data integration. In *Proc. 16th. AAAI*, pages 67–73, 1999.
20. A. Kakas, B. Van Nuffelen, and M. Denecker. A-system : Problem solving through abduction. In *Proc. IJCAI'01*, pages 591–596, 2001.
21. A. Levy, A. Rajaraman, and Ordille J.J. Querying heterogeneous information sources using source descriptions. In *Proc. VLDB-96*, pages 251–262, 1996.
22. P. Liberatore and M. Schaerf. BReLS: A system for the integration of knowledge bases. In *Proc. KR'2000*, pages 145–152. Morgan Kaufmann Publishers, 2000.
23. M. Mariën, D. Gilis, and M. Denecker. On the relation between ID-logic and answer set programming. In *Proc. JELIA'04*, LNCS 3229, pages 108–120. Springer, 2004.
24. P. McBrien and A. Poulovassilis. Data integration by bi-directional schema transformation rules. In *Proc. 19th ICDE*. IEEE, 2003.
25. P. Simons, I. Niemelä, and T. Soinien. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1–2), 2002.
26. J. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.
27. K. Van Belleghem, M. Denecker, and D. De Schreye. A strong correspondence between description logics and open logic programming. In *Proc. ICLP'97*, pages 346–360, 1997.
28. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
29. B. Van Nuffelen. *Abductive constraint logic programming*. PhD thesis, Katholieke Universiteit Leuven, 2004.
30. B. Van Nuffelen, A. Cortés-Calabuig, M. Denecker, O. Arieli, and M. Bruynooghe. Data integration using ID-logic. In *Proc. CAiSE'04*, LNCS 3084, pages 67–81. Springer, 2004.
31. S. Verbaeten. Termination analysis for abductive general logic programs. In *Proc. ICLP'99*, pages 365–379. MIT Press, 1999.