Simplification of integrity constraints for data integration

Henning Christiansen and Davide Martinenghi

Roskilde University, Computer Science Dept.
P.O.Box 260, DK-4000 Roskilde, Denmark
E-mail: {henning,dm}@ruc.dk

Abstract. When two or more databases are combined into a global one, integrity may be violated even when each database is consistent with its own local integrity constraints. Efficient methods for checking global integrity in data integration systems are called for: answers to queries can then be trusted, because either the global database is known to be consistent or suitable actions have been taken to provide consistent views. The present work generalizes simplification techniques for integrity checking in traditional databases to the combined case. Knowledge of local consistency is employed, perhaps together with given *a priori* constraints on the combination, so that only a minimal number of tuples needs to be considered. Combination from scratch, integration of a new source, and absorption of local updates are dealt with for both the local-as-view and global-as-view approaches to data integration.

1 Introduction

Data integration has attracted much attention in the recent years due to the explosion in online data sources and the whole aspect of globalization of society and business.

To integrate a set of different local data sources means to provide a common database schema, often called a global or mediator schema, and to describe a relationship between the different local schemata and the global one. Two common paradigms for defining such relationships are the so-called *local-as-view* (LaV, a.k.a. *source-centric*) and *global-as-view* (GaV, a.k.a. *global-centric*); see [28] for definitions and comparison.

Integrity constraints of a database are overall conditions that must be met by any instance of the database in order for it to provide a meaningful semantics, and maintenance of integrity constraints is a standard issue in traditional databases.

In combined databases, on the other hand, integrity constraints have been used mainly for query reformulation and optimization, but the problem of checking and maintaining integrity constraints in this context seems to be largely ignored (a few exceptions are mentioned in section 2). This is problematic as even though each local database may satisfy its specific integrity constraints, the combined database may not have a good semantics and the answers to queries cannot be trusted.

Consider, as an example, two databases of marriages for two different countries. Both may satisfy a non-bigamist integrity constraint, but combining the two may violate this. We need methods to identify such violations and, ideally, provide means for restoring consistency, which may be done in different ways.

As in a traditional database of nontrivial size, it is not practically feasible to check integrity constraints for the entire database in one operation: an incremental approach is needed so that only a small amount of work is required for each update. For the combined case, it is even more urgent to optimize integrity checking and distribute it over time: Transition delays over network links need to be considered, and an update in this context may mean the addition of a new data source.

Theoretically sound methods, called *simplification* of integrity constraints, have been developed for relational and deductive databases, although the common practice is still based on *ad hoc* techniques. Typically, database experts need to design and hand-code either complicated tests in the program producing the update requests or triggers within the database management system that react upon certain update actions (and in combined databases, we expect it to be the same, if integrity is considered at all).

It seems obvious that a generalization of simplification techniques will be of great advantage for combined databases, and we can present some first results in adapting a newly developed and highly flexible simplification framework [13]. The main idea in simplification is to keep the database invariantly consistent, so that this knowledge can be utilized in the investigation whether the database will be consistent following a suggested update. For example, for a non-bigamist constraint, it means that it is sufficient to check the condition for new husbands and wives against the database when a marriage is proposed, as all other combinations of tuples have been checked earlier. Some simplification methods need to perform the update before the simplified constraint can be checked so that a roll-back operation may become necessary, whereas others such as [13] can do with the current state before the update, which seems to be a better approach. However, the method of [13] can also be adapted to perform post-update checks, which seems of relevance in a combined database where a local update is registered as a message that it has been performed.

With our approach we can check the integrity both when several sources are integrated and when an updated source notifies the mediator with a message about the update. In order to have a simplification, we need to have some knowledge about the current state. When integrating a new source, we may trust a statement that it satisfies its local constraints; when a message about a local update is received, we may trust two things: The combined database was consistent before the update and the update has been verified locally, i.e., the local constraints are maintained. As a natural result, we obtain that only the possible interference between the update and the other sources needs to be checked.

If global inconsistencies are found, we also indicate how integrity can be restored by maintaining at the global level a small database of virtual local changes, when it is not possible to modify the local sources directly.

The paper is organized as follows. In section 2 we review existing literature in the field. The simplification framework of [13] is introduced in section 3 and its uses for data integration are explained in section 4. Examples of the GaV and LaV approaches are given in section 5 and 6 respectively, while the problem of dealing with updates at the source level is addressed in section 7. Concluding remarks and future directions are provided in section 8.

2 Related work

The contribution of integrity constraints to data integration is usually confined to query reformulation and query optimization problems. In a GaV approach the global schema is expressed in terms of views over the sources, whereas in LaV the sources are formulated as views over the global schema. The former approach is usually considered simpler for query answering, as this typically amounts to unfolding a query with respect to the view definitions; on the other hand it is less flexible if new sources need to be added to (or removed from) the system. The latter has more involved query answer mechanisms, but enjoys good scalability, as changes at the sources do not require any modification in the global schema.

In [17] the problem of answering queries using views (*query folding*) under LaV is addressed with a technique based on resolution, and several cases, including integrity constraints, negation and recursion, are dealt with.

A foundational description of the problem of consistent query answering, without considering data integration issues, is given in [8], where analytic tableaux are used to characterize the repairs of databases that do not comply with given integrity constraints.

A short survey on the role of integrity constraints in data integration is given in [9]. They are regarded as means to extract more information from incomplete sources as well as components that raise the issue of dealing with possibly inconsistent global databases. Several GaV typologies are studied that include the treatment of key and foreign key constraints and both sound and exact mappings. In [10] the same authors develop these ideas to show that, in the presence of integrity constraints, query answering in GaV becomes as difficult as in LaV, as the problem of incomplete information implicitly arises. Further discussion on the expressive power of the two approaches, in terms of query-preserving transformations, is given in [11].

In [28], Ullman relates the problem of constructing answers to queries using views to query containment algorithms and compares two implemented systems in these terms.

Levy [20] applies techniques from artificial intelligence to the problem of data integration and shows examples of both GaV and LaV query reformulation with integrity constraints, including particular data access patterns.

Li [21] considers the use of integrity constraints in LaV query processing and optimization and distinguishes between local and global constraints and, for these, between general global constraints and source-derived global constraints.

Others, e.g., [6, 7, 22], have approached the global consistency problem by introducing *disjunctive databases*, that we shall not consider in this paper. Conflicts in the data are usually resolved by means of majority principles; to this end, [26] extensively analyzes the general problem of arbitration, which is the process of settling a divergence, by considering preferences and weights. In addition to data-related inconsistencies, the authors of [22] also consider the problem of merging sources whose compatibility is affected by the presence of synonyms, homonyms or type conflicts in the schemata.

The presence of semi-structured data, i.e., information that does not match a strict predefined schema, is a common phenomenon in current applications, such as web databases, as also reflected in the emerging XML standard. This is an important issue in data integration and the database community has developed various techniques for handling this kind of information [1, 5]; however, we will not focus on this problem in the remainder of the paper.

Another direction of research concerns automatic identification of a common global schema with mappings from different source schemata. Various techniques, such as linguistic and ontological similarity between relation and attribute names and type structures, can be used; see [27] for an overview. Recent work, e.g., [29], attempts to extract indirectly expressed similarities using a kind of shallow semantic analysis. Apart from referential integrity and key constraints, this research has not paid much attention to integrity constraints, but it seems very likely that a more fine-grained comparison of integrity constraints may help to identify semantic similarities.

Paraconsistent logics can be used to model the possible inconsistencies coming from database integration and update. In [3] it is shown how one such logic, called LFI1, has the capability of storing and managing inconsistent information.

What we want to do instead is to *check* the integrity of a database that consists of several data sources. In order to deal with this in an optimal way, in the next section we introduce new tools for the simplification of integrity constraints that can be used to manage a number of different configurations of database schemata and integrity constraints at the sources and at the mediator.

The principle of simplification of integrity constraints dates back to at least [24] and has been elaborated by many other authors. We develop this paper upon the framework described in [13] and we refer to that for further discussion, relevant proofs and references.

3 A framework for simplification of integrity constraints

We review here, with small variations, the simplification framework presented in [13] and assume a function-free first-order language equipped with negation and built-ins for equality ($\doteq$) and inequality ($\neq$), where *terms* ($t, s, \ldots$), *variables* ($x, y, \ldots$), *constants* ($a, b, \ldots$), *predicates* ($p, q, \ldots$), *atoms*, *literals* and *formulas* in general are defined as usual. The set of all ground (i.e., variable-free) atoms that can be formed from the predicate symbols and the terms in the language is referred to as the *Herbrand base*.

Clauses are written in the form *Head* $\leftarrow$ *Body* where the head, if present, is an atom and the body a (perhaps empty) conjunction of literals. A *denial* is a headless clause and a *fact* is a bodiless clause; all other clauses are called *rules*. We identify a set of formulas with their conjunction and thus *true* with $\emptyset$ and ϕ with the set $\{\phi\}$. Logical equivalence between formulas is denoted by $\equiv$, entailment by $\models$. The notation $\vec{t}$ indicates a sequence of terms $t_1, \ldots, t_n$ and the expressions $p(\vec{t})$, $\vec{s} \doteq \vec{t}$ and $\vec{s} \neq \vec{t}$ are defined accordingly. We further assume that all clauses are *range restricted* (see, e.g., [13]). For simplicity, we do not allow recursion, but our method is relevant for all database environments in which range restricted queries produce a finite set of ground tuples.

We distinguish three components of a database: the *extensional database* (the facts), the *intensional database* (the rules) and the *constraint theory* (the integrity constraints) [16]. With no loss of generality [12], we shall assume throughout the rest of the paper that the constraint theory contains only extensional predicates. By *database state* we refer to the union of the extensional and the intensional parts only.

Definition 1 (Database). *A database is a triple $\langle S, \Gamma, D\rangle$, where Γ is a constraint theory, D a database state and S a set of signatures for the predicates of Γ and D, called a database signature.*

As semantics of a database state D, with default negation for negative literals, we take its *standard model*, as D is here recursion-free and thus *stratified*. The truth value of a closed formula F, relative to D, is defined as its valuation in the standard model and denoted $D(F)$. (See, e.g., [25] for exact definitions for these and other common logical notions.)

Definition 2 (Consistency). *A database state D is* consistent *with a constraint theory Γ iff $D(\Gamma) = true$.*

The method can handle general forms of update, including additions, deletions and changes. Furthermore, [13] allows also for so-called *parameters*, i.e., placeholders for constants that permit to generalize updates into update patterns, which can be evaluated before knowing the actual values of the update itself. For example, the notation $\{p(\mathbf{a}), \neg q(\mathbf{a})\}$, where $\mathbf{a}$ is a parameter, refers to the class of updates that add a tuple to the unary relation p and remove the same tuple from the unary relation q. For simplicity, in this paper we restrict our attention to parameter-free additions, but the results we present also hold in the presence of parameters as well as for deletions and changes.

Definition 3 (Update). *An* update *is a non-empty set of ground facts.*

3.1 Semantic notions

We introduce now a few concepts that allow us to characterize the notion of simplification from a semantic point of view. We refer to [13] for further discussion.

The notion of weakest precondition is a semantic correctness criterion for a test to be run prior to the execution of the update, i.e., a test that can be checked in the present state but indicating properties of the prospective new state.

Definition 4 (Weakest precondition, strongest postcondition). *Let Γ and Σ be constraint theories and U an update. Σ is a* weakest precondition *of Γ with respect to U and Γ is a* strongest postcondition *of Σ with respect to U whenever $D(\Sigma) \equiv (D \cup U)(\Gamma)$ for any database state D.*

Weakest preconditions [14, 18] resemble the standard axiom for defining assignment statements in a programming language, whose side effects are analogous to a database update. The concept of strongest postcondition characterizes how questions concerning the previous state may be answered in the updated state.

The essence of simplification is the optimization of a weakest precondition based on the invariant that the constraint theory holds in the present state.

Definition 5 (Conditional weakest precondition). *Let Γ, Δ be constraint theories and U an update. A constraint theory Σ is a* Δ-conditional weakest precondition *(Δ-CWP) of Γ with respect to U whenever $D(\Sigma) \equiv (D \cup U)(\Gamma)$ for any database state D consistent with Δ.*

Typically, Δ will include Γ, but it may also contain other knowledge, such as further properties of the database that are trusted.

The notion of CWP alone is not sufficient to fully characterize the principle of simplification: an optimality criterion is needed, which serves as an abstraction over actual computation times without introducing assumptions about any particular evaluation mechanism or referring to any specific database state. According to definition 5, semantically different Δ-CWPs may exist, as their truth values are only fixed in the states that are consistent with Δ. Among these, we characterize as optimal a constraint theory that depends on as small a part of the database as possible (see definition 8). The following example demonstrates this idea and shows that a semantically weakest Δ-CWP, i.e., one that holds in as many states as possible, does not, in general, enjoy this property.

Example 1. Consider the constraint theory $\Gamma = \Delta = \{\leftarrow p(a) \wedge q(a),\ \leftarrow r(a)\}$ and the update $U = \{p(a)\}$. The strongest, optimal and weakest Δ-CWPs of Γ with respect to U are shown in the following table.

Strongest	Optimal	Weakest
$\{\leftarrow q(a),\ \leftarrow r(a)\}$	$\{\leftarrow q(a)\}$	$\{\leftarrow q(a) \wedge \neg p(a) \wedge \neg r(a)\}$

We base our definition of optimality on the notion of cover, i.e., a portion of the Herbrand base that does not affect the semantics of a constraint theory: the larger the cover, the better the constraint theory.

Definition 6 (Cover). *Two database states D_1, D_2 are said to* agree *on a subset $\mathcal{S}$ of the Herbrand base $\mathcal{B}$ whenever $D_1(A) = D_2(A)$ for every ground atom $A \in \mathcal{S}$. A set $\mathcal{C} \subseteq \mathcal{B}$ is a* cover *for a constraint theory Γ whenever*

$$D(\Gamma) = D'(\Gamma)$$

for any two database states D, D' that agree on $\mathcal{B} \setminus \mathcal{C}$. If, furthermore, Γ admits no other cover $\mathcal{C}' \supset \mathcal{C}$, $\mathcal{C}$ is a maximal cover *for Γ.*

Definition 7 (Conditional equivalence). *Let Δ, Γ_1, Γ_2 be constraint theories, then Γ_1 and Γ_2 are conditionally equivalent with respect to Δ, denoted $\Gamma_1 \overset{\Delta}{\equiv} \Gamma_2$, whenever $D(\Gamma_1) \equiv D(\Gamma_2)$ for any database state D consistent with Δ.*

Definition 8 (Optimality). *Given two constraint theories Δ and Σ, Σ is* Δ-optimal *if it admits a maximal cover $\mathcal{C}$ such that there exists no other constraint theory $\Sigma' \overset{\Delta}{\equiv} \Sigma$ with maximal cover $\mathcal{C}' \supset \mathcal{C}$. A Δ-optimal Δ-CWP of a constraint theory Γ with respect to an update U is a* Δ-optimal conditional weakest precondition *(Δ-OCWP) of Γ with respect to U.*

Obviously the integrity constraint $\{\leftarrow q(a)\}$ indicated as optimal in example 1 satisfies this definition, as it admits the maximal cover $\mathcal{B} \setminus \{q(a)\}$. The only larger cover is $\mathcal{B}$ and any constraint theory with cover $\mathcal{B}$ would not be a Δ-CWP of Γ with respect to U in the example.

Ideally, the test for possible inconsistency introduced by a given update should be performed prior to the update, but in the setting of data integration this might not always be feasible. A consistency test that can be made after the update is called a weakest post-precondition. Similarly to Δ-CWPs, we refer to the following conditional notion.

Definition 9 (Conditional weakest post-precondition). *Let Γ, Δ be constraint theories and U an update. A constraint theory Σ is a* Δ-conditional weakest post-precondition *(Δ-CWPP) of Γ with respect to U whenever $(D \cup U)(\Sigma) \equiv (D \cup U)(\Gamma)$ for any database state D consistent with Δ.*

The notion of optimal Δ-CWPP can be defined analogously to definition 8 and is indicated as Δ-OCWPP.

3.2 Transformations on integrity constraints

In the following, we define the transformations that are used to compose a simplification procedure.

Definition 10. *Let Γ be a constraint theory and U an update:*

$$\begin{aligned} U = \{\ & p_1(\vec{a}_{1,1}), p_1(\vec{a}_{1,2}), \ldots, p_1(\vec{a}_{1,n_1}), \\ & \ldots \\ & p_m(\vec{a}_{m,1}), p_m(\vec{a}_{m,2}), \ldots, p_m(\vec{a}_{m,n_m}))\}, \end{aligned}$$

where the p_i's are distinct predicates and the $\vec{a}_{i,j}$'s are sequences of constants. The notation $\mathsf{After}^U(\Gamma)$ refers to a copy of Γ in which all atoms of the form $p_i(\vec{t})$ have been simultaneously replaced by

$$p_i(\vec{t}) \vee \vec{t} \doteq \vec{a}_{i,1} \vee \cdots \vee \vec{t} \doteq \vec{a}_{i,n_i}.$$

The notation $\mathsf{Before}^U(\Gamma)$ is as $\mathsf{After}^U(\Gamma)$ but where the replacement is

$$p_i(\vec{t}) \wedge \vec{t} \neq \vec{a}_{i,1} \wedge \cdots \wedge \vec{t} \neq \vec{a}_{i,n_i}.$$

It follows immediately from the definition that After and Before distribute over $\cup$. We furthermore assume that the result of these transformations is always given as a set of denials, obtained by applications of De Morgan's laws, and in a "normalized" form, i.e., without redundant constraints and sub-formulas (such as, e.g., $a \doteq a$). We refer to [13] for a proposed implementation. The semantic correctness of After and Before is expressed by the following property.

Theorem 1. *For any update U and constraint theory Γ, $\mathsf{After}^U(\Gamma)$ is a weakest precondition and $\mathsf{Before}^U(\Gamma)$ is a strongest postcondition of Γ with respect to U.*

An essential step in the achievement of simpler integrity constraints is to employ the fact that they hold in the current database state, and remove those parts of the condition about the possible updated state that are implied by this. For this purpose, we introduce a transformation Optimize that produces a constraint theory which is optimal with respect to a given set of hypotheses.

Definition 11 (Optimize). *Given two constraint theories Δ, Γ, $\mathsf{Optimize}_\Delta(\Gamma)$ refers to a Δ-optimal theory Σ such that $\Sigma \overset{\Delta}{\equiv} \Gamma$.*

It should be observed that Optimize is defined in a purely mathematical way that does not indicate how to construct such a constraint theory. In the following, we shall refer to the implementation given in [13], based on the purely syntactic notion of subsumption (see, e.g., [16]): Optimize removes from Γ all denials that are subsumed by a denial in Δ. Although we have no proof for it yet, we believe that this implementation produces the desired results, at least under reasonable conditions for Γ. From definition 11 we have immediately the following.

Proposition 1. *Let Σ be a weakest precondition of constraint theory Γ with respect to an update U. Then $\mathsf{Optimize}_\Delta(\Sigma)$, Δ a constraint theory, is a Δ-OCWP of Γ with respect to U.*

The operators introduced so far can be combined to define a procedure for simplification of integrity constraints, where the updates always take place from a consistent state.

Definition 12. *For a constraint theory Γ and an update U, we define*

$$\mathsf{Simp}^U(\Gamma) = \mathsf{Optimize}_\Gamma(\mathsf{After}^U(\Gamma)).$$

As a consequence of the previous results, Simp enjoys the following property.

Proposition 2. *Let Γ be a constraint theory and U an update. Then $\mathsf{Simp}^U(\Gamma)$ is a Γ-OCWP of Γ with respect to U.*

No other work we are aware of has based simplification on the notion of optimal conditional weakest precondition.

Example 2. Consider a database containing information about marriages, which receives updates of the form $U = \{m(a, b)\}$ (husband a is married to wife b) and has the following integrity constraint (no husband has more than a wife):

$$\Gamma = \leftarrow m(x, y) \wedge m(x, z) \wedge y \neq z.$$

The simplification, showing intermediate steps in After, is calculated as follows.

$$\begin{aligned}\mathsf{After}^U(\Gamma) \equiv \{ &\leftarrow m(x,y) \wedge m(x,z) \wedge y \neq z,\\ &\leftarrow m(x,y) \wedge x \doteq a \wedge z \doteq b \wedge y \neq z,\\ &\leftarrow x \doteq a \wedge y \doteq b \wedge m(x,z) \wedge y \neq z,\\ &\leftarrow x \doteq a \wedge y \doteq b \wedge x \doteq a \wedge z \doteq b \wedge y \neq z \,\} \equiv\\ \equiv \mathsf{After}^U(\Gamma) = \{ &\leftarrow m(x,y) \wedge m(x,z) \wedge y \neq z,\\ &\leftarrow m(a,y) \wedge y \neq b\}\end{aligned}$$

$$\mathsf{Simp}^U(\Gamma) = \{\leftarrow m(a,y) \wedge y \neq b\}$$

There may be specific applications where consistency needs to be checked directly on the updated database. In these cases, Before comes in handy.

Definition 13. *For a constraint theory Γ and an update U, we define*

$$\mathsf{PostSimp}^U(\Gamma) = \mathsf{Before}^U(\mathsf{Simp}^U(\Gamma)).$$

Proposition 3. *Let D be a database state consistent with a constraint theory Γ and U an update. Then $(D \cup U)(\Gamma) \equiv (D \cup U)(\mathsf{PostSimp}^U(\Gamma))$.*

Proposition 4. *Let Γ be a constraint theory and U an update. Then $\mathsf{PostSimp}^U(\Gamma)$ is a Γ-OCWPP of Γ with respect to U.*

4 Integrity constraints in combined databases

4.1 Extending the framework for data integration

In section 3 we described a framework that applies to a single updatable database. In the context of data integration, we have in general several databases (the sources and the mediator) and other operations than database updates need to be considered, such as database combination. We shall therefore generalize the notation in order to describe the relevant cases for data integration. In order to provide a unified view of the data residing at different sources, we need to indicate how the global predicates are expressed in terms of the source predicates. We shall therefore introduce the notion of mapping, which we assume to be *sound*, i.e., the information produced by the views over the sources contains only, but not necessarily all the data associated to the global predicates. *Complete* mappings and *exact* mappings are defined in a similar way (see, e.g., [10]), but we do not consider them in this paper.

Definition 14 (Mapping). *A mapping $M : (\mathcal{D}_1, ..., \mathcal{D}_n) \rightarrow \mathcal{D}$, where $\mathcal{D}$, $\mathcal{D}_1$, ..., $\mathcal{D}_n$ are databases with disjoint signatures, is a set of range restricted rules in which the predicates in the heads are in $\mathcal{D}$ and their terms are distinct variables, and the predicates in the bodies are in one of the $\mathcal{D}_1, ..., \mathcal{D}_n$ or are built-ins.*

Notice that it is always possible to rewrite a set of range restricted rules as a mapping: for a rule whose head contains some constants or non distinct variables, they can be replaced by new variables, provided that equalities taking track of the replacements are added in the body.

We can now extend the After operator to handle data integrations described by a mapping from the sources to the mediator.

Definition 15. *Let $\mathcal{D}, \mathcal{D}_1, ..., \mathcal{D}_n$ be some databases with disjoint signatures, Γ a constraint theory concerning the predicates in $\mathcal{D}$ and M a mapping of the form:*

$$M = \{\ p_1(\vec{x}_1) \leftarrow B_{1,1}, \quad \ldots,\ p_1(\vec{x}_1) \leftarrow B_{1,n_1},$$
$$\vdots$$
$$p_m(\vec{x}_m) \leftarrow B_{m,1},\ \ldots,\ p_m(\vec{x}_m) \leftarrow B_{m,n_m}\},$$

where the p_i's are all the (distinct) predicates in $\mathcal{D}$, the $\vec{x}_i$'s are sequences of variables and the $B_{i,j}$'s are conjunctions of literals whose predicates are in one of the $\mathcal{D}_1, ..., \mathcal{D}_n$ or are built-ins. The notation $\mathsf{After}^M(\Gamma)$, where $M : (\mathcal{D}_1, ..., \mathcal{D}_n) \to \mathcal{D}$ is a mapping, refers to a copy of Γ in which all atoms of the form $p_i(\vec{t})$ have been simultaneously replaced by

$$B_{i,1}\theta_i\rho_{i,1} \vee ... \vee B_{i,n_i}\theta_i\rho_{i,n_i},$$

where θ_i is a substitution that replaces the variables of $\vec{x}_i$ with the terms of $\vec{t}$ and $\rho_{i,j}$ a renaming giving fresh new names to the variables of $B_{i,j}$ not in $\vec{x}_i$ to avoid name clashes.

We use the term *operation* to refer to either an update or the application of a mapping. The After operator, as stated by definitions 10 and 15, behaves as follows: for an update, it translates a theory concerning the updated state to one referring to the state before the update; similarly, for a database combination, it moves from the state after the integration to the non-integrated state, i.e., from a theory concerning the mediator to one concerning the sources. It is therefore meaningful to extend the notion of weakest precondition accordingly.

Definition 16. *Let Γ and Σ be constraint theories and $M : (\mathcal{D}_1, ..., \mathcal{D}_n) \to \mathcal{D}$ a mapping where $\mathcal{D}, \mathcal{D}_1, ..., \mathcal{D}_n$ are databases. Σ is a* weakest precondition *of Γ with respect to M whenever $(D_1 \cup ... \cup D_n)(\Sigma) \equiv D(\Gamma)$ for any database states $D, D_1, ..., D_n$ in $\mathcal{D}, \mathcal{D}_1, ..., \mathcal{D}_n$.*

The notions of CWP and OCWP can be extended in a similar way. We immediately have the extension of theorem 1 and proposition 1, where the word "operation" can be used instead of "update".

In definition 12 we implicitly assumed that the argument of Simp was both the theory to be simplified and the condition known to hold prior to the operation; it is now useful to extend the notation as follows.

Definition 17. *For two constraint theories Γ, Δ and an operation O, we define*

$$\mathsf{Simp}^O_\Delta(\Gamma) = \mathsf{Optimize}_\Delta(\mathsf{After}^O(\Gamma)).$$

Proposition 2 can immediately be generalized as follows.

Proposition 5. *Let Γ, Δ be constraint theories and O an operation. Then $\mathsf{Simp}^{O}_{\Delta}(\Gamma)$ is a Δ-OCWP of Γ with respect to O.*

The notation $\mathsf{Simp}^{U}(\Gamma)$ of definition 12, U an update, can therefore be considered as a shorthand for $\mathsf{Simp}^{U}_{\Gamma}(\Gamma)$. Examples of the application of these extended versions of the operators are shown in the next sections.

4.2 Consistent views on inconsistent states

When the information contained in the source databases is conflicting with the global requirements and the sources cannot be modified, it is possible to repair inconsistencies by maintaining a virtual database of exceptions $\mathcal{D}_E$ at the global level. Suppose sources $\mathcal{D}_1, ..., \mathcal{D}_n$ are given and the global database $\mathcal{D}$ is defined via a mapping $M : (\mathcal{D}_1, ..., \mathcal{D}_n) \rightarrow \mathcal{D}$. Let $\mathcal{D}_E$'s signature contain for each predicate p in $\mathcal{D}$ two predicates p_+ and p_- of the same arity, the former representing the tuples that should be in p but are not, the latter those that should not be in p but are. The consistent global database $\mathcal{D}_C$ can then be thought of as a combination of $\mathcal{D}_E$ and $\mathcal{D}$ that contains a predicate p_C for every predicate p in $\mathcal{D}$, as defined in the mapping $M_E : (\mathcal{D}, \mathcal{D}_E) \rightarrow \mathcal{D}_C$ by the following entries:

$$p_C(\vec{x}) \leftarrow p(\vec{x}) \wedge \neg p_-(\vec{x}),$$
$$p_C(\vec{x}) \leftarrow p_+(\vec{x}).$$

In order to identify suitable p_+ and p_- to re-establish consistency, an abductive algorithm can be integrated with the procedure for evaluating the simplified integrity constraints. A full treatment of this topic is outside the scope of this paper; see, e.g., [15, 19] for an overview of abductive methods. Applications of abduction to database repair that fit our model are described in [2, 4].

5 Integrity constraints under global-as-view

In a global-centric approach it is required that the global schema is expressed in terms of the sources. The global predicates must be associated with views over the sources: this is exactly what the notion of mapping makes precise, as stated in definition 14. A mapping containing entries for all global predicates is called a GaV-mapping.

We start our analysis by considering a borderline case of GaV-mapping consisting of the combination of two[1] databases having the same signature and constraint theory. Let S_1, S_2, S be three disjoint database signatures that are identical up to consistent renaming of predicates and Γ_1, Γ_2 and Γ the constraint theories (also identical up to renaming) defined at the sources and the mediator, respectively. The global database state consists of the union of the local ones

[1] The case with more than two sources is similar.

and the mapping M used for the combination is defined as a set containing, for every predicate p in S, the entries:

$$p(\vec{x}) \leftarrow p_1(\vec{x}),$$
$$p(\vec{x}) \leftarrow p_2(\vec{x}),$$

where $\vec{x}$ is a sequence of variables and p_1, p_2 are the predicates corresponding to p in S_1, S_2, respectively. A simplified test for checking that the combined database is consistent with Γ is then given by $\mathsf{Simp}^M_{\Gamma_1 \wedge \Gamma_2}(\Gamma)$.

Example 3. Let us refer to example 2 and consider two sources $\mathcal{D}_1 = \langle S_1, \Gamma_1, D_1 \rangle$, $\mathcal{D}_2 = \langle S_2, \Gamma_2, D_2 \rangle$ and a mediator $\mathcal{D} = \langle S, \Gamma, D \rangle$ with signatures[2] $S_1 = \{m_1/2\}$, $S_2 = \{m_2/2\}$, $S = \{m/2\}$ and the following integrity constraints:

$$\begin{aligned} \Gamma &= \leftarrow m(x,y) \wedge m(x,z) \wedge y \neq z, \\ \Gamma_1 &= \leftarrow m_1(x,y) \wedge m_1(x,z) \wedge y \neq z, \\ \Gamma_2 &= \leftarrow m_2(x,y) \wedge m_2(x,z) \wedge y \neq z. \end{aligned}$$

D_1, D_2 are consistent database states and D is their combination, as expressed by the mapping: $M = \{m(x,y) \leftarrow m_1(x,y),\ m(x,y) \leftarrow m_2(x,y)\}$. We have:

$$\begin{aligned} \mathsf{After}^M(\Gamma) \equiv \{ \leftarrow\ & (m_1(x,y) \vee m_2(x,y)) \wedge \\ & (m_1(x,z) \vee m_2(x,z)) \wedge y \neq z \} \equiv \\ \equiv \mathsf{After}^M(\Gamma) = \{ \leftarrow\ & m_1(x,y) \wedge m_1(x,z) \wedge y \neq z, \\ \leftarrow\ & m_1(x,y) \wedge m_2(x,z) \wedge y \neq z, \\ \leftarrow\ & m_2(x,y) \wedge m_2(x,z) \wedge y \neq z\} \end{aligned}$$

The only check that is needed is, as expected, a cross-check between the two databases, as the other denials are removed by Optimize:

$$\mathsf{Simp}^M_{\Gamma_1 \wedge \Gamma_2}(\Gamma) = \{ \leftarrow m_1(x,y) \wedge m_2(x,z) \wedge y \neq z \}.$$

We can get even better results when extra knowledge concerning the combination of the sources is available. This simply amounts to adding the extra knowledge to the conditions in the subscript of Simp.

Example 4. Consider example 3, where we have now the knowledge $\Gamma_{1,2}$ that the data concerning the husbands in the two databases are disjoint:

$$\Gamma_{1,2} = \leftarrow m_1(x,y) \wedge m_2(x,z).$$

A much stronger simplification is obtained now, as

$$\mathsf{Simp}^M_{\Gamma_1 \wedge \Gamma_2 \wedge \Gamma_{1,2}}(\Gamma) = \mathit{true}.$$

The cross-check that was found in example 3 is subsumed by $\Gamma_{1,2}$ and thus discarded, so no check is needed, as the combined state will anyhow be consistent.

[2] As usual, predicate signatures are indicated as $name/arity$.

When the mapping is arbitrary, the method can be applied in a similar way.

Example 5. We consider a data integration problem inspired by Levy [20] but here extended with global and local integrity constraints. Suppose we have two sources containing information about movies. We use the variables i, t, y and r for movie identifiers, titles, years and reviews respectively. The first source contains movies $m(i,t,y)$, where i is key, whereas the second contains reviews $r(i,r)$. Furthermore, we know that the identifiers in r are a subset of the identifiers in m. The mediator assembles this information in a relation $f(i,t,r)$ (film), as defined by the following GaV-mapping:

$$M = \{f(i,t,r) \leftarrow m(i,t,y) \wedge r(i,r)\}.$$

The following conditions are therefore known to hold on the ensemble of sources:

$$\begin{aligned} \Gamma_1 = \{ &\leftarrow m(i,t_1,y_1) \wedge m(i,t_2,y_2) \wedge t_1 \neq t_2, \\ &\leftarrow m(i,t_1,y_1) \wedge m(i,t_2,y_2) \wedge y_1 \neq y_2\}, \\ \Gamma_{1,2} = \quad &\leftarrow r(i,r) \wedge \neg m(i,t,y). \end{aligned}$$

Let Γ express the fact that i is a primary key for f:

$$\begin{aligned} \Gamma = \{ &\leftarrow f(i,t_1,r_1) \wedge f(i,t_2,r_2) \wedge t_1 \neq t_2, \\ &\leftarrow f(i,t_1,r_1) \wedge f(i,t_2,r_2) \wedge r_1 \neq r_2\}. \end{aligned}$$

In order to check whether Γ holds globally, we proceed as follows.

$$\begin{aligned} \mathsf{After}^M(\Gamma) = \{ &\leftarrow m(i,t_1,y_1) \wedge r(i,r_1) \wedge m(i,t_2,y_2) \wedge r(i,r_2) \wedge t_1 \neq t_2, \\ &\leftarrow m(i,t_1,y_1) \wedge r(i,r_1) \wedge m(i,t_2,y_2) \wedge r(i,r_2) \wedge r_1 \neq r_2\}. \end{aligned}$$

The first constraint is obviously subsumed by the first constraint in Γ_1 and the second one can be simplified with $\Gamma_{1,2}$, which gives the following:

$$\mathsf{Simp}^M_{\Gamma_1 \wedge \Gamma_{1,2}}(\Gamma) = \{\leftarrow r(i,r_1) \wedge r(i,r_2) \wedge r_1 \neq r_2\},$$

which evidently corresponds to the fact that i must be a key for r as well.

6 Integrity constraints under local-as-view

A LaV-mapping is usually understood as a set of views of source predicates over global predicates. From now on with the word LaV-view we will refer to a formula of the form $A \rightarrow B$, where A is an atom (the antecedent), B a conjunction of atoms (the consequents) and universal quantification at the outmost level is understood for all variables. A LaV-view $A \rightarrow B$ is *safe* whenever all the variables in B appear in A as well. A set of safe LaV-views is called a safe LaV-mapping. Given a safe[3] LaV-mapping, and assuming it is sound as discussed in section 4, we can always rewrite it as an equivalent mapping. This is shown in

[3] Safeness is used to avoid skolemization.

the following examples and can be done by using the fact that, with safeness, whenever $A \rightarrow A_1 \wedge ... \wedge A_n$, then $A_1 \leftarrow A$ and ... and $A_n \leftarrow A$, where $A, A_1, ..., A_n$ are atoms, and perhaps by adding some equalities in the bodies in order to have only distinct variables in the heads. Note that in general a LaV-mapping is not necessarily a mapping in the sense of definition 14.

Example 6. Reconsider the scenario discussed in example 3. The global database combines now two sources where in the first one the husbands are Italian, and in the second one Danish, which is expressed by the LaV-mapping $L = \{m_1(x,y) \rightarrow m(x,y) \wedge n(x,it),\ m_2(x,y) \rightarrow m(x,y) \wedge n(x,dk)\}$, where m and n are global predicates. An equivalent mapping is as follows:

$$\begin{aligned} M_L = \{\ & m(x,y) \leftarrow m_1(x,y), \\ & n(x,z) \leftarrow m_1(x,y) \wedge z \doteq it, \\ & m(x,y) \leftarrow m_2(x,y), \\ & n(x,z) \leftarrow m_2(x,y) \wedge z \doteq dk\}. \end{aligned}$$

We note that, given the local no-bigamist assumptions Γ_1 and Γ_2, the simplification of uniqueness of nationality $\mathsf{Simp}^{M_L}_{\Gamma_1 \wedge \Gamma_2}(\leftarrow n(x,y) \wedge n(x,z) \wedge y \neq z) = \leftarrow m_1(x,y) \wedge m_2(x,z)$ corresponds to the disjointness of m_1 and m_2.

Example 7. This example is also inspired by Levy ([20]) and extended with global and local integrity constraints. The global database integrates three different sources that provide information about movies. We use the variables t, y, d and g for movie titles, years, directors, and genres respectively. The global predicates are $m(t,y,d,g)$, representing a given movie, and $d(d)$, $i(d)$, $a(d)$, ..., representing nationalities of directors, here Danish, Italian, American, etc. The following integrity constraints are assumed: a key constraint on m (t,y is key), a domain constraint on film genre, and uniqueness of nationality. Underlines are used as anonymous variables *à la* Prolog for ease of notation.

$$\begin{aligned} \Gamma = \{\ & \leftarrow m(t,y,d_1,\_) \wedge m(t,y,d_2,\_) \wedge d_1 \neq d_2, \\ & \leftarrow m(t,y,\_,g_1) \wedge m(t,y,\_,g_2) \wedge g_1 \neq g_2, \\ & \leftarrow m(\_,\_,\_,g) \wedge g \neq \mathit{comedy} \wedge g \neq \mathit{drama} \wedge \cdots, \\ & \leftarrow d(d) \wedge i(d), \\ & \leftarrow d(d) \wedge a(d), \\ & \cdots\}. \end{aligned}$$

There are three source databases. The first one contains American comedies given as $m_1(t,y,d)$ with t,y as key and a LaV-mapping as follows.

$$\begin{aligned} \Gamma_1 &= \leftarrow m_1(t,y,d_1) \wedge m_1(t,y,d_2) \wedge d_1 \neq d_2 \\ L_1 &= \{m_1(t,y,d) \rightarrow m(t,y,d,\mathit{comedy}) \wedge a(d)\}. \end{aligned}$$

The second source contains Danish movies only, with a key constraint Γ_2 on t,y as usual and the following LaV-view.

$$L_2 = \{m_2(t,y,d,g) \rightarrow m(t,y,d,g) \wedge d(d)\}.$$

The third source is a general list of movies with predicates similar to the global ones (m_3, d_3, i_3, a_3 etc.) and with similar integrity constraints Γ_3. The LaV-views are specified as follows.

$$\begin{aligned} L_3 = \{\ & m_3(t,y,d,g) \rightarrow m(t,y,d,g), \\ & d_3(d) \rightarrow d(d), \\ & \cdots\} \end{aligned}$$

As $L_1 \cup L_2 \cup L_3$ is safe we can rewrite it as the following mapping.[4]

$$\begin{aligned} M = \{\ & m(t,y,d,g) \leftarrow m_1(t,y,d) \wedge g \doteq \mathit{comedy}, \\ & m(t,y,d,g) \leftarrow m_2(t,y,d,g), \\ & m(t,y,d,g) \leftarrow m_3(t,y,d,g), \\ & a(d) \leftarrow m_1(\_,\_,d), \\ & d(d) \leftarrow m_2(\_,\_,d,\_), \\ & a(d) \leftarrow a_3(d), \\ & d(d) \leftarrow d_3(d), \\ & i(d) \leftarrow i_3(d), \\ & \cdots\} \end{aligned}$$

The simplified integrity constraints for the integration of the three databases, given as $\Sigma = \mathsf{Simp}^{M}_{\Gamma_1 \wedge \Gamma_2 \wedge \Gamma_3}(\Gamma)$, are, as expected, simplified rules covering possible conflicts in cross combinations only, as local consistency is assumed.

$$\begin{aligned} \Sigma = \{\ & \leftarrow m_1(\_,\_,d) \wedge m_2(\_,\_,d,\_), \\ & \leftarrow m_1(\_,\_,d) \wedge \mathbf{p}_3(d), \quad (\mathbf{p} \text{ any nationality pred. different from } a) \\ & \leftarrow m_2(\_,\_,d,\_) \wedge \mathbf{p}_3(d), \quad (\mathbf{p} \text{ any nationality pred. different from } d) \\ & \leftarrow m_1(t,y,\_) \wedge m_2(t,y,\_,\_), \\ & \leftarrow m_1(t,y,d_1) \wedge m_3(t,y,d_2,\_) \wedge d_1 \neq d_2, \\ & \leftarrow m_2(t,y,d_1,\_) \wedge m_3(t,y,d_2,\_) \wedge d_1 \neq d_2, \\ & \leftarrow m_1(t,y,\_) \wedge m_3(t,y,\_,g) \wedge g \neq \mathit{comedy}, \\ & \leftarrow m_2(t,y,\_,g_1) \wedge m_3(t,y,\_,g_2) \wedge g_1 \neq g_2, \\ & \leftarrow m_2(\_,\_,\_,g) \wedge g \neq \mathit{comedy} \wedge g \neq \mathit{drama} \wedge \cdots\}. \end{aligned}$$

The example can be changed a bit assuming that the third source is an unchecked database to which enthusiastic amateurs can add arbitrary information, which is not unrealistic in case of the world wide web. In that case, the simplified constraints include also a copy of the full set of global constraints with predicate names m_3, a_3, etc.

7 Absorption of local updates

A data integration system needs to be able to adjust itself dynamically as sources are updated over time. We assume that source databases maintain their own consistency and that reports about which updates have been performed are

[4] If, say, in m_1 the genre was left unspecified, skolemization would be needed.

available at the global level; this may be supplied by the source administrator or generated by a process monitoring the sources. Consistency needs then to be checked globally. This problem, that we refer to as *absorption* of local updates, can be handled in an optimal way as described in the following proposition.

Proposition 6. *Given n sources with database states D_i, $1 \leq i \leq n$, a mediator $\langle S, \Gamma, D\rangle$ obtained with mapping M and a set of conditions Δ holding in $D_S = D_1 \cup ... \cup D_n$, let $\Sigma = \mathsf{Simp}^M_\Delta(\Gamma)$ hold in D, U be an update for D_S and D^U the state at the mediator after the update. Then $D^U(\Gamma) \equiv D_S(\mathsf{PostSimp}^U(\Sigma))$.*

Note that the condition expressed by $\mathsf{PostSimp}^U(\Sigma)$ is optimal in the sense of proposition 4.

Example 8. In example 3, the following integrity constraint was generated for the integration of two sources referred to by predicates m_1 and m_2:

$$\Sigma = \leftarrow m_1(x, y) \wedge m_2(x, z) \wedge y \neq z.$$

If the update $m_1(a, b)$ is reported, the optimal way to check the global consistency is to test $\mathsf{PostSimp}^U(\Sigma) = \leftarrow m_2(a, z) \wedge b \neq z$ on the updated state.

Example 9. Consider Σ from the LaV integrated movie database of example 7 and assume that the second source reports the addition of a new movie

$$U = \{m_2(\textit{“Dogville”}, 2003, \textit{“von Trier”}, \textit{drama})\}.$$

The following tests, calculated as $\mathsf{PostSimp}^U(\Sigma)$, remain:

$$\begin{array}{l}\{ \leftarrow m_1(\_, \_, \textit{“von Trier”}),\\ \quad \leftarrow \mathbf{p}_3(\textit{“von Trier”}), \quad (\mathbf{p} \text{ any nationality pred. different from } d)\\ \quad \leftarrow m_1(\textit{“Dogville”}, 2003, \_),\\ \quad \leftarrow m_3(\textit{“Dogville”}, 2003, d, \_) \wedge d \neq \textit{“von Trier”},\\ \quad \leftarrow m_3(\textit{“Dogville”}, 2003, \_, g) \wedge g \neq \textit{drama}\}.\end{array}$$

8 Conclusion

We revisited simplification techniques for integrity constraints and applied them to the problem of consistency checking in a data integration setting. Examples were discussed that showed that the described method lends itself well to both the LaV and GaV approaches and is useful for the problem of update absorption.

The novelty of this approach mainly consists of two aspects: firstly, the reuse of a simplification framework that was initially conceived for a single database and, secondly, the characterization of the semantic notions involved in the combined case that underlie the optimality of the method.

The treatment of cases that require skolemization when converting LaV-mappings to mappings has not been attempted in this paper. Future directions include the investigation of new strategies, if needed, that deal with such cases

as well as the extension of the method to situations where the mapping is not sound, but exact or complete.

Acknowledgements The authors wish to thank Amos Scisci for his comments on the first draft of the manuscript. This research is supported in part by the IT-University of Copenhagen.

References

1. Abiteboul, S. (1997). Querying semi-structured data. *Proc. of the Int. Conf on Database Theory (ICDT)*, pp. 1–18, Vol. 1186, Springer LNCS.
2. Arenas, M., Bertossi, L., Chomicki, J., (2000). Specifying and Querying Database Repairs using Logic Programs with Exceptions. *Proceedings of FQAS 2000*, Larsen, H., Kacprzyk, J. Zadrozny, S., Andreasen, T., Christiansen, H. (eds.), pp. 27–41, Physica-Verlag Heidelberg New York, A Springer-Verlag Company.
3. De Amo, S., Carnielli, W., Marcos, J. (2002). A Logical Framework for Integrating Inconsistent Information in Multiple Databases. *FoIKS 2002*, Proceedings, Eds. Eiter, T. and Schewepp, K., pp. 67–84, Vol. 2284, Springer LNCS.
4. Arieli, O., Denecker, M., Van Nuffelen, B., Bruynooghe, M. (2002). Repairing Inconsistent Databases: A Model-Theoretic Approach and Abductive Reasoning. *PCL 2002*, Decker, H., Villadsen, J., Waragaipp, T. pp. 51–65, Datalogiske Skrifter, Vol. 95, Roskilde University, Roskilde, Denmark.
5. Buneman, P. (1997). Semistructured data. *Proc. of the ACM SIGACT-SIGMOD-SIGART Smposium on Principles of Database Systems (PODS)*, pp-117–121, ACM Press.
6. Baral, C., Kraus, S., Minker, J., (1991). Combining multiple knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*. 3(2), pp. 208–220.
7. Baral, C., Kraus, S., Minker, J., Subrahmanian, S. (1992). Combining knowledge bases consisting of first-order theories. *Computational Intelligence*, 8, pp. 45–71.
8. Bertossi, L., Schwind, C. (2002). Analytic Tableaux and Database Repairs: Foundations.*FoIKS 2002*, Proceedings, Eds. Eiter, T. and Schewepp, K., pp. 32–48, Vol. 2284, Springer LNCS.
9. Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M. (2002). On the Role of Integrity Constraints in Data Integration, *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 25, n. 3, pp. 39–45.
10. Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M. (2002). Data Integration under Integrity Constraints. *Proceedings of CAiSE 2002*, pp. 262–279, Vol. 2348, Springer LNCS.
11. Calì, A., Calvanese, D., De Giacomo, G., Lenzerini, M. (2002). On the Expressive Power of Data Integration Systems. *Proc. of the 21st Int. Conf. on Conceptual Modeling (ER 2002)*, pp. 338–350, Vol. 2503, Springer LNCS.
12. Chakravarthy, U. S., Grant, J., Minker, J. (1987). Foundations of semantic query optimization for deductive databases. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed., Morgan Kaufmann.
13. Christiansen, H., Martinenghi, D., Simplification of database integrity constraints revisited: A transformational approach. Presented at LOPSTR, Uppsala, Sweden, 25–27 august 2003. Preliminary version available at `http://www.dat.ruc.dk/~henning/LOPSTR03.pdf`

14. Dijkstra, E.W. (1976). *A Discipline of Programming*. Prentice-Hall.
15. Flach. P., Kakas, A., (eds.), *Abductive and Inductive Reasoning: Essays on their Relation and Integration*, Kluwer Academic Publishers. pp. 195–211, 2000.
16. Godfrey, P., Grant, J., Gryz, J., Minker, J. (1988). Integrity Constraints: Semantics and Applications. *Logics for Databases and Information System*, Eds. Chomicki, J. Saake, G., Kluwer, pp. 265–306.
17. Grant, J., Minker, J. (2002). A logic-based approach to data integration. *Theory and Practice of Logic Programming (TPLP)*, vol. 2, pp. 323–368.
18. Hoare, C.A.R. (1969). An axiomatic basis for computer programming. *Communications of the ACM* 12, no. 10. pp. 576–580.
19. Kakas, A.A., Kowalski, R.A., Toni, F. (1998). The role of abduction in logic programming, *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 5, Gabbay, D.M, Hogger, C.J., Robinson, J.A., (eds.), Oxford University Press, pp. 235–324.
20. Levy, A. Y. (1999). Combining Artificial Intelligence and Databases for Data Integration. *Artificial Intelligence Today*, LNAI 1600. Eds. Wooldridge, M. J., Veloso, M., Springer-Verlag Berlin Heidelberg, pp. 249–268.
21. Li, C. (2003). Describing and Utilizing Constraints to Answer Queries in Data-Integration Systems. *IJCAI 2003 workshop on Information Integration on the Web*, On-line proceedings available at `http://www.isi.edu/info-agents/workshops/ijcai03/proceedings.htm`.
22. Lin, J., Mendelzon, A. (1998). Merging Databases Under Constraints. *International Journal of Cooperative Information Systems* 7, no. 1. pp. 55–76.
23. Martinenghi, D. (2003). A Simplification Procedure for Integrity Constraints. *World Wide Web*, `http://www.dat.ruc.dk/~dm/spic/index.html`.
24. Nicolas, J.-M. (1982). Logic for Improving Integrity Checking in Relational Data Bases., *Acta Informatica* 18, pp. 227–253.
25. Nilsson, U., Małuzyński, J. (1995). *Logic, Programming and Prolog (2nd ed.)*, John Wiley & Sons Ltd.
26. Revesz, P. (1997). On the Semantics of Arbitration. *Journal of Algebra and Computation*, 7(2), pp. 133–160.
27. Rahm, E., Bernstein, P. (2001). A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4): 334–350.
28. Ullman, J. (1997). Information integration using logical views. *International Conference on Database Theory*, pp. 19–40.
29. Xu, L., Embley, D. (2003). Discovering Direct and Indirect Matches for Schema Elements. *Eighth International Conference on Database Systems for Advanced Applications (DASFAA '03), Kyoto, Japan*, pp. 39–46, IEEE Computer Society.