Combining a Logical and a Numerical Method for Data Reconciliation

Fatiha Saïs, Nathalie Pernelle, Marie-Christine Rousset

Combining a Logical and a Numerical Method for Data Reconciliation

Fatiha Saïs[1], Nathalie Pernelle[1], and Marie-Christine Rousset[2]

[1] LRI, Paris-Sud 11 University, and INRIA Futurs,
2-4 rue J. Monod, F-91893 ORSAY, FRANCE
{Fatiha.Sais, Nathalie.Pernelle}@lri.fr

[2] LIG - Laboratoire d'Informatique de Grenoble
BP 72, 38402 St MARTIN D'HERES, FRANCE
Marie-Christine.Rousset@imag.fr

Abstract. The reference reconciliation problem consists in deciding whether different identifiers refer to the same data, i.e. correspond to the same real world entity. In this article we present a reference reconciliation approach which combines a logical method for reference reconciliation called L2R and a numerical one called N2R. This approach exploits the schema and data semantics, which is translated into a set of Horn FOL rules of reconciliation. These rules are used in L2R to infer exact decisions both of reconciliation and non-reconciliation. In the second method N2R, the semantics of the schema is translated in an informed similarity measure which is used by a numerical computation of the similarity of reference pairs. This similarity measure is expressed in a non linear equation system, which is solved by using an iterative method. The experiments of the methods made on two different domains, show good results for both recall and precision. They can be used separately or in combination. We have shown that their combination allows to improve runtime performance.

Key words: Semantic Data Integration, Ontologies, Automatic reasoning, Reference reconciliation, Equation system, Iterative resolution.

1 Introduction

The data reconciliation problem is one of the main problems encountered when different sources have to be integrated. It consists in deciding whether different data descriptions refer to the same real world entity (e.g. the same person or the same publication). For example, in a standard relational database a data description is a set of tuples referring to a given identifier. In the context of data integration, data descriptions are coming from different sources. These sources are heterogeneous, built in an autonomous way and for different business requirements. In such a context, the assumption of unique identifier does not hold: two different identifiers can refer to the same real world entity. We therefore prefer to use the term of *reference* instead of *identifier*. In the following like [1] we

will use "*the reference reconciliation problem*" to refer to the data reconciliation problem. This problem is also known as the record linkage or the record matching problem [2–4], the entity resolution problem [5, 6] or the object identification problem [7].

Schema heterogeneity is a major cause of the mismatch of the data descriptions between sources. Extensive research work has been done recently (see [8, 9] for surveys) to reconcile schemas and ontologies through mappings. In this work, we assume that the schema heterogeneity problem has been solved. We focus on the data heterogeneity problem when data conform to the same global schema.

The conformity to a same global schema does not indeed prevent variations between data descriptions. For example, two descriptions of persons with the same attributes Last Name, First Name, Address can vary on the values of those attributes while referring to the same person, for instance, if the First Name is given entirely in one tuple, while it is abbreviated in the second tuple.

Therefore, the reference reconciliation problem is a crucial issue for data integration and raises multiple difficulties. First, different conventions and vocabularies can be used to represent and describe data. For example, in one source a contact attribute can be represented by a set of phone numbers while in another source it is represented by an e-mail address. Second, information can be incomplete, i.e. the values of some attributes can be missing. Third, data descriptions can contain syntactic errors which are specially frequent when they are automatically extracted from the Web. Fourth, as data descriptions can be created and updated independently in different sources, their freshness over sources is a real issue which can lead to have apparently different descriptions representing the same real world entity.

Data cleaning which aims at detecting duplicates in databases is faced with the same problems. Most of the existing works (e.g., [10–12]) perform comparisons between strings for computing the similarity between the values of the same attribute, and then combine them for computing the similarity between tuples. In [5] the matching between data descriptions is generic but is still based on local comparisons. Some recent works [13, 1, 14, 6] follow a global approach that exploits the dependencies possibly existing between reference reconciliations. Those dependencies often result from the semantics of the domain of interest. For example, the reconciliation between two courses described by their titles and the name of the professors in charge of them can entail the reconciliation between two descriptions of persons. This requires some knowledge of the domain to be made explicit, like the fact that a professor is a person, that a course is identified by its title and has only one professor in charge of it. In [1], such knowledge is taken into account but must be encoded in the weights of the edges of the dependency graph.

In this paper, we study the problem of reference reconciliation in the case where the data are described relatively to a rich schema expressed in RDFS [15] extended by some primitives of OWL-DL [16] and SWRL [17]. OWL-DL and SWRL are used to enrich the semantics of the classes and properties declared

in RDFS. This enriched data model that we have called RDFS+ enables to express that two classes are disjoint or that some properties (or their inverse) are functional. Note that relational data and schema can be easily mapped into RDFS [18, 19] and their constraints translated in RDFS+.

For the reference reconciliation problem we propose a knowledge-based and unsupervised approach, based on two methods, a logical one called L2R and a numerical one called N2R. The logical method for reference reconciliation (L2R) is based on the translation in first order logic Horn rules of some of the schema semantics. These Horn rules enable to infer both exact reconciliations and non-reconciliations among a subset of reference pairs. Rules inferring synonymies and non synonymies between basic values are also generated from the schema constraints. Since L2R is based on logical inferences, it has a 100% precision, under the assumption that the schema and data are error-free. Such an assumption is not necessarily satisfied when the data is "dirty" or the global schema is an integrated schema resulting from an automatic reconciliation process. In order to complement the partial results of L2R, we have designed a Numerical method for Reference Reconciliation (N2R). It exploits the L2R result and allows to compute similarity scores for each pair of references. The distinguishing features of N2R compared to existing numerical methods of reference reconciliation are (i) it is unsupervised and (ii) the similarity computation takes into account some of the schema semantics : the functional dependencies of the properties are captured by aggregating the similarities of the involved references and values using the maximum function. Consequently, the mutual influences between similarity scores are expressed in a non linear equation system. In order to solve it, we use an iterative method inspired from *Jacobi* method [20] for which we have proved the convergence.

In the two methods the (non) reconciliation decisions or the similarity scores are propagated to other reference pairs. Therefore, L2R and N2R are two global methods, which can be applied separately or in combination. They are based on the most recent proposals for the Semantic Web (RDF, OWL-DL and SWRL). They can be used for reconciling data in most of the applications based on the Semantic Web technologies. The experimentations done on two different data sets (scientific publication domain and tourism domain) show good results for both precision and recall. Furthermore, the recall is significantly increased if the schema is enriched by adding constraints.

The paper is organized as follows. In Section 2, we define the data model, and in Section 3, the problem of reference reconciliation that we consider. Then, in Section 4, we describe the logical method implemented in L2R, and in Section 5, the numerical method implemented in N2R. In Section 6, we present the results that we have obtained for the experimental evaluation of L2R an N2R on two real data sets. We summarize the related work in Section 7. Finally, we conclude and sketch some future work in Section 8.

2 The RDFS$^+$ data model

We describe the data model, that we have called RDFS$^+$ because it extends RDFS with some OWL-DL primitives and SWRL rules. RDFS$^+$ can be viewed as a fragment of the relational model (restricted to unary or binary relations) enriched with typing constraints, inclusion and exclusion between relations and functional dependencies.

2.1 Schema representation and its constraints

A RDFS schema consists of a set of classes (unary relations) organized in a taxonomy and a set of typed properties (binary relations). These properties can also be organized in a taxonomy of properties. Two kinds of properties can be distinguished in RDFS: the so-called *relations*, the domain and the range of which are classes and the so-called *attributes*, the domain of which is a class and the range of which is a set of basic values (e.g. Integer, date, String). This distinction is also made in OWL which allows to declare *Object* and *datatype* properties. For example, in the RDFS schema presented in figure 1 and corresponding to a cultural application, we have as relation *Located* having as domain the class *Museum* and as range the class *City*. We also have an attribute*MuseumName* having as domain the class *Museum* and as range the data type *Literal*.

Note that, relations and attributes can be renamed in order to ensure that every attribute and every relation has only one domain and one range.

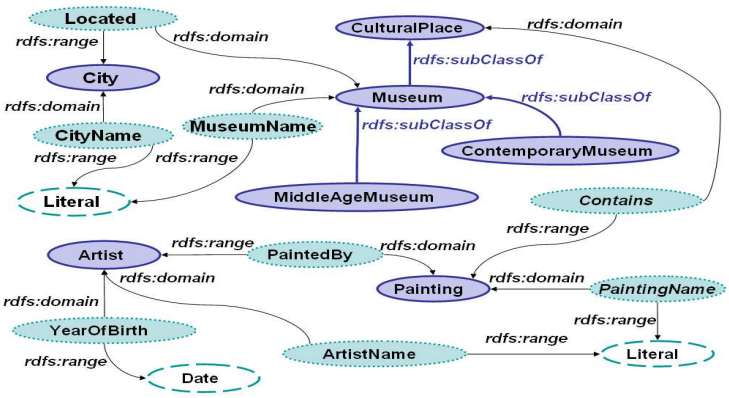

Fig. 1. Example of a RDFS schema

This schema could be extracted from the following relational schema : CulturalPlace(IDCulturalPlace), Museum(IDCulturalPlace#, MuseumName, IDCity#, Category), Painting(IDPainting,PaintingName, IdArtist#, IDCulturalPlace#),

Artist(IDArtist, ArtistName, YearOfBirth), City(IDCity, CityName).

We allow the declaration of constraints expressed in OWL-DL or in SWRL in order to enrich the RDFS schema. The constraints that we consider are of the following types.

- **Constraints of disjunction between classes** $DISJOINT(C, D)$ is used to declare that the two classes C and D are disjoint, for example : $DISJOINT$ ($CulturalPlace$, $Artist$).
- **Constraints of functionality of properties** $PF(P)$ is used to declare that the property P (relation or attribute) is a functional property. It is similar to functional dependencies in relational databases [21]. For example, $PF(Located)$ and $PF(MuseumName)$ express respectively that a museum is located in one and only one city and that a museum has only one name. These constraints can be generalized to a set $\{P_1, \ldots, P_n\}$ of relations or attributes to state a combined constraint of functionality that we will denote $PF(P_1, \ldots, P_n)$. It means that for a set of n references which instantiate the domains of $P_1, \ldots, P_n$ there is only one reference or one value of their ranges. We note that in the RDFS schema (cf. Figure 1) all the attributes and all the relations are functional, except, the relation $Contains$.
- **Constraints of inverse functionality of properties** $PFI(P)$ is used to declare that the property P (relation or attribute) is an inverse functional property. For example, $PFI(Contains)$ expresses that a painting cannot belong to several cultural places. These constraints can be generalized to a set $\{P_1, \ldots, P_n\}$ of relations or attributes to state a combined constraint of inverse functionality that we will denote $PFI(P_1, \ldots, P_n)$. For example, $PFI(MuseumAddress, MuseumName)$ expresses that one address and one museum name cannot be associated to several museums (i.e. both are needed to identify a museum). In summary, the set of inverse functional properties of the RDFS schema (cf. Figure 1) are : $\{PFI(MuseumAddress$,$MuseumName)$, $PFI(PaintingName)$, $PFI(Contains)$, $PFI(ArtistName)$, $PFI(CityName)\}$.

It is important to note that the constraints of disjunction and of simple functionality (i.e., of the form $PF(P)$ or $PFI(P)$) can be expressed in OWL-DL while the constraints stating combined constraints of functionality (i.e., of the form $PF(P_1, \ldots, P_n)$ or $PFI(P_1, \ldots, P_n)$) require the expressive power of SWRL.

2.2 Data description and their constraints

A datum has a reference which has the form of a URI (e.g. `http://www.louvre.fr`, `NS-S1/painting243`), and a description which is a set of RDF facts involving its reference. An RDF fact can be:

- either a class-fact $C(i)$, where C is a class and i is a reference,
- or a relation-fact $R(i_1, i_2)$, where R is a relation and i_1 and i_2 are references,

– or an attribute-fact $A(i, v)$, where A is an attribute, i a reference and v a basic value (e.g. integer, string, date).

The data description that we consider is composed of the RDF facts coming from the data sources enriched by applying the RDFS entailment rules [22]. We consider that the descriptions of data coming from different sources conform to the same RDFS$^+$ schema (possibly after schema reconciliation). In order to distinguish the data coming from different sources, we use the source identifier as the prefix of the reference of the data coming from that source. For example, Figure 2 provides examples of data coming from two RDF data sources S1 and S2 which conform to a same RDFS$^+$ schema describing the cultural application previously mentioned.

Source S1 :
MuseumName(S1_m1, *"LE LOUVRE"*); Contains(S1_m1,S1_p1); Located(S1_m1,S1_c1);
CityName(S1_c1,"Paris"); PaintingName(S1_p1, *"La Joconde"*);

Source S2 :
MuseumName(S2_m1, *"musee du LOUVRE"*); Located(S2_m1,S2_c1); Contains(S2_m1,S2_p1);
Contains(S2_m1, S2_p2); CityName(S2_c1, *"Ville de paris"*);
PaintingName(S2_p1, *"Abricotiers en fleurs"*); PaintingName(S2_p2, *"Joconde'*);

Fig. 2. *Example of RDF data of cultural places domain.*

We consider two kinds of constraints accounting for the Unique Name Assumption (UNA). The UNA states that two data of the same data source having distinct references refer to two different real world entities (and thus cannot be reconciled). Such an assumption is valid when a data source is clean. We have defined another kind of constraint called the Local Unique Name Assumption (denoted LUNA). The LUNA is weaker than the UNA, and states that all the references related to a same reference by relation refer to real world entities that are pairwise distinct. For example, from the facts $Authored(p, a_1)$, ..., $Authored(p, a_n)$ coming from the same data source, we can infer that the references $a_1, \ldots, a_n$ correspond to distinct authors of the paper referred to by p. In practice, it is often the case that all the values of a multi-valued property of a given instance, are provided as a group coming from a single source, like for instance the authors of a given paper, or the list of paintings in a given museum. It is therefore realistic to suppose that this group of values does not contain any duplicate. It is what LUNA means.

3 The reference reconciliation problem

Let S_1 and S_2 be two data sources which conform to the same RDFS$^+$ schema. Let I_1 and I_2 be the two reference sets that correspond respectively to the data

of S_1 and S_2. The problem consists in deciding whether references are reconciled or not reconciled.

A method of reference reconciliation is said complete if it provides a decision for each reference pair $(i_1, i_2) \in I_1 \times I_2$. It is numerical if the decision is based on similarity scores. It will be said symbolic if the yes/no answers for the reconciliation between reference pairs is based on symbolic inferences.

L2R is symbolic but incomplete, while N2R is complete but numerical.

Let *Reconcile*[3] be a binary predicate. *Reconcile*(X, Y) means that the two references denoted by X and Y refer to the same world entity.

The reference reconciliation problem considered in L2R consists in extracting from the set $I_1 \times I_2$ of reference pairs two subsets REC and NREC such that:

$$\begin{cases} REC = \{(i, i')|\ Reconcile(i, i')\} \\ NREC = \{(i, i')|\ \neg Reconcile(i, i')\} \end{cases}$$

The reference reconciliation problem considered in N2R consists in, given a similarity function $Sim_r : I_1 \times I_2 \longrightarrow [0..1]$, and a threshold T_{rec} (a real value in [0..1] given by an un expert, fixed experimentally or learned on a labeled data sample), computing the following set:

$$REC_{n2r} = \{(i, i') \in (I_1 \times I_2) \backslash (REC \cup NREC)|\ Sim_r(i, i') > T_{rec}\}$$

In order to evaluate the quality of the results of a reference reconciliation method, well-known measures in the Information Retrieval (IR) domain can be employed. It consists essentially in *Precision*, *Recall* and *F-Measure* defined as follows:

- *Precision* of a reconciliation method is the ratio of correct reconciliations and non-reconciliations among those found by the method.
- *Recall* of a reconciliation method is the ratio of correct reconciliations and non-reconciliations found by the method among the whole expected set of correct reconciliations and non-reconciliations.
- *F-Measure* of a reconciliation method is computed to balance the recall and precision values : $F - Measure = (2 * Recall * Precision) \div (Recall + Precision)$.

4 L2R: a Logical method for Reference Reconciliation

In this section we present an extended version of the L2R method presented in [23]. The method is based on the inference of facts of reconciliation ($Reconcile(i, j)$) and of non reconciliation ($\neg Reconcile(i', j')$) from a set of facts and a set of rules which transpose the semantics of the data sources and of the schema into logical dependencies between reference reconciliations. Facts of synonymy ($SynVals(v_1, v_2)$) and of no synonymy ($\neg SynVals(u_1, u_2)$) between basic values

[3] Reconcile and not Reconcile can also be expressed in OWL by using *sameAs* and *differentFrom* predicates.

(strings, dates) are also inferred. For instance, the synonymy $SynVals(\text{“}JoDS''$, $\text{“}Journal\ of\ Data\ Semantics'')$ may be inferred. This binary predicate is analogous to the predicate *Reconcile* but applied on basic values.

The L2R distinguishing features are that it is global and logic-based: every constraint declared on the data and on the schema in RDFS+ is automatically translated into first-order logic Horn rules (rules for short) that express dependencies between reconciliations. The advantage of such a logical approach is that if the data are error-free and if the declared constraints are valid, then the reconciliations and non reconciliations that are inferred are correct, thus guaranteeing a 100 % precision of the results.

We first describe the generation of the reconciliation rules. Then we present the generation of the facts and finally the reasoning which is performed on the set of rules and facts to infer reconciliation decisions.

4.1 Generation of the set of reconciliation rules

They are automatically generated from the constraints that are declared on the data sources and on their common schema. We omit to write the quantifiers applied to variables because all the variables are universally quantified in the scope of each rule. According to standard first-order logic conventions, variables will be denoted by lower case letters, and predicate names will start by a capital letter.

Translation of the constraints on the data sources. We introduce the unary predicates Src1 and Src2 in order to label each reference according to its original source ($Srci(x)$ means that the reference x is coming from the source Si).

The UNA assumption, if it is stated on the sources $S1$ and $S2$, is translated automatically by the following four rules:

$$R1 : Src1(x) \wedge Src1(y) \wedge (x \neq y) \Rightarrow \neg Reconcile(x, y)$$

$$R2 : Src2(x) \wedge Src2(y) \wedge (x \neq y) \Rightarrow \neg Reconcile(x, y)$$

$$R3 : Src1(x) \wedge Src1(z) \wedge Src2(y) \wedge Reconcile(x, y) \wedge (x \neq y) \Rightarrow \neg Reconcile(z, y)$$

$$R4 : Src1(x) \wedge Src2(y) \wedge Src2(z) \wedge Reconcile(x, y) \wedge (x \neq y) \Rightarrow \neg Reconcile(x, z)$$

The first two rules express the fact that two distinct references coming from the same source cannot be reconciled. The last ones mean that one reference coming from a source $S2$ (resp. $S1$) can be reconciled with at most one reference coming from a source $S1$(resp. $S2$).
For each relation R, the LUNA assumption is translated automatically by the following rules denoted respectively $R11(R)$ and $R12(R)$:

$$R11(R) : R(z, x) \wedge R(z, y) \wedge (x \neq y) \Rightarrow \neg Reconcile(x, y)$$

$$R12(R) : R(x, z) \wedge R(y, z) \wedge (x \neq y) \Rightarrow \neg Reconcile(x, y)$$

For example, if the LUNA is declared, the two following two rules are generated for the relation *Authored* relating references to papers to references to persons: they express that there is no duplicates in the set of authors of a given paper (respectively in the set of papers of a given author) and that there is no duplicates in the set of papers of a given author.

$$R11(Authored) : Authored(z, x) \wedge Authored(z, y) \wedge (x \neq y) \Rightarrow \neg Reconcile(x, y)$$

$$R12(Authored) : Authored(x, z) \wedge Authored(y, z) \wedge (x \neq y) \Rightarrow \neg Reconcile(x, y)$$

Translation of the schema constraints. For each pair of classes C and D involved in a $DISJOINT(C, D)$ statement declared in the schema, or such that their disjunction is inferred by inheritance, the following rule is generated:

$$R5(C, D) : \ C(x) \ \wedge \ D(y) \ \Rightarrow \ \neg Reconcile(x, y)$$

For example, the following rule is generated if the classes $Painting$ and $Artist$ are declared disjoint:

$$R5(Painting, Artist) : \ Painting(x) \ \wedge \ Artist(y) \ \Rightarrow \ \neg Reconcile(x, y)$$

For each relation R declared as functional by the axiom $PF(R)$, the following rule $R6.1(R)$ is generated :

$$R6.1(R) : Reconcile(x, y) \wedge R(x, z) \wedge R(y, w) \Rightarrow Reconcile(z, w)$$

For example, the following rule is generated concerning the relation *Located* which relates references of museums to references of cities and which is declared functional:

$$R6.1(Located) : Reconcile(x, y) \wedge Located(x, z) \wedge Located(y, w) \Rightarrow Reconcile(z, w)$$

For each attribute A declared as functional by the axiom $PF(A)$, the following rule $R6.2(A)$ is generated :

$$R6.2(A) : Reconcile(x, y) \wedge A(x, z) \wedge A(y, w) \Rightarrow SynVals(z, w)$$

The binary predicate *SynVals* replaces the predicate *Reconcile* in the conclusion of the rule. For example, the following rule is generated concerning the attribute $MuseumName$ which relates references of museums to their name and which is declared functional:

$$R6.2(MuseumName) : Reconcile(x, y) \wedge MuseumName(x, z) \wedge MuseumName(y, w)$$
$$\Rightarrow SynVals(z, w)$$

For each relation R declared as inverse functional by the constraint $PFI(R)$, the following rule $R7.1(R)$ is generated:

$$R7.1(R) : Reconcile(x, y) \wedge R(z, x) \wedge R(w, y) \Rightarrow Reconcile(z, w)$$

For each attribute A declared as inverse functional by the constraint $PFI(A)$, the following rule $R7.2(A)$ is generated:

$$R7.2(A) : SynVals(x, y) \wedge A(z, x) \wedge A(w, y) \Rightarrow Reconcile(z, w)$$

Likewise, analogous rules are generated for translating constraints $PF(P_1, \ldots, P_n)$ of combined constraints of functionality and $PFI(P_1, \ldots, P_n)$ of combined constraints of inverse functionality. $PF(P_1, \ldots, P_n)$, where all the P_i's are relations, is translated by the following rule:

$$R7.1(P_1, \ldots, P_n) : \bigwedge_{i \in [1..n]} [P_i(z, x_i) \wedge P_i(w, y_i) \wedge Reconcile(x_i, y_i)] \Rightarrow Reconcile(z, w)$$

If some P_i's are attributes, the corresponding $Reconcile(x_i, y_i)$ must be replaced by $SynVals(x_i, y_i)$. For example, the declaration $PF(PaintedBy, PaintingName)$ states a composed functional dependency which expresses that the artist who painted it jointly with its name functionally determines a painting. It is translated in the following rule:
$R7.1(PaintedBy, PaintingName)$:
$PaintedBy(z, x_1) \wedge PaintedBy(w, y_1) \wedge Reconcile(x_1, y_1)$
$\wedge PaintingName(z, x_2) \wedge PaintingName(w, y_2) \wedge SynVals(x_2, y_2) \Rightarrow Reconcile(z, w)$
Similarly, $PFI(P_1, \ldots, P_n)$, where all the P_i's are relations, is translated into the rule:

$$R7.2(P_1, \ldots, P_n) : \bigwedge_{i \in [1..n]} [P_i(x_i, z) \wedge P_i(y_i, w) \wedge Reconcile(x_i, y_i)] \Rightarrow Reconcile(z, w)$$

Transitivity rule : it allows inferring new reconciliation decisions by applying transitivity on the set of already inferred reconciliations. Its logical semantics is:

$$R8 : Reconcile(x, y) \wedge Reconcile(y, z) \Rightarrow Reconcile(x, z)$$

We note that this rule is generated only if the UNA constraint is not stated on the data sources. Indeed, when the UNA is stated, a reference can not be reconciled with more than one reference. Therefore, the transitivity rule is not meaningful in this setting. We do not generate a rule for expressing transitivity between synonymies values, since, according to Fischer [24], the synonymy between basic values is not transitive because of polysemy.

4.2 Reasoning method for reference reconciliation

In order to infer sure reconciliation and non-reconciliation decisions, we apply an automatic reasoning method based on the resolution principle [25]. This method applies to the clausal form of the set of rules described in Section 4.1 and a set of facts describing the data which is generated as follows.

Generation of the set of facts . The set of RDF facts corresponding to the description of the data in the two sources S_1 and S_2 is augmented with the generation of:

- new class-facts, relation-facts and attribute-facts derived from the domain and range constraints that are declared in RDFS for properties, and from the subsumption statements between classes and properties that are stated in RDFS. For example if the fact *ContemporaryMuseum(i)* is present in one of the sources, the class-facts $Museum(i)$ and $CulturalPlace(i)$ are added to the description of that source;
- facts of the form $Src1(i)$ and $Src2(j)$ for each reference $i \in I_1$ and each reference $j \in I_2$,
- synonymy facts of the form $SynVals(v_1, v_2)$ for each pair (v_1, v_2) of basic values that are identical (up to some punctuation or case variations): for instance, the fact *SynVals("La Joconde", "la joconde")* is added because these two values differ only by two capital letters,
- non synonymy facts of the form $\neg SynVals(v_1, v_2)$ for each pair (v_1, v_2) of distinct basic values of a functional attribute (PF) for which it is known that each possible value of this attribute has a single form . For instance, $\neg SynVals(''2004'','' 2001'')$, $\neg SynVals(''France'', ''Algeria'')$ are added.

Resolution-based algorithm for reference reconciliation. The reasoning is applied to $\mathcal{R} \cup \mathcal{F}$: the set of rules (put in clausal form) and the set of facts generated as explained before. It aims at inferring all unit facts in the form of $Reconcile(i, j)$, $\neg Reconcile(i, j)$, $SynVals(v1, v2)$ and $\neg SynVals(v1, v2)$.

The resolution is a reasoning method for theorem proving by a successive application of the *resolution rule*[26] on the set of clauses. Several resolution strategies have been proposed so that the number of computed resolutions to obtain the theorem proof are reduced (for more details about these strategies see [26]). We have chosen to use the *unit resolution*[27], defined as follows:

Definition 1. *Unit resolution: it is a resolution strategy where at least one of the two clauses involved in the resolution is a unit clause, i.e. reduced to a single literal.*

The unit resolution is complete for refutation[4] in the case of Horn clauses without functions [27]. Furthermore, the unit resolution method is linear with respect to the size of clause set [28].

The Conjunctive Normal Form (CNF) of the knowledge base $\mathcal{R} \cup \mathcal{F}$ is made of Horn clauses and contains a lot of unit clauses. It is important to notice that the reconciliation and synonymy rules though having negative conclusions still correspond to Horn clauses. For all these reasons, unit resolution is a method which is appropriate for our problem.

[4] Proving by refutation that a literal L is logically entailed from a theory T is viewed as the unsatisfiability of the theory $T \cup \{\neg L\}$, i.e. deduce the empty clause ($\square$).

The unit resolution algorithm that we have implemented consists in computing the set $SatUnit(\mathcal{R} \cup \mathcal{F})$ of unit instantiated clauses contained in $\mathcal{F}$ or inferred by unit resolution on $\mathcal{R} \cup \mathcal{F}$. Its termination is guaranteed because there are no function symbols in $\mathcal{R} \cup \mathcal{F}$. Its completeness for deriving all the facts that are logically entailed is stated in the following theorem. Because of the limited form of inequalities appearing in the rules (and thus in the clauses) we avoid to use paramodulation in combination with resolution by a preprocessing step on $\mathcal{R} \cup \mathcal{F}$. This consists in generating all the propositional rules that can be obtained from $\mathcal{R}$ by matching their conditions to facts in $\mathcal{F}$ with substitutions satisfying the inequality statements. This results into a set of clauses (a subset of which being totally instantiated) without inequalities. A simple optimization is also implemented that prunes all the unit clauses of the form $Reconcile(i, i)$ that may be generated: only unit clauses of the form $Reconcile(i, j)$ such that $i \neq j$ are then used as resolvents.

Theorem 1. – *Completeness of unit resolution for deriving facts from function-free Horn clauses.*
Let $\mathcal{R}$ be a set of Horn clauses without functions. Let $\mathcal{F}$ be a set of ground unit clauses. If $\mathcal{R} \cup \mathcal{F}$ is satisfiable then:

$$\forall p(\boldsymbol{a}),\ (\mathcal{R} \cup \mathcal{F} \models p(\boldsymbol{a})) \Rightarrow (p(\boldsymbol{a}) \in SatUnit(\mathcal{R} \cup \mathcal{F})),$$

where, $p(\boldsymbol{a})$ is a ground unit clause.

The theorem relies on the assumption that $\mathcal{R} \cup \mathcal{F}$ is satisfiable. $\mathcal{R} \cup \mathcal{F}$ is the logical transposition of the constraints that are declared on the schema and the data. Therefore, if those constraints are correct and if the data are error-free then $\mathcal{R} \cup \mathcal{F}$ is satisfiable. In any case, our resolution-based algorithm will detect if $\mathcal{R} \cup \mathcal{F}$ is unsatisfiable. A by-product of our logical approach is to detect whether the set of declared constraints on the schema and on the data sources is contradictory. If that it the case, this set of constraints must be revised by the database administrator in charge of the data integration.

Other reasoners, like for instance description logic reasoners, could be used for the derivation of reconciliation facts. However, description logics are not specially appropriate to express some of the reconciliation rules that we consider, which require explicit variable bindings. In addition, up to our knowledge, the existing description logic reasoners are not guaranteed to be complete for the computation of prime implicates.

Illustrative example. We now illustrate on the example of data and RDFS$^+$ schema, given in Section 2, the resolution-based reasoning for the reference reconciliation. We will also show how reconciliation and non-reconciliation decisions are inferred and chained.

The successive application of the unit resolution on the knowledge base $\mathcal{R} \cup \mathcal{F}$, presented in the Figure 3, allows inferring the set of (non) reconciliation and synonymy facts presented in Figure 3.

The clauses R1 and R2 allow inferring a set of non-reconciliations between all the references coming from the same source, e.g. $\neg Reconcile(S1\_m1, S1\_c2)$. The clauses $R5$ translating the disjunction between classes, allow inferring non-reconciliations between references which instantiate disjoint classes, (e.g. $\neg$ $Reconcile$ $(S1\_m1, S2\_p1)$) is obtained thanks to the clause $R5(Painting, Museum)$. Furthermore, the successive application of the unit resolution between the unit clauses contained in $\mathcal{F}$ and the clause $R7.2(PaintingName)$ allows inferring $Reconcile(S2\_p1, S1\_p1)$, which means that the two paintings $S2\_p1$ and $S1\_p1$ refer to the same painting. Then, the museums $S1\_m1$ and $S2\_m2$ which contain these paintings are reconciled as well, thanks to the clause $R7.1(Contains)$. The propagation of the new reconciliation facts allows inferring the synonymie fact $SynVals(''Le\ LOUVRE'', ''musee\ du\ LOUVRE'')$ thanks to the clause $R6.2(MuseumName)$. Finally, thank to the clause $R6.1(Located)$ we entail the reconciliation $Reconcile$ $(S1\_c1, S2\_c1)$ of the two cities. Therefore, this new reconciliation leads to the entailment of the synonymy $SynVals(''ville\ de\ Paris'', ''Paris'')$ thanks to the clause $R6.2(CityName)$.

Knowledge base

$\mathcal{R}$= { $R1 : \neg\ Src1(x) \vee \neg\ Src1(y) \vee \neg Reconcile(x, y)$
$R2 : \neg\ Src2(x) \vee \neg\ Src2(y) \vee \neg Reconcile(x, y)$ ...
$R5(Painting, City) : \neg\ Painting(x) \vee \neg\ City(y) \vee \neg Reconcile(x, y)$
$R5(Painting, Museum) : \neg\ Painting(x) \vee \neg\ Museum(y) \vee \neg Reconcile(x, y)$
$R5(City, Museum) : \neg\ City(x) \vee \neg\ Museum(y) \vee \neg Reconcile(x, y)$ ...
$R6.1(Located) : \neg\ Reconcile(x, y) \vee \neg\ Located(x, z) \vee \neg\ Located(y, w) \vee Reconcile(z, w)$
$R6.2(MuseumName) : \neg\ Reconcile(x, y) \vee \neg\ MuseumName(x, z)$
$\vee \neg\ MuseumName(y, w) \vee SynVals(z, w)$
$R6.2(CityName) : \neg\ Reconcile(x, y) \vee \neg\ CityName(x, z) \vee \neg\ CityName(y, w) \vee SynVals(z, w)$ ...

$R7.2(PaintingName) : \neg SynVals(x, y) \vee \neg\ PaintingName(z, x) \vee$
$\neg\ PaintingName(w, y) \vee Reconcile(z, w)$
$R7.1(Contains) : \neg Reconcile(x, y) \vee \neg\ Contains(z, x) \vee \neg\ Contains(w, y) \vee Reconcile(z, w)$ ...}
$\mathcal{F}$= { $MuseumName(S1\_m1, \text{“}LE\ LOUVRE'')$; $Contains(S1\_m1, S1\_p1)$;
$Contains(S2\_m1, S2\_p2)$; $CityName(S2\_c1, \text{“}Ville\ de\ paris'')$; ...
$Src1(S1\_m1)$; $Src1(S1\_p1)$; $Src1(S1\_c1)$; $Src2(S2\_m1)$; $Src2(S2\_p1)$; $Src2(S2\_c1)$
$SynVals(''La\ Joconde'', ''\ Joconde'')$}

Reference reconciliation result

SatUnit$(\mathcal{R} \cup \mathcal{F}) = \{...;$
$\neg Reconcile(S1\_m1, S1\_c2)$; $\neg Reconcile(S1\_m1, S1\_p1)$; $\neg Reconcile(S1\_p1, S1\_c1)$;
$\neg Reconcile(S2\_m1, S2\_p1)$; $\neg Reconcile(S2\_m1, S2\_c1)$; $\neg Reconcile(S2\_c1, S2\_p1)$;
$\neg Reconcile(S1\_m1, S2\_p1)$; $\neg Reconcile(S1\_m1, S2\_c1)$; $\neg Reconcile(S1\_p1, S2\_c1)$;
$\neg Reconcile(S1\_c1, S2\_p1)$;
$Reconcile(S2\_p1, S1\_p1)$; $Reconcile(S1\_m1, S2\_m1)$; $Reconcile(S1\_c1, S2\_c1)$;
$SynVals(\text{“}musee\ du\ LOUVRE'', \text{“}LE\ LOUVRE'')$; $SynVals(''ville\ de\ Paris'', ''\ Paris'')\}$

Fig. 3. *Illustrative exemple of unit resolution-based reference reconciliation*

4.3 Dictionary of synonyms and no synonyms.

The set of synonymies and no synonymies between basic values inferred by the the reference reconciliation algorithm are saved in two dictionaries. The dictionaries can also be exploited by a numerical method for reference reconciliation based on similarities between strings. We will see (see Section 5) how these (no) synonymies can be used by N2R method.

We distinguish different kinds of synonyms: (i) Codes, like *1* for *yes* *75* for *Paris* and *(*)* for *star*. (ii) Abbreviations, like *apt* for *appartement*, or acronyms like *ACM* for *Association for Computing Machinery*. (iii) Real synonyms, like *good* for *comfortable*. (iv) Translations, like *Royaume-Uni* for *United Kingdom* .

The (no) synonymies can be viewed as knowledge learnt in an automatic and a unsupervised way. Indeed, this allows our method to capitalize its experience by learning more and more on the syntactic variations that characterize an application domain.

5 N2R: a Numerical method for Reference Reconciliation

In this section we describe the numerical method for reference reconciliation (N2R) that we have designed and implemented. Like existing numerical methods (e.g., [1, 6]), it computes a similarity score for each pair of references. However, N2R has two main distinguishing characteristics. First, it is fully unsupervised: in contrast with the existing methods, it does not require any training phase from manually labeled data to set up coefficients or parameters. Second, it is based on equations that model the influence between similarities. In the equations, each *variable* represents the (unknown) similarity between two references while the similarities between values of attributes are *constants* that are computed by using standard similarity measures on strings or on sets of strings. The *functions* modeling the influence between similarities are a combination of *maximum* and *average* functions in order to take into account the constraints of functionality and inverse functionality declared in the RFDS$^+$ schema in an appropriate way.

Solving this equation system is done by an iterative method inspired from the *Jacobi* method [20], which is fast converging on linear equation systems. The point is that the equation system that results for modeling the global influence of similarities is not linear, due to the use of the *max* function for the numerical translation of the functionality and inverse functionality axioms declared in the RFDS$^+$ schema. Therefore, we had to prove the convergence of the iterative method for solving the resulting non linear equation system.

N2R can be applied alone or in combination with L2R. In this case, the results of non-reconciliation inferred by L2R are exploited for reducing the reconciliation space, i.e., the size of the equation system to be solved by N2R. In addition, the results of reconciliations and of synonymies or non synonymies inferred by L2R are used to set the values of the corresponding constants or variables in the equations.

We first use a simple example to illustrate how the equation system is built from the data descriptions related to the references to reconcile. We then introduce the notations and the similarity measures that are used in order to distinguish the different types of constants and functions involved in the equations. Finally, we describe in the general case how the equation system is built from the data descriptions, and we provide the iterative method for solving it.

5.1 Illustrative example

Let us consider the data descriptions of Figure 2. They conform to the RFDS$^+$ schema given in the Figure 1, the constraints of which are described in Section 2.1. Let us assume that the UNA is stated in both the sources S1 and S2.

Let us suppose that L2R has been applied, resulting on the non-reconcilations of all the pairs of references coming from the same source and those belonging to two disjoint classes. The only remaining pairs of references to consider for N2R are then:

$< S1\_m1, S2\_m1 >, < S1\_c1, S2\_c1 >, < S1\_p1, S2\_p1 >$ and $< S1\_p1, S2\_p2 >$.

The similarity score $Sim_r(ref, ref')$ between the references ref and ref' of each of those pairs is modeled by a variable:

- x_1 models $Sim_r(S1\_m1, S2\_m1)$,
- x_2 models $Sim_r(S1\_p1, S2\_p1)$,
- x_3 models $Sim_r(S1\_p1, S2\_p2)$,
- x_4 models $Sim_r(S1\_c1, S2\_c1)$.

We obtain the following equations that model the dependencies between those variables from the relations relating the corresponding references and the constraints declared on them in the schema:

$x_1 = max(0.68, x_2, x_3, x_4/4)$
$x_2 = max(0.1, x_1/2)$
$x_3 = max(0.9, x_1/2)$
$x_4 = max(0.42, x_1)$

The first equation expresses that the variable x_1:

- strongly and equally depends on the variables x_2 and x_3, and also on 0.68, which is the similarity score between the two strings "LE LOUVRE" and "musee du LOUVRE" computed by the Jaro-Winkler function [29],
- weakly depends on x_4.

The reason of the strong dependencies is that $Contains$ is an inverse functional relation (a painting is contained in only one museum) relating $S1\_m1$ and $S2\_m1$ (the similarity of which is modeled by x_1) to $S1\_p1$ for $S1\_m1$ and $S2\_p1$ and $S2\_p2$ for $S2\_m1$, and $MuseumName$ is a functional attribute (a museum has only one name) relating $S1\_m1$ and $S2\_m1$ respectively to the two strings "LE LOUVRE" and "musee du LOUVRE".

The weak dependency of x_4 onto x_1 is expressed by the term $x_4/4$ in the equation, where the ratio 1/4 comes from that there are 4 properties (relations or

attributes) involved in the data descriptions of $S1\_m1$ and $S2\_m1$. The dependency of x_4 onto x_1 is weaker than the previous ones because x_4 expresses the similarity between the two cities in which the museums modeled by x_1 are located and *Located* is not an inverse functional relation.
Conversely, because *Located* is functional, x_1 has a strong influence on the variable x_4, which translates into the second equation, where 0.42 is the Jaro-Winkler score of similarity between the strings "Paris" and "Ville de Paris" which are the respective values associated to the references $S1\_c1$ and $S2\_c1$ by the inverse functional attribute $CityName$.
The weak influence of x_1 on x_2 and x_3 is due to the fact that $Contains$ is not functional. It is expressed through the two last equations, where 0.1 is the Jaro-Winkler score of similarity between the strings "*La Joconde*" and "*Abricotiers en fleurs*", and 0.9 is the Jaro-Winkler score of similarity between the strings "La Joconde" and "*Joconde*".

5.2 Notations and similarity measures on (sets of) basic values

As illustrated in the previous example, the constants in the equations are similarities between basic values (e.g., String, Date, Numbers) or between sets of basic values, for which a lot of similarity measures have been extensively studied [29].

Similarity measures on basic values. We denote Sim_v the similarity measure used to compute the similarity score between two basic values.
For pairs of basic values $< v_1, v_2 >$ such that $SynVal(v_1, v_2)$ (respectively $\neg SynVal(v_1, v_2)$) has been inferred by L2R , we set: $Sim_v(v_1, v_2) = 1$ (respectively $Sim_v(v_1, v_2) = 0$).
For computing the similarity scores between basic values that are not dealt with by L2R, depending on the attributes and the characteristics of their values (e.g short/long values, values containing abbreviations), we set Sim_v to the most appropriate similarity measure according to [29].

***SSim*$_v$: the similarity measure on sets of basic values.** Some attributes are multi-valued (e.g. a person can have a set of phone numbers).

$SSim_v(S1, S2)$ denotes the similarity score between the two sets of basic values $S1$ and $S2$.

In order to compute the similarity between two sets of values we need to take into account their size and also the similarity scores of the pairs of values formed from these two sets.

We propose a similarity measure that we have named *SoftJaccard* which is inspired from *SoftTFIDF* measure defined by [29] and from the *Jaccard* measure. *SoftJaccard* allows relaxing the constraint of equality of the tokens used in *Jaccard*. $CLOSE_v(S1, S2, \theta)$ represents the values $(v1, v2) \in S1 \times S2$ that have a similarity score $Sim_v(v1, v2) > \theta$. *SoftJaccard* is defined by the expression:

$$SoftJaccard(S1, S2, \theta) = \frac{|CLOSE_v(S1, S2, \theta)|}{|S1|}, \ with\ |S1| \geq |S2|$$

Example 1. $SoftJaccard(\{"Fatiha\ Sais","Marie{-}Christine\ Rousset","Helene\ Gagliardi"\}, \{"Nathalie\ Pernelle","Fatiha\ Sais"\}), 0.7) = 1/3$

Common attributes and relations: definitions and notations. Given a pair of references $< i, i' >$, we will denote:

- $CAttr(< i, i' >)$ the set of its *common attributes*: A is a common attribute to $< i, i' >$ if there exists atleast a fact $A(i, v)$ in the data description of i and also atleast a fact $A(i', v')$ in the data description of i'.
- $CRel(< i, i' >)$ the set of its *common relations*: R is a common relation to $< i, i' >$ if there exists atleast a fact $R(i, r)$ in the data description of i and also atleast a fact $R(i', r')$ in the data description of i'.

Example 2. In Figure 2, we have: $CAttr(< S1\_m1, S2\_m1 >) = \{MuseumName\}$
$CRel(< S1\_m1, S2\_m1 >) = \{Located, Contains\}$

Among $CAttr(< i, i' >)$ and $CRel(< i, i' >)$ we need to distinguish those which are (inverse) functional from those which are not:

- $FD_A(< i, i' >)$ denotes the set of common attributes of the reference pair $< i, i' >$ that are inverse functional.
- $FD_R(< i, i' >)$ denotes the set of common relations of the reference pair $< i, i' >$ that are functional or inverse functional.

Example 3. $FD_A(< S1\_p1, S2\_p1 >) = \{PaintingName\}$
$FD_R(< S1\_c1, S2\_c1 >) = \{Located\}$

These sets are generalized by considering the generalized constraints of (inverse) functionality which involve sets of attributes and relations. We note these sets $FD^M_A(< i, i' >)$ and $FD^M_R(< i, i' >)$. Among those attributes and relations that are not functional or inverse functional, we distinguish those for which i and i' are mono-valued from those for which i or i' are multi-valued:

- $NFD_A(< i, i' >)$ denotes the set of common attributes of $< i, i' >$ that are not inverse functional but that are mono-valued for i and for i'.
- $NFD^*_A(< i, i' >)$ denotes the set of common attributes of $< i, i' >$ that are not inverse functional and that are multi-valued for i or for i'.
- $NFD_R(< i, i' >)$ denotes the set of common relations of $< i, i' >$ that are not (inverse) functional but that are mono-valued for i and for i'.
- $NFD^*_R(< i, i' >)$ denotes the set of common relations of $< i, i' >$ that are not (inverse) functional and that are multi-valued for i or for i'.

5.3 The equations modeling the dependencies between similarities.

The variables in the equation system. For each pair of references, its similarity score is modeled by a variable x_i and the way it depends on other similarity scores is modeled by an equation: $x_i = f_i(X)$, where n is the number of reference pairs for which we apply N2R, and $X = (x_1, x_2, \ldots, x_n)$.

When a reference pair $< ref_h, ref'_h >$, represented by a similarity score x_j, is involved in the similarity equation of another reference pair $< ref, ref' >$ corresponding to the variable x_i, a contextual suffix i is added to the variable x_j. In addition, we denote by three secondary suffixes the three types of functional dependencies that we distinguish : *df* for a functional dependency, *dfm* for a multiple functional dependency, and *ndf* for non functional dependency. Thus, for the i-th reference pair $< ref, ref' >$ we distinguish the following variables x_{ij-df}, x_{ij-dfm} and x_{ij-ndf} they depend on.

In addition, we define a variable XS_{ij-ndf} in order to express the similarity score of the reference sets S1 and S2 of the j-th common relation of the i-th reference pair $< ref, ref' >$. This variable is defined only when the relation belongs to $NFD^*_R(< ref, ref' >)$. Its value is obtained by the function $SSim_r(S1, S2)$[5].

The constants in the equation system. They represent the similarity score of basic values. For each pair of values (v, v') of the j-th common attribute of the i-th reference pair, a constant b_{ij-df}, b_{ij-dfm} or b_{ij-ndf} is assigned. These constants represent the similarity score obtained by the function $Sim_v(v, v')$.

In addition, we define a constant BS_{ij-ndf} in order to express the similarity score of value sets S1 and S2 of the j-th common attribute. This constant is defined only when the attribute belongs to $NFD^*_A(< ref, ref' >)$. Its value is obtained by the function $SSim_v(S1, S2)$.

The equations. Each equation $x_i = f_i(X)$ is of the form:

$$f_i(X) = \max(f_{i-df}(X), f_{i-ndf}(X))$$

The function $f_{i-df}(X)$ is an aggregation function of the similarity scores of the value pairs and the reference pairs of attributes and relations with which the i-th reference pair is functionally dependent. The function $f_{i-ndf}(X)$ allows to aggregate the similarity scores of the values pairs (and sets) and the reference pairs (and sets) of attributes and relations with which the i-th reference pair is not functionally dependent.

Modeling the influence of functional attributes and functional relations. $f_{i-df}(X)$ is defined by the maximum of similarity scores of the value pairs and reference pairs of attributes and relations with which the two references ref and ref' are functionally dependent. The maximum function allows propagating the similarity scores of the values and the references having a strong impact. $f_{i-df}(X)$ is defined as follows:

$$f_{i-df}(X) = \max\Big(\bigcup_{j=0}^{j=|FD_A(<ref,ref'>)|}(b_{ij-df}), avg\Big(\bigcup_{j=0}^{j=|FD^M_A(<ref,ref'>)|}(b_{ij-dfm})\Big),$$

[5] *SoftJaccard* applied on the sets of references.

$$\bigcup_{j=0}^{j=|FD_R(<ref,ref'>)|}(x_{ij-df}), avg(\bigcup_{j=0}^{j=|FD_R^M(<ref,ref'>)|}(x_{ij-dfm})))$$

Note that the similarity scores of the values and the references of attributes and relations which belongs to a same multiple functional dependency are first aggregated by an *average* function.

Modeling the influence of non functional attributes and non functional relations. $f_{i-ndf}(X)$ is defined by a weighted average of the similarity scores of the values and the references of attributes and relations with which the two references ref and ref' are not functionally dependent. $f_{i-ndf}(X)$ is defined as follows:

$$f_{i-ndf}(X) = \sum_{j=0}^{j=|NFD_A(<ref,ref'>)|}(\lambda_{ij} * b_{ij-ndf}) + \sum_{j=0}^{j=|NFD_A^*(<ref,ref'>)|}(\lambda_{ij} * BS_{ij-ndf}) +$$

$$\sum_{j=0}^{j=|NFD_R(<ref,ref'>)|}(\lambda_{ij} * x_{ij-ndf}) + \sum_{j=0}^{j=|NFD_R^*(<ref,ref'>)|}(\lambda_{ij} * XS_{ij-ndf})$$

Where λ_{ij} represents the weight of the j-th attribute or relation in the similarity computation of the i-th reference pair. Since we have neither expert knowledge nor training data, λ_{ij} is computed in function of the number of the common attributes and relations.

5.4 Iterative algorithm for reference pairs similarity computation

To compute the similarity scores, we have implemented an iterative resolution method inspired from the *Jacobi* method [20] for the resolution of linear equation systems. At each iteration, the method computes the variables values by using those computed in the precedent iteration.

Iterative similarity scores computation. Starting from an initial vector $X^0 = (x_1^0, x_2^0, \ldots, x_n^0)$, the value of the vector X at the k-th iteration is obtained by the expression : $X^k = F(X^{k-1})$. At each iteration k we compute the value of each x_i^k : $x_i^k = f_i(x_1^{k-1}, x_2^{k-1}, \ldots\ldots, x_n^{k-1})$ until a fix-point with precision ϵ is reached. The fix-point is reached when : $\forall\ i,\ |x_i^k - x_i^{k-1}| \leq \epsilon$. The value of ϵ is fixed at a very small positive real number. The more ϵ value is small the more the set of reconciliations may be large.

The complexity of this method is in (n^2) for each iteration, where n is the number of variables. The same kind of approach has been followed by [30] in the context of schema matching. It is important to note that the convergence of the *Jacobi* method is not always guaranteed. We have proved its convergence for the resolution of our equation system.

Illustration of the iterative similarity computation. We illustrate the similarity computation on the system of equations obtained from the data descriptions of Figure2. The constants, the variables and the weights are given in the table 1. The constants correspond to the similarity scores of pairs of basic values computed by using the Jaro-Winkler measure [29]. The weights are computed in function of the number of common attributes and common relations of the reference pairs. We assume that point-fix precision ϵ is equal to 0.005.

Variables	Constants	Weights
$x_1 = Sim_r(S1\_m1, S2\_m1)$	$b_{11} = Sim_v(\text{“}LOUVRE''\text{, “}Musee\ du\ LOUVRE'''') = 0.68$	$\lambda_{11} = \frac{1}{4}$
$x_2 = Sim_r(S1\_p1, S2\_p1)$	$b_{21} = Sim_v(\text{“}La\ Joconde''\text{, “}Abricotiers\ en\ fleurs'') = 0.1$	$\lambda_{21} = \frac{1}{2}$
$x_3 = Sim_r(S1\_p1, S2\_p2)$	$b_{31} = Sim_v(\text{“}La\ Joconde''\text{, “}Joconde'') = 0.9$	$\lambda_{31} = \frac{1}{2}$
$x_4 = Sim_r(S1\_c1, S2\_c1)$	$b_{41} = Sim_v(\text{“}Paris''\text{, “}Ville\ de\ Paris'') = 0.42$	$\lambda_{41} = \frac{1}{2}$

Table 1. *The variables, the constants and the weights of the equation system*

The equation system is the one given in Section 5.1. The different iterations of the resulting similarity computation are provided in Table 2.

Iterations	0	1	2	3	4
$x_1 = \max(0.68, x_2, x_3, \frac{1}{4} * x_4)$	0	0.68	0.9	0.9	0.9
$x_2 = \max(0.1, \frac{1}{2} * x_1)$	0	0.1	0.34	0.45	0.45
$x_3 = \max(0.9, \frac{1}{2} * x_1)$	0	0.9	0.9	0.9	0.9
$x_4 = \max(0.42, x_1)$	0	0.42	0.68	0.9	0.9

Table 2. *Illustrative example – Iterative similarity computation.*

The solution of the equation system is $X = (0.9, 0.45, 0.9, 0.9)$. This corresponds to the similarity scores of the four reference pairs. The fix-point has been reached after four iterations. The error vector is then equal to 0.

This example, shows how the similarity scores are propagated between the reference pairs through the relations having a strong impact on reference pairs. For instance, at the iteration (2), the similarity 0.9 of the painting pair $< S1\_p1, S2\_p2 >$ has been propagated to the museum pair $< S1\_m1, S2\_m1 >$ through the relation *contains* which belongs to $FD_R(< S1\_m1, S2\_m1 >)$. At the following iteration (3) the same similarity score has been propagated to city pair $< S1\_c1, S2\_c1 >$ through the relation *located* which belongs to $FD_R(< S1\_c1, S2\_c1 >)$. Furthermore, we have a weaker propagation of the similarity scores. For example, at the iteration (3) the similarity score 0.90 of the museums obtained at the iteration (2) has been propagated to the pair of paintings $< S1\_p1, S2\_p1 >$. Its similarity score grows to 0.45.

If we fix the reconciliation threshold T_{rec} at 0.80, then we obtain three reconciliation decisions: two cities, two museums and two paintings.

We note that the reconciliation threshold is empirically fixed. In supervised approaches [11, 3], it is learnt on labeled data.

6 Experiments

The logical method (L2R) and the numerical one (N2R) have been implemented and tested on data sets related to two different domains: the tourism domain and the scientific publications.

6.1 Presentation of the data sets (HOTELS and Cora)

The first real data set HOTELS, provided by an industrial partner, corresponds to a set of seven data sources which leads to a pairwise data integration problem of 21 pairs of data sources. These data sources contain 28,934 references to hotels located in Europe. The UNA is stated for each source.The hotel descriptions in the different sources are very heterogeneous. First, the instantiated properties are different from one to another. Second, the basic values are multilingual, contain abbreviations, and so on.

The second data set Cora[6] (used by [1] and [14]) is a collection of 1295 citations of 112 different research papers in computer science. In this data set, the objective of the reference reconciliation is the cleaning of a given data source (i.e. duplicates elimination). The reference reconciliation problem applies then to $I \times I$ where I is the set of references of the data source S to be cleaned. For this data set, the UNA is not stated and the RDF facts describe references which belong to three different classes (*Article*, *Conference*, *Person*).

The RDFS+ schemas: HOTELS conforms to a RDFS schema of tourism domain, which is provided by the industrial partner. We have added a set of disjunction constraints (e.g. DISJOINT(Hotel, Service)), a set of (inverse) functional property constraints (e.g. *PF(EstablishmentName)*, *PF(Name)*, *PFI(EstablishmentName, AssociatedAddress)*).

For the Cora data set, we have designed a simple RDFS schema on the scientific publication domain, which we have enriched with disjunction constraints (e.g. DISJOINT(Article, Conference)), a set of functional property constraints (e.g. *PF(Published)*, *PF(ConfName)*) and a set of inverse functional property constraints (e.g. *PFI(Title, Year, Type)*, *PFI(ConfName, ConfYear)*).

For the Cora data set, the expected results for reference reconciliation are provided. Therefore, the recall and the precision can be easily obtained by computing the ratio of the reconciliations or non-reconciliations obtained by L2R and N2R among those that are provided.

For the HOTELS data set, we have manually detected the correct results of reconciliations or non-reconciliations between the references of two data sources containing respectively 404 and 1392 references to hotels.

[6] another version of Cora is available at `http://www.cs.umass.edu/~mccallum/data/cora-refs.tar.gz`

6.2 L2R results

Since the set of reconciliations and the set of non-reconciliations are obtained by a logical resolution-based algorithm the precision is of 100% by construction. Then, the measure that it is meaningful to evaluate in our experiments is the recall.

In the following, we summarize the results obtained on the HOTELS data set and then those obtained on the *Cora* data set. We emphasize the impact on the recall of increasing the expressiveness of the schema by adding constraints.

	RDFS+		**RDFS+ & {DA or DP}**	
	HOTELS	Cora	HOTELS	Cora
Recall (REC)	54 %	52.7 %	54 %	52.7 %
Recall (NREC)	8.2 %	50.6 %	75.9 %	94.9 %
Recall	8.3 %	50.7 %	75.9 %	94.4 %
Precision	100 %	100 %	100 %	100 %

Fig. 4. *L2R results on HOTELS and Cora data sets*

L2R Results on HOTELS data set. In the figure 4, we show the recall that we have obtained on the two sources on which we have manually detected the reconciliation and no reconcilation pairs. We distinguish the recall computed only on the set of reconciled references (REC) and only on not reconciled references (NREC). To examine the 532368 reference pairs we have first automatically extracted the name and address of the hotels belonging to the smaller source. Then, we have used the standard string search commands of Unix to search in the file of the second source the truncated corresponding strings (in order to be robust with typographical errors). Then, we have set the correct no reconcilations to be the remaining pairs.

As it is shown in the column named "RDFS+ (HOTELS)" of the figure 4, we have obtained a recall of 8.3%. If we only consider the reconciliations subset (REC) the recall is 54%. The REC subset corresponds to the reconciliations inferred by exploiting the inverse functional constraint *PFI(EstablishmentName, AssociatedAddress)*. It is important to emphasize that those reconciliations are inferred in spite of the irregularities in the data descriptions: not valued addresses and a lot of variability in the values, in particular in the addresses : "*parc des fees*" vs. "*parc des fees, (nearby Royan)*". In addition, in one of the data sources, several languages are used for the basic values.

If we only consider the non-reconciliations subset (NREC) the recall is 8.2%. Actually, the only rules that are likely to infer no reconciliations are those translating the UNA assumption. Now, if we enrich the schema just by declaring pairwise disjoint specializations of the *Hotel* class (by distinguishing hotels by their countries), we obtain an impressive increasing of the recall on NREC, from 8.2% to 75.9%, as it is shown in the "RDFS+ (HOTELS) & DA" column.

L2R Results on Cora data set. We focus on the results obtained for the *Article* and *Conference* classes, which contain respectively 1295 references and 1292 references.

As presented in the column named "RDFS+ (Cora)" of the figure 4, the recall obtained on the Cora data set is 50.7%. This can be refined in a recall of 52.7% computed on the REC subset and a recall of 50.6% computed on NREC subset. The set of inferred reconciliations (REC subset) for references to articles is obtained by exploiting the constraint PFI(*Title, Year*) of combined inverse functionality on the properties $Title$ and $Year$. For the conferences, 35.8% of the reconciliations are obtained by exploiting the constraint PFI(*ConfName, ConfYear*) of combined inverse functionality on the attributes $ConfName$ and $ConfYear$, and 64.1% are obtained by propagating the reconciliations of references to articles, using the constraint PF(*Published*) of functionality of the relation $Published$. The set of inferred no reconciliations (NREC subset) are obtained by exploiting the constraint of disjunction between the *Article* and *Conference* classes.

For this data set, the RDFS+ schema can be easily enriched by the declaration that the property *confYear* is discriminant. When this discriminant property is exploited, the recall on the REC subset remains unchanged (52.7%) but the recall on NREC subset grows to 94.9%, as it is shown in the "RDFS+ (Cora) & DP" column. This significant improvement is due to chaining of different rules of reconciliations: the non-reconciliations on references to conferences for which the values of the *confYear* are different entail in turn non-reconciliations of the associated articles by exploiting the constraint PF(*published*).

This recall is comparable to (while a little bit lower than) the recall on the same data set obtained by *supervised* methods like e.g., [1]. The point is that L2R is *not supervised* and guarantees a 100% precision.

6.3 N2R Results

In the following we summarize the results obtained on the HOTELS data set and on the Cora data set by N2R after the application of L2R.

N2R results on HOTELS data set. The results obtained by N2R on the HOTELS data set are given in Figure 5, where the values in the x-axis correspond to values of the reconciliation threshold T_{rec}.

When $T_{rec} = 1$, N2R do not obtain more results than L2R. When T_{rec} is decreased to 0.70 the recall increases of 31 % while the precision remains at 100%. The best results have been obtained at $T_{rec} = 0.55$. For this value, the F-Measure reaches a maximum value of 94 %, with a recall of 98 % and a precision of 91 %. Some mis-reconciliations are due to the fact that some hotel references have the same name and a different addresses and vis versa.

The reconciliation space of N2R has been reduced of 75.9 % which corresponds to 427 212 of reference pairs among 562 368 reference pairs in total.

When N2R is applied independently, the results are very close than those obtained when N2R is combined with L2R.

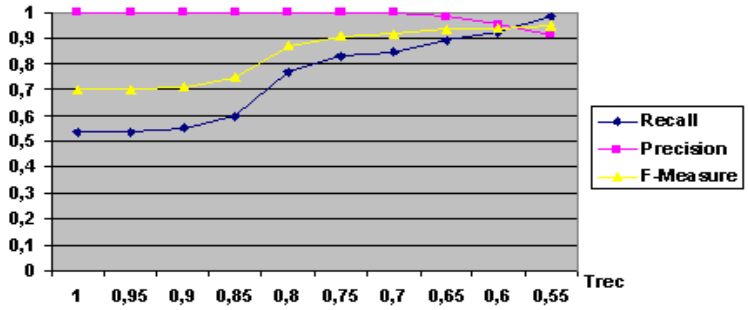

Fig. 5. N2R results obtained on HOTELS data set

N2R results on Cora data set . The results obtained by N2R on the Cora data set are given in the Figure 6.

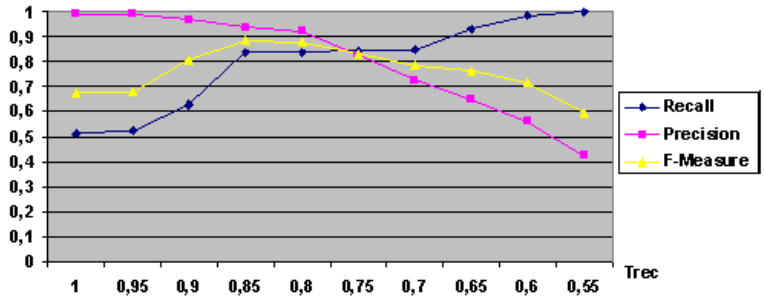

Fig. 6. N2R results obtained on *Cora* data set

For $T_{rec} = 1$, N2R do not obtain more results than L2R. We also emphasize the interesting evolution of the recall and precision values in function of T_{rec}. Indeed, when the threshold is decreased to 0.85, the recall increases by 33% while the precision only falls by 6%. The best results are obtained when $T_{rec} = 0.85$. The F-measure is then at its maximum value of 88%. Besides, when the recall value is almost of 100%, for $T_{rec} = 0.5$, the precision value is still about 40%.

The exploitation of the non-reconciliation inferred by L2R allows an important reduction of the reconciliation space handled in N2R. For the *Cora* data set the size of the reconciliation space is about 37 millions of reference pairs. It has been reduced of 32.8 % thanks to the correct no reconciliations inferred by

L2R. This reduction corresponds to 12 millions of reference pairs. Moreover, the reconciliation inferred by L2R are not recomputed in N2R.

These experimentations show that good results can be obtained by an automatic and unsupervised method if it exploits knowledge declared in the schema. Furthermore, the method is able to obtain F-Measure which is better than some supervised methods such that [14]. This collective record linkage method obtains an F-Measure of 87% for the same data set. Nevertheless, the results obtained by other supervised methods are slightly better than ours: [1] obtain a F-Measure of 90 % by using a method based on a dependency graph where the dependencies between reference pairs are learnt on labeled data ; and [31] obtain a F-Measure of 95 % by using an adaptive approach where the used similarity measures are learnt on labeled data and adapted to the specificities of the data sets. Since, in our numerical method, the similarity computation takes into account the schema semantics, it obtains results that are comparable to those obtained by supervised methods, even without using any labeled data.

When N2R is applied separately, we obtain only a slight regression of N2R results.

6.4 Efficiency results

We have conducted efficiency experiments of the reconciliation methods L2R and N2R on *Cora* data set. We aim by these experiments to show how the efficiency of N2R is improved when the L2R results are exploited. We have applied the reference reconciliation methods on reference sets selected from *Cora* data set ranging from 632 to 6108 references. At each stage we have increased the data set of $\frac{1}{10}$th of the whole set of references.

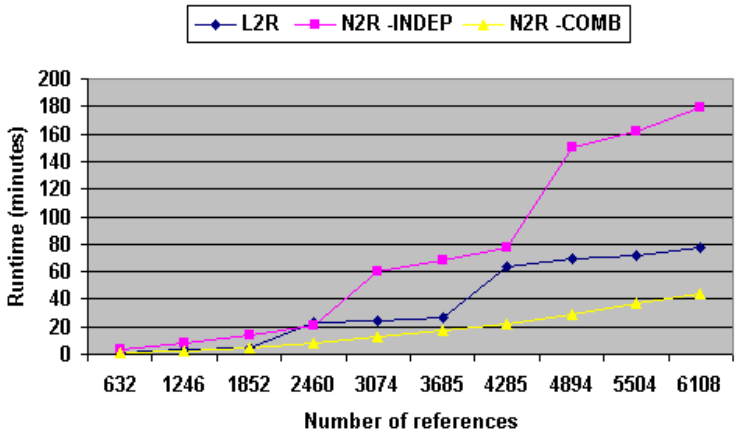

Fig. 7. Execution time of L2R and N2R methods obtained on *Cora* data set

We note that the reference reconciliation code was implemented in Java, and our experiments were run on 2,2GHz Intel Core 2 Duo with 2GB of RAM.

To analyze effenciency, we have measured the execution time of the methods without considering the runtime of the data loading step, because it is common for both methods. Figure 7 shows the execution time obtained for the methods of reference reconciliation L2R and N2R, when they are applied independently and when they are combined, i.e. N2R is preceeded by L2R. For N2R method we have fixed the parameter ϵ at 0.0001.

The first result concerns the case when L2R and N2R are applied independently (curves labeled L2R and N2R-INDEP of the Figure 7. We notice that when the dataset gets larger, L2R outperforms N2R up to 59 % for 3685 references.

The second result concerns the execution time of N2R when it uses the results inferred by L2R. The curves labeled N2R-INDEP and N2R-COMB show a real improvement of N2R runtime when it is preceeded by L2R. The runtime falls of 74% for the 6108 references. This improvement in the N2R runtime is due to the large amount of non-reconciliations inferred by L2R, that are not considered by N2R i.e. they are not added in the equation system. Even the inferred reconciliations between references contribute in the speeding up of N2R. Actually, the similarity scores of these references is fixed at the maximum value then there is no need to compute their similarity scores be N2R. We have also noticed that the convergence of N2R-COMB is achieved in less iterations (3 iterations in average) than when is is applied independently (9 iterations in average). Finally, Figure 7 shows that N2R-COMB outperforms L2R up to 42% for the 6108 references.

7 Related work

The problem of reference reconciliation was introduced by the geneticist Newcombe [32] and was first formalized by Fellegi and Sunter [2]. Since then, various approaches have been proposed in different areas and under different names – record linkage[2, 3], object matching [7], or entity resolution [6, 5]. We distinguish the different approaches : (i) according to the exploitation of the reference descriptions, i.e. if the relations between references are exploited in addition to the attributes ; and (ii) according to how knowledge is acquired, i.e. if knowledge is learnt on labelled data or is declared by domain expert.

The naive way to decide on the reconciliation or on the non-reconciliation of references, is the comparison of their unstructured textual description [33, 34]. In these approaches, the similarity is computed by using only the textual values of the attributes in the form of a single long string without distinguishing which value corresponds to which attribute. This kind of approaches is useful in order to have a fast similarity computation [33], to obtain a set of reference pairs that are candidates for the reconciliation [34] or when the attribute-value associations may be incorrect. That is why this technique is used in *CiteSeer* portal to reconcile data which is automatically extracted from Web pages.

The traditional approaches of reference reconciliation consider the reference description as structured in several attributes. To decide on the reconciliation or on the non-reconciliation of references, some of these approches use probabilistic models [2–4], such as Bayesian network or SVM. However, these probabilistic models need to be trained on labeled data. This training step can be very time-consuming what is not desirable in online applications. Indeed, in such online contexts, labeled data can not be acquired and runtime constraints are very strong. Alternative approaches have been proposed like [35] where the similarity measures (see [29] for a survey) are used to compute smilarity scores between attribute values which are then gathered in a linear combination, like a weighted average. Although these approaches do not need training step, they however need to learn some parameters like weights associated to similarity scores of the different attributes. In order to improve the result quality some methods [11] use adaptive supervised algorithms that learn string similarity measures from labeled data.

The idea of exploiting relations that link references together has been recently explored in several works on reference reconciliation. The relations can be either explicitly expressed in data [12, 1, 14], such as foreign-keys in relational databases, or discovered [13] and then used during the reference reconciliation. To model the dependencies between reference pairs induced by the relations, [1] build a dependency graph and use it to iteratively propagate similarity scores and reconciliation decisions. However, the weights associated to the dependencies are learnt on labeled data. In [13] , the authors translate the *Context Attraction Principle* in a linear equation system and then, by its resolution they compute the connection strength between entities, through relations. In [14], a probabilistic dependency model has been proposed. It allows propagating reconciliation decisions through shared relations. In our approach, the relations between references are exploited by both logical and numerical reference reconciliation methods. The relations are used in the logical method to iteratively propagate reconciliation and non-reconciliation decisions through the logical rules. They are exploited in the numerical method to iteratively propagate similarity scores thanks to the iterative resolution of the non-linear equation system. Furthermore, our logical method infer correct non-reconciliations between references which is very usefull in applications where there are very few redundancies. These correct non-reconciliations can be used by the numerical method to reduce its reconciliation space and therefore speed-up its excecution time.

In order to improve the quality of their results, some recent methods exploit knowledge like the importance of the different attributes and relations, similarity measures or reconciliation and non-reconciliation rules. Knowledge can be either learnt on labeled data or declaratively specified by a domain expert. For instance, in [35, 1], knowledge about the impacts of the different attributes or relations are encoded in weights learnt on labeled data. In [7] the rules of value normalization (i.e. date format, phone number) and of reconciliation are learnt on labeled data by using a decision tree model. In the declarative approach proposed by [36], profiles of the representative entities are expoited. These profiles that are

manually specified by a domain expert contain a set of constraints on correlations between attributes which should be satisfied by the references.

In the system AJAX [10] a declarative language has been proposed to express different kinds of knowledge like knowledge on value normalization, on mapping operations and on the extraction of groups of references. Although these supervised and declarative methods ensure good results, they remain however vulnerable to changes of application domain and of data sources features. The supervised method should re-learn the knowledge on new labeled data and for the declarative methods the expert should be asked to re-specify the used knowledge. In the spirit of knowledge-based approches, we propose two declarative methods which exploit general knowledge declared on the schema and on the data sources, such as functional dependencies and Unique Name Assumption. In L2R, knowledge semantics is automatically translated into Horn rules and used to infer (non) reconciliations between reference pairs and (non) synonymies between values. Knowledge semantics is also automatically translated into non-linear equations which allow to compute similarity scores of reference pairs. Comparing to the viewed knowledge-based approches, our methods are not sensitive to domain and data changes. The exploited knowledge are general and domain-independent, indeed. Futhermore, the logical and numerical methods are unsupervised since no labeled data is needed by neither L2R nor N2R.

8 Conclusion and future work

We have presented the combination of a logical and numerical approach for the reference reconciliation problem. Both approaches exploit schema and data knowledge given in a declarative way by a set of axioms. This guarantees their genericity: if the domain or the sources change it is sufficient to update the set of axioms. Secondly, the relations between references are exploited either by L2R for propagating (non) reconciliation decisions through logical rules or by N2R for propagating similarity scores thanks to the resolution of the equation system. Third, the two methods are unsupervised because no labeled data set is used. Fourth, the combined approach is able to capitalize its experience by saving the correct (no) synonymies inferred by L2R in a dictionary. This allows to learn the syntactic variations of an application domain.

Furthermore, by using the logical method we obtain reconciliations and non-reconciliations that are sure. This distinguishes L2R from other existing works. This is an important point since, as it has been emphasized in [3], unsupervised approaches which deal with the reference reconciliation problem have a lot of difficulties to estimate in advance the precision of their system when it is applied to a new set of data. The numerical method complements the results of logical one. It exploits the schema and data knowledge and expresses the similarity computation in non linear equation system. This distinguishes N2R from other existing work.

The experiments show promising results for recall, and most importantly its significant increasing when axioms are added. This shows the interest and the

power of the generic and flexible approach of L2R since it is quite easy to add rules to express constraints on the domain of interest.

As a future work, we first plan to exploit the results of the logical step to learn the weighting coefficients involved in the combination of the different similarity scores. We also plan to adapt the method to be used in a peer-to-peer settings. A system such SomeWhere [37] is a P2P infrastructure that exploits mappings between peer's ontologies to answer queries in a sound and complete way. This could be completed by a reference reconciliation method based on L2R and N2R to discover and exploit reconciliation decisions between references stored at different peers. Finally, we plan to study how the reference reconciliation could help the schema reconciliation and conversely how the schema reconciliation could help the reference reconciliation.

References

1. Xin Dong, Alon Halevy, and Jayant Madhavan. Reference reconciliation in complex information spaces. In *ACM SIGMOD*, pages 85–96. ACM Press, 2005.
2. Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.
3. William E Winkler. Overview of record linkage and current research directions. Technical report, Statistical Research Division U.S. Census Bureau Washington, DC 20233, 2006.
4. Vassilios S. Verykios, Ahmed K. Elmagarmid, and Elias N. Houstis. Automating the approximate record-matching process. *Inf. Sci. Inf. Comput. Sci.*, 126(1-4):83–98, 2000.
5. Omar Benjelloun, Hector Garcia-Molina, Hideki Kawai, Tait Eliott Larson, David Menestrina, Qi Su, Sutthipong Thavisomboon, and Jennifer Widom. Generic entity resolution in the serf project. *IEEE Data Eng. Bull.*, 29(2):13–20, 2006.
6. Indrajit Bhattacharya and Lise Getoor. *Entity Resolution in Graphs*, chapter Entity Resolution in Graphs. Wiley, 2006.
7. Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.
8. Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10:334–350, 2001.
9. Pavel Shvaiko and Jerome Euzenat. A survey of schema-based matching approaches. *Journal on Data semantics*, 2005.
10. Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, and Cristian Saita. Declarative data cleaning: Language, model and algorithms. In *VLDB '01*, 2001.
11. Misha Bilenko and Ray Mooney. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD'03*, 2003.
12. Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. Eliminating fuzzy duplicates in data warehouses. In *ACM SIGMOD*. ACM Press, 2002.
13. Dmitri Kalashnikov, Sharad Mehrotra, and Zhaoki Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM Data Mining '05*, 2005.
14. Singla Parag and Domingos Pedro. Multi-relational record linkage. In *MRDM Workshop*, 2004.
15. Brian McBride. The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS. In *Handbook on Ontologies*, pages 51–66. 2004.

16. Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview, February 2004.
17. Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML (a submission), May 2004.
18. Jesús Barrasa Rodriguez and Asunción Gómez-Pérez. Upgrading relational legacy data to the semantic web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 1069–1070. ACM, 2006.
19. Chuck Murray, Nicole Alexander, Souri Das, George Eadon, and Siva Ravada. Oracle spatial resource description framework (rdf). Technical report, Oracle., 2005.
20. G. H. Golub and C. F. Van Loan. *Matrix Computations*. Baltimore, MD, USA, second edition, 1989.
21. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
22. P. Hayes. RDF Semantics, http://www.w3.org/tr/rdf-mt/. Technical report, 2004.
23. Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. L2R: A Logical Method for Reference Reconciliation. In *AAAI*, pages 329–334, 2007.
24. Susan Fischer. Two processes of reduplication in the american sign language. *Foundations of Language*, 9:460–480, 1973.
25. Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal ACM*, 12(1):23–41, 1965.
26. Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Inc., Orlando, FL, USA, 1997.
27. Lawrence J. Henschen and Larry Wos. Unit refutations and horn sets. *J. ACM*, 21(4):590–605, 1974.
28. Kenneth D. Forbus and Johan de Kleer. *Building problem solvers*. MIT Press, Cambridge, MA, USA, 1993.
29. William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, pages 73–78, 2003.
30. Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in owl-lite. In *ECAI*, pages 333–337, 2004.
31. William W. Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD '02*, pages 475–480, NY, USA, 2002. ACM.
32. H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records. *Science*, 130:954–959, October 1959.
33. William W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18(3):288–321, 2000.
34. Alexander Bilke and Felix Naumann. Schema matching using duplicates. In *ICDE*, pages 69–80, 2005.
35. Debabrata Dey, Sumit Sarkar, and Prabuddha De. A probabilistic decision model for entity matching in heterogeneous databases. *Manage. Sci.*, 44(10):1379–1395, 1998.
36. AnHai Doan, Ying Lu, Yoonkyong Lee, and Jiawei Han. Object matching for information integration: A profiler-based approach. In *IIWeb*, pages 53–58, 2003.
37. Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *J. Artif. Intell. Res. (JAIR)*, 25:269–314, 2006.