# Sharing Co-reference Knowledge for Data Integration

Carlo Meghini[1], Martin Dörr[2], and Nicolas Spyratos[3]

[1] Istituto della Scienza e delle Tecnologie della Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy
[2] Institute of Computer Science Foundation for Research and Technology – Hellas Crete, Greece
[3] Université Paris-Sud, Laboratoire de Recherche en Informatique, Orsay Cedex, France

**Abstract.** Data integration requires solving the basic problem of determining whether two identifiers from different terminologies co-refer, that is refer to the same object. The problem arises in many different contexts and provokes intense philosophical debate, yet it has never been addressed in a systematic way from an information system point of view. This paper presents a formal model of co-reference, as a relation which is part of the reality created by language structures. A complete mathematical characterization of co-reference is derived. A service handling co-reference knowledge is then introduced, and the basic functionality of such a service is formally defined in terms of a certain mathematical structure. Finally, the implementation of the service is studied from the computational point of view, and efficient algorithms for carrying out the basic operations of the service are provided.

## 1 Introduction

Information integration is traditionally done by reconciling data coming from different sources under a common schema. This is a well-known approach in heterogeneous database integration, which recently also involves the use of formal ontologies. Integration of data under a common schema allows for accessing information about the same kinds of things and same kinds of relations as defined by the integrated schema.

Our work relies on a different approach to integration. Namely, rather than bringing data under a common schema, we let the sources operate in their own independent ways, and simply connect them into a coherent network of knowledge. The two basic requirements for such a network to be created and operate are the following: (1) Decide whether two data elements residing in different sources refer to the same thing. [13] describes this relationship between data elements as the co-reference relation. (2) Maintain the co-reference relation and make it accessible to the individual sources participating in the network. We note that co-reference between two data elements is independent of their nature or whether they belong to the same source. Actually, the problem exists even if

the data elements are not classified under any schema. For instance, historical documents fall under no particular schema whatsoever and yet establishing co-reference between text elements in two documents is fundamental for historians in order to reconstruct history.

Indeed, if we regard the instantiation relation between data elements and respective schema elements they belong to as yet another association of data elements, schema integration can also be regarded as the. co-reference problem to match the referred schema elements. Therefore co-reference recognition may even be regarded as the most elementary operation of information integration.

Curiously enough, the problem of establishing and maintaining a co-reference relation has been addressed so far only in some symptomatic ways, as described in more detail in Section 2. Typically, after integrating databases on the schema level, various heuristics are employed to remove "duplicate" entries coming from different sources and referring to the same thing. On the other hand, librarians invest a lot of time to create large name spaces, such as thesauri, "authority files" of person names, or gazetteers of place-names, in the hope that people will use the names in these spaces as identifiers in their local sources. These methods suffer either from precision or scalability, and there is no widely accepted concept of long-term preservation of the knowledge about co-reference itself.

This paper deals with a more fundamental theory, which aims at providing a base for generalizing over current methods and for making effective algorithms to manage co-reference information and factual integration in a scalable way. We base this theory on the distinction between symbols on the one hand, and things referred to by those symbols on the other hand [6]. As usual, the association between symbols and objects is done by an interpretation function, or *meaning*. We assume that this function is private to each information source. Moreover, we assume that each source has initially no knowledge of the symbols of any other source. Finally, we assume that any two information sources have the ability to negotiate in order to decide co-reference, however we do not model the decision process. We are only interested in whether or not an agreement is reached. The idea is that the agreed-on co-reference information is *preserved* and can be used to collaboratively build large-scale networks of knowledge. Our theory is motivated by situations that occur between research or business groups maintaining huge, consolidated information records, with billions of identifiers for things, people, places etc. Typically, we distinguish between the state of reality, the states of private knowledge and the state of shared knowledge, and we analyze questions of consistency between reality and the different units of knowledge maintained by different sources.

## 2 Related work

The current approaches to the co-reference problem can roughly be divided into *pro-active* and *reactive* methods. In reactive methods, typically, data are first integrated under a common schema and then possible *duplicate* data entries are detected and merged. *Duplicate detection* methods (e.g., [3]), sometimes

also found under the more general term *data cleaning* methods, employ various heuristics to compare the attributes of items in different data sources, and estimate the probability that they mean actually the same thing. Unfortunately, different sources tend to register different properties for the same things, properties of items may change over time or actually be unknown, making the method quite unreliable. Furthermore, data cleaning requires *pre-existing shared* knowledge about the values of the properties being compared, and it is unclear how such knowledge can be obtained. This is why, in our opinion, theoretical foundations are needed. In *pro-active* methods, identifiers of things are normalized before integration takes place, in order to increase the chance that other sources will normalize their identifiers in precisely the same way. One way to do that is by using rules, such as the cataloging rules librarians use to encode identifiers for books or authors–the most important system being AACR [2]. Rules do not help in cases when people use pseudonyms, when books do not expose standard front pages, when historical places are known under multiple names, etc. They are more helpful to avoid false matches. Even when applying rules, encoding errors may cause identifier variants. Therefore library organizations such as OCLC (Ohio, US) offer data cleaning services that validate entries sent by libraries against a huge database of entries they know of. This method has the advantages and disadvantages of *authority files*. In a way, it is a duplicate detection process between the source to be "cleaned" and the reference source. A variant of the pro-active methods is the *referent tracking* suggested by [20]. The idea is, that in controlled environments, such as health-care institutions, things like patients, body-parts, blood and tissue samples are multiply referred to in different documentation systems accompanying diagnostic and therapeutic processes. The authors suggest methods and a good practice to make sure that the notion of identity is not lost between the different documents pertaining to related processes.

Librarians, scholars and scientists from many disciplines have been heavily investing in so-called authority files or better knowledge organization systems (KOS) [16], which register names and characteristics of authors [10], historical persons [9], places ([8], [7]), books and other items, as well as categories (Library of Congress Subject Headings [5], the Art & Architecture Thesaurus [18], Unified Medical Language System UMLS [19], etc.), and associate them with a preferred, unique representation in a central resource. Preferably, the KOS should list enough properties of the described items so that a user may match this description with another description under her control, and use the KOS suggested preferred identifier as a unique identifier in her source. The method is successful as long as the preferred identifiers are globally unique, correctly applied, and the KOS does not change the representation later. It is more reliable than automated data cleaning. Unfortunately there are many KOS in use, and few maintainers, notably the projects VIAF [17], LEAF [10] for person authorities, and MACS [12] for subject headings, have understood the relevance of co-reference relations between KOS. In order to find all references made via a preferred identifier from a KOS, still a global index similar to current Web search

engines would be necessary. Unfortunately, there are no efforts to do so. KOS are also used to support data mining, i.e. the so-called Named Entity Recognition [4]. They help to decide if a word in a text is the name of a place, person, etc.

The major draw-back of KOS-based methods is that a central authority is assumed, which makes the method non-scalable, always lagging behind application. There are *far more* particulars (persons, objects, places, events) than categories (universals), making the lack of scalability very severe for particulars. Whereas the Semantic Web research is strongly focused on matching or mapping universals (e.g. [11]), here we address situations that a priori apply to particulars. We describe configurations that imply the connection of multiple KOS by co-reference links, and thereby implicitly show that a central authority and exhaustive description of the item is not necessary, as useful it might be. Actually in [15], the authors recognize the fact that standardization and centralization is not necessary for co-reference, but they still assume that there must be a unique digital surrogate for a real world item to ensure unique meaning, whereas we show that even that is not necessary. Only a strongly distributed approach has the potential to deal with co-reference of particulars.

[14] describes a statistical information retrieval method, in which related documents are traced via characteristic co-occurrences of terms common to documents. In a way, co-occurrence of terms can be regarded as signature of relations. Hence the method may be regarded to pertain to co-reference of relations.

Common to all current methods is that they provide partial, isolated solutions. In contrast, in this paper, we present a theoretical framework that treats co-reference semantics in a unified manner. We assume a community of collaborating agents, whose communications relies on co-reference relations. These relations are built on the basis of bilateral agreements, and maintained in a decentralized manner, similarly to what is postulated in "emergent semantics" [1]. They are consolidated into an ever-growing, coherent system of meaning by virtue of a few global rules.

## 3 Language structures and co-reference relations

We assume a countable, non-empty set of objects $Obj$ and a collection $\mathcal{A}$ of $n > 2$ agents $A_1, \ldots, A_n$ who speak about $Obj$. An agent may represent a whole community of speakers that share some knowledge about $Obj$. For example, an agent might represent the community of users of some source that stores information about $Obj$. For the purposes of this paper, however, we do not need to make the distinction between users of the source and the agents representing them. We assume that each agent $A_i$ is endowed with:

- a *vocabulary* $V_i$ that is a non-empty, countable set of identifiers which the agent uses to refer to (or denote) objects in $Obj$; we will use $i, i', i_1, \ldots$ to denote the identifiers of $V_i$;
- a *reference function* $R_i$ associating each identifier in $V_i$ with an object in $Obj$.

| $Obj = \{1, 2, 3\}$ | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| $V_1 = \{i_1, i_2\}$ | $i_1$ 1 | $j_1$ 1 | $k_1$ 1 |
| $V_2 = \{j_1, j_2\}$ | $i_2$ 2 | $j_2$ 3 | $k_2$ 2 |
| $V_3 = \{k_1, k_2, k_3\}$ | | | $k_3$ 3 |

**Fig. 1.** Elements of a language structure

Without loss of generality we can think of an identifier as a kind of URI (Universal Resource Identifier), independent from who knows its meaning. The only requirement is that its encoded form is different from the encoded form of any other identifier. In practice, this can be achieved by a mechanism adding a name-scope identifier to the encoded form of each local identifier. For example, `French:diffusion` is regarded as distinct from `English:diffusion`. The successful worldwide management of domain names shows that there is no reason to question the feasibility of assigning unique domain names (*i.e.*, unique agent names). We make the following assumptions:

**Assumption 1**
The vocabularies $V_1, \ldots, V_n$ are all distinct and pairwise disjoint; we will let $\mathcal{V}$ stand for the collection of such vocabularies (*i.e.*, $\mathcal{V} = \{V_1, \ldots, V_n\}$);

**Assumption 2**
Each reference function $R_i$ is:
- *total,* that is each identifier of vocabulary $V_i$ denotes some object, for $i = 1, \ldots, n$;
- *injective,* that is no two identifiers from the same vocabulary denote the same object;
- *private,* that is accessible *only* to agent $A_i$.

We will let $\mathcal{R}$ stand for the collection of reference functions (*i.e.*, $\mathcal{R} = \{R_1, \ldots, R_n\}$);

**Assumption 3**
Every object is denoted by at least one identifier of some vocabulary; in other words, there is no object unknown to all agents.

A *language structure* $\lambda$ is a 4-tuple $\lambda = \langle Obj, \mathcal{A}, \mathcal{V}, \mathcal{R} \rangle$ satisfying all the above assumptions. The *vocabulary* of $\lambda$ is denoted by $\mathcal{L}_\lambda$ (or simply by $\mathcal{L}$ when no ambiguity may arise) and it is defined as follows:

$$\mathcal{L} = \bigcup \mathcal{V} = V_1 \cup \ldots \cup V_n$$

Figure 1 shows a language structure, consisting of three agents $A_1$ to $A_3$ which speak about a domain of three objects, represented as the first three positive integers. Throughout the paper, we will use this language structure as our running example.

In a language structure $\lambda$, it may happen that two identifiers from different vocabularies, say $x \in V_i$ and $y \in V_j$, refer to the same object that is $R_i(x) = R_j(y)$. In this case, we say that $x$ and $y$ *co-refer.* In formal terms, this defines a relation $\approx_\lambda$ (or simply $\approx$) over identifiers as follows:

$$x \approx y \text{ iff } R_i(x) = R_j(y).$$

We shall refer to this relation as the *co-reference* relation of the language structure. It is easy to see that the co-reference relation is an equivalence relation, as it satisfies the following properties:

- it is reflexive, since $R_i(x) = R_i(x)$, for all vocabularies $V_i$ and identifiers $x \in V_i$;
- it is symmetric, since $R_i(x) = R_j(y)$ implies $R_j(y) = R_i(x)$, for all vocabularies $V_i, V_j$ and identifiers $x \in V_i$ and $y \in V_j$;
- it is transitive, since $R_i(x) = R_j(y)$ and $R_j(y) = R_k(z)$ imply $R_i(x) = R_k(z)$, for all vocabularies $V_i, V_j, V_k$ and identifiers $x \in V_i$, $y \in V_j$ and $z \in V_k$.

As an equivalence relation, co-reference induces a partition on $\mathcal{L}$, given by:

$$[\mathcal{L}] = \{\ [i] \mid i \in \mathcal{L}\}.$$

Each block $[i]$ of $[\mathcal{L}]$ consists of the identifiers denoting the same object as $i$. It follows from Assumption 3, that there exists a bi-jection between $Obj$ and $[\mathcal{L}]$, associating every object $o \in Obj$ with the block of the identifiers denoting $o$. This association captures *naming,* therefore we will call the block $[i]$ of identifiers denoting the object $o \in Obj$ as the *names* of $o$; in the opposite direction, we will say that $o$ *is named by* $[i]$. As a consequence of Assumption 2 (injectivity of reference functions), the names of an object come *each* from a different vocabulary. Formally, for all vocabularies $V_i$ and identifiers $x_1, x_2 \in V_i$,

$$x_1 \neq x_2 \text{ implies } [x_1] \neq [x_2] \tag{1}$$

In proof, $[x_1] = [x_2]$ implies $R_i(x_1) = R_i(x_2)$ and therefore $x_1 = x_2$. Consequently, each block has at most as many identifiers as vocabularies: $|[x]| \leq |\mathcal{V}|$. If a vocabulary $V_j$ has no identifier in the block $[x]$, then $V_j$ has no name for the object named by $[x]$. Another way of saying this, is to say that if $x$ and $y$ are co-referring identifiers from different vocabularies $V_i$ and $V_j$, then $x$ does not co-refer with any other identifier in $V_j$:

$$x \in V_i,\ y, y' \in V_j,\ V_j \neq V_i,\ x \approx y,\ y' \neq y \text{ imply } x \not\approx y' \tag{2}$$

(1) and (2) are easily seen to be equivalent, but the latter highlights the fact that co-reference has implicit negative facts, in addition to the positive ones.

As a last remark, it is easy to verify that the complement of co-reference (with respect to $\mathcal{V} \times \mathcal{V}$), $\not\approx$ is irreflexive and symmetric.

Figure 2 shows the co-reference relation and its equivalence classes in our running example. To simplify the presentation, we only show the a reduced $\approx$ (*i.e.*, we do not show reflexive, symmetric or transitive pairs), denoted as $\approx^r$ . For clarity, equivalence classes are associated with the object they name.

## 4 Sharing co-reference knowledge

We view the language structure $\lambda$ as being the "reality" under study. Initially, each agent $A_i$ knows part of the reality, namely his own vocabulary $V_i$ and

| $\approx^r$ | |
|---|---|
| $i_1$ $j_1$ | $1 \longleftrightarrow \{i_1, j_1, k_1\}$ |
| $j_1$ $k_1$ | $2 \longleftrightarrow \{i_2, k_2\}$ |
| $i_2$ $k_2$ | $3 \longleftrightarrow \{j_2, k_3\}$ |
| $j_2$ $k_3$ | |

**Fig. 2.** A (reduced) co-reference relation and its equivalence classes

reference function $R_i$. The knowledge of the agent increases when he engages in communication with another agent $A_j$. This communication can be direct when both the agents are present, or indirect if it happens through the exchange of documents. During communication, agent $A_i$ uses an identifier, say $x$, in his own vocabulary $V_i$ to refer to an object in $Obj$. But this creates a problem, because $A_j$ must be able to *de-reference* $x$, that is to determine what is the object under discussion (*i.e.*, the object $R_i(x)$). However, $R_i$ is only accessible to $A_i$, thus $A_j$ is left with an undoable task.

In order to overcome this problem, we envisage a service, called *co-reference knowledge service,* to which $A_j$ can *ask* which identifiers are known to co-refer with $x$. If, amongst the returned identifiers, $A_j$ finds one in his own language, say $y$, then $A_j$ is able to identify the referenced object by applying to $y$ his reference function $R_j$. If no identifier in $V_j$ is returned by the service, then the object named by $x$ is unknown to agent $A_j$, and no de-reference is possible. In this case, $A_j$ may ask the identifiers which are known not to co-refer with $x$, thus being able to determine which objects are *not* named by $x$.

The question arises how the co-reference knowledge service can acquire the knowledge it is asked about. The answer is that the service *is told* this knowledge by agents, as a result of *negotiations.* A negotiation involves two agents $A_i$ and $A_j$ and two identifiers in their respective vocabularies, say $x \in V_i$ and $y \in V_j$, and aims at ascertaining whether $x$ and $y$ co-refer. A negotiation may have one of the following outcomes:

- The agents agree that $x$ and $y$ co-refer.
- The agents agree that $x$ and $y$ do not co-refer.
- The agents are not able to reach an agreement.

In the first two cases, agents tell the service the found relationship between identifiers, thus increasing the global knowledge. In the latter case, no knowledge is gained, thus nothing is told to the service. It is important to notice that negotiations are supposed to take place *externally* to the service, which gives no support to their making. Rather, the service manages the outcomes of negations, whether positive or negative, in order to make communication successful according to the schema outlined above.

In this schema, the co-reference knowledge service must support the following operations:

- `tell-co`$(i, j)$, by means of which the user tells the service that $i$ and $j$ are known to co-refer;

| *co* |
|---|
| $i_1$ $j_1$ |
| $j_1$ $k_1$ |

| *nco* |
|---|
| $j_2$ $k_2$ |

**Fig. 3.** Tables of a co-reference structure

- `tell-nco`$(i, j)$, by means of which the user tells the service that $i$ and $j$ are known not to co-refer;
- `ask-co`$(i)$, for asking the service the identifiers that are known to co-refer with $i$;
- `ask-nco`$(i)$, for asking the identifiers that are known not to co-refer with $i$.

In addition, we include operations for retracting co-reference knowledge, which are necessary due to the possible incorrectness of agents in negotiations:

- `untell-co`$(i, j)$, for retracting that $i$ is known to co-refer with $j$;
- `untell-nco`$(i, j)$, for retracting that $i$ is known not to co-refer with $j$;

In order to be able to perform these operations, the service relies on a *co-reference knowledge structure* $\mathcal{K}$ which we define as a triple $\mathcal{K} = \langle \lambda, co, nco \rangle$ where:

- $\lambda$ is a language structure;
- *co* is the *co-reference table,* a binary relation over $\mathcal{L}$ storing the pairs that are found to co-refer;
- *nco* is the *non co-reference table,* a binary relation over $\mathcal{L}$ storing the pairs that are found not to co-refer.

The co-reference tables for our running example are shown in Figure 3. We assume that a co-reference structure is accessible by all agents.

In the following, we illustrate how a co-reference structure can be used to formally specify a semantics for the operations defined above.

### 4.1 Semantics of `ask` operations

Co-reference tables hold the *explicit* knowledge agreed upon by the agents during negotiations. However, there is also *implicit* knowledge; in our example, the fact that $i_1$ co-refers with $j_1$ and $j_1$ with $k_1$ is explicit knowledge, from which we can infer by the transitivity of co-reference (see Section 3) that $i_1$ and $k_1$ co-refer, or by the symmetry of co-reference that $j_1$ and $i_1$ co-refer; and maybe other. All this is implicit co-reference knowledge, which the user expects the service to be able to discover and serve in response to `ask` operations. Therefore, in defining the semantics of `ask` operations, we must take into account what is implicitly known from the explicit facts stored in a co-reference structure.

To this end, we first observe that a co-reference structure can be legitimately said to be such, only if it exhibits the basic properties of co-reference highlighted in Section 3. In order to capture this requirement, we introduce *co-reference models:* A co-reference structure $\mathcal{K} = \langle \lambda, co, nco \rangle$ is a *co-reference model* (or simply *model* for short) iff it satisfies the following conditions:

**(c1)** $co$ is an equivalence relation;
**(c2)** for every pair $(i,j) \in co$ such that $i \in V_i$, $j \in V_j$ and $V_i \neq V_j$, there is a set of pairs $(i,j') \in nco$, where $j' \in V_j$ and $j \neq j'$;
**(c3)** $nco$ is symmetric;
**(c4)** $co$ and $nco$ are disjoint.

Conditions (c1)-(c3) simply state the properties which characterize co-reference and non-co-reference, while (c4) adds the requirement that $co$ and $nco$ be disjoint. The next step is to apply these properties to a co-reference structure $\mathcal{K}$, thus completing the explicit knowledge in $\mathcal{K}$ with the underlying implicit knowledge. The closure operation does precisely this. The *closure* of a co-reference structure $\mathcal{K} = \langle \lambda, co, nco \rangle$, is a co-reference structure $\mathcal{K}^\star = \langle \lambda, co^\star, nco^\star \rangle$, where:

- $co^\star$ is the smallest equivalence relation containing $co$, thus satisfies condition (c1) above. $co^\star$ can be formally defined as follows. For any set of pairs $X$, we let:

$$\begin{aligned}
\rho(X) &= \{(x,x) \mid (\exists y)(x,y) \in X \vee (y,x) \in X\} \\
\sigma(X) &= \{(x,y) \mid (y,x) \in X\} \\
\tau(X) &= \{(x,y) \mid (\exists z)(x,z) \in X \wedge (z,y) \in \tau(X)\} \\
f(X) &= \rho(X) \cup \sigma(X) \cup \tau(X)
\end{aligned}$$

  The first three functions realize, respectively, the reflexive, symmetric and transitive closure of the given argument (the specification of the last function can be made more explicit, but we omit these details for brevity). Finally, $f$ includes the results of these functions by taking their union. Now, for a co-reference structure $\mathcal{K} = \langle \lambda, co, nco \rangle$, define the *domain* of $\mathcal{K}$ to be the set $\mathcal{D}_\mathcal{K} = \{co \cup A \mid A \subseteq (\mathcal{L} \times \mathcal{L})\}$. It can be easily verified that $(\mathcal{D}_\mathcal{K}, \subseteq)$ is a complete lattice having $co$ as least element. Since $X \subseteq \tau(X)$, $f$ is monotonic (hence continuous) on this lattice. Thus, by the Knaster-Tarski theorem, $co^\star = f^n(co)$, for $n$ finite. As a consequence, $co^\star$ exists, is unique and finite, since at each application of the $f$ function only a finite number of pairs are added.
- $nco^\star = X \cup \sigma(X)$ where

$$X = nco \cup \{(i,j) \in V_i \times V_j \mid (i,j') \in co^\star, V_i \neq V_j, j' \in V_j \text{ and } j \neq j'\}$$

  Existence, uniqueness and finiteness of $nco^\star$ immediately follow from those of $co^\star$.

Resuming our example, the pair $(j_1, i_1)$ ends up in $co^\star$ because it is symmetric to the $co$ pair $(i_1, j_1)$. The same does the pair $(k_1, i_1)$, as it is symmetric to the pair $(i_1, k_1)$ which can be obtained by transitivity from $co$.

On the other hand, the closure of the non-co-reference table adds to $nco$ the pairs required for satisfying condition (c2) (yielding $X$), then closing the result under symmetry in order to satisfy also condition (c3). In our example, the pair $k_2$ is implicitly known not to co-refer with $i_1$ because $(i_1, k_1)$ are known

to co-refer, thus $i_1$ cannot co-refer with any other identifier in $V_k$; assuming it is known that $k_2$ is an identifier in $V_k$, this means that $i_1$ and $k_2$ are known not to co-refer; by the symmetry of *nco*, we then obtain that $k_2$ and $i_1$ are known not co-refer.

The closure of a co-reference structure $\mathcal{K}$ embodies the explicit co-reference knowledge in $\mathcal{K}$ and, being an equivalence relation, exhibits the behavior of co-reference. Moreover, being the smallest structure satisfying these two properties, it is a most natural candidate for query answering on $\mathcal{K}$. We therefore define the semantics of `ask` operations by viewing them as functions associating a co-reference structure with sets of identifiers, as follows:

$$\texttt{ask-co}(i)(\mathcal{K}) = \{j \in \mathcal{L} \mid (i,j) \in co^\star\}$$
$$\texttt{ask-nco}(i)(\mathcal{K}) = \{j \in \mathcal{L} \mid (i,j) \in nco^\star\}$$

We simplify notation by writing $\texttt{ask-co}(i, \mathcal{K})$ instead of $\texttt{ask-co}(i)(\mathcal{K})$, and the same for `ask-nco.`

**Inconsistent co-reference structures** It is important to notice that in closing a co-reference structure $\mathcal{K}$, (c4) may be violated, so that the resulting $\mathcal{K}^\star$ is not a model. In this case, some pair $(i, j)$ is known both to co-refer and not to co-refer. This is clearly a contradiction, thus we define $\mathcal{K}$ a *consistent* co-reference structure iff its closure $\mathcal{K}^\star$ is a model. Accordingly, we define the `cons-ch` operation as a function returning the pairs causing inconsistency:

$$\texttt{cons-ch}(\mathcal{K}) = co^\star \cap nco^\star$$

If `cons-ch` returns the empty set, then the current co-reference structure is consistent. Otherwise, the user knows which pairs are causing trouble and can intervene as described later.

In our example, if we add the pair $(i_1, j_1)$ to the *nco* table in Figure 3, we have an explicitly inconsistent co-reference structure, since the same pair shows up in the *co* table. If we add the pair $(k_2, i_1)$ to the *co* table, we have an implicitly inconsistent co-reference structure, because the same pair shows up in the closed non-co-reference table $nco^\star$, as shown in a previous example.

Inconsistent co-reference structures arise from negotiations whose outcome does not correctly reflect the language structure $\lambda$ and, ultimately, from *unsound* agents, that is agents that may make mistakes in negotiations. The assumption that only negotiation mistakes can cause inconsistency of the co-reference structure holds, as long as the agents share an unambiguous principle of identity about the described objects, and they maintain their reference functions injective. As it turns out, this is not an unrealistic case. It appears that inconsistency of the co-reference structure is the only diagnostic we have at information system level of errors in the negotiations. If someone finds out with whatever means that two identifiers do not co-refer, this information enters the *nco* table, which may cause an inconsistency. Any consistent arrangement of false negotiations remains undetected.

### 4.2 Semantics of `tell` operations

The semantics of `tell` operations establish how co-reference structures evolve. Redundancy of explicit knowledge is not helpful as long as believe values are not taken into account. Therefore, an obvious requirement is minimality, in the sense that a co-reference structure should stay as simple as possible, while embodying the knowledge that it has been told. However, manipulating the knowledge inserted by the user is not a good idea, because it may cause loss of knowledge. As an illustration, let us consider the co-reference structure shown in Figure 1. Assume that the user wants to add the knowledge that $i_1$ and $k_1$ co-refer via the operation `tell-co`$(i_1, k_1)$. Now this knowledge is implicit in the co-reference structure, as it can be obtained by transitivity from $(i_1, j_1)$ and $(j_1, k_1)$. We could therefore be tempted to do nothing in response to the above `tell-co.` But if we did so, a successive `untell-co`$(i_1, j_1)$ would cause the loss of the knowledge that $i_1$ and $k_1$ co-refer. We therefore give `tell` operations the following, straightforward semantics, where $\mathcal{K} = \langle \lambda, co, nco \rangle$ is the current co-reference structure:

$$\texttt{tell-co}(i, j, \mathcal{K}) = \langle \lambda, co \cup \{(i, j)\}, nco \rangle$$
$$\texttt{tell-nco}(i, j, \mathcal{K}) = \langle \lambda, co, nco \cup \{(i, j)\} \rangle$$

Note that the addition of knowledge may cause an inconsistency in the co-reference structure. However, there is no guarantee that the piece of knowledge being added reflects an incorrect co-reference relationship. In other words, it might be the case that the inconsistency is caused by knowledge which has been told previously. Thus a `tell` always results in a larger co-reference structure. Clearly, good practice suggests a consistency check after each knowledge insertion. Analogously:

$$\texttt{untell-co}(i, j, \mathcal{K}) = \langle \lambda, co \setminus \{(i, j)\}, nco \rangle$$
$$\texttt{untell-nco}(i, j, \mathcal{K}) = \langle \lambda, co, nco \setminus \{(i, j)\} \rangle$$

Notice that if the pair $(i, j)$ is not present in $co$ ($nco$, respectively), then the former (latter) operation has no effect.

## 5 Implementation

We now discuss the implementation of the operations introduced so far. We begin by introducing the basic data structure for the implementation of co-reference, the co-reference graph.

Given a co-reference structure $\mathcal{K} = \langle \lambda, co, nco \rangle$, the *co-reference graph* of $\mathcal{K}$, $G_{\mathcal{K}}$ (simply $G$ when no ambiguity may arise), is the undirected graph $G = (\mathcal{L}, (co \cup nco))$. Arcs arising from pairs of identifiers in $co$ will be called *co-arcs*, while arcs arising from pairs in $nco$ will be called *nco-arcs*. Likewise, a *co-path* is a path in $G$ including only co-reference arcs. Finally, an identifier $i$ is *co-reachable* from an identifier $j$ if there exists a co-path from $i$ to $j$. Figure 4 shows the

co-graph for the co-reference structure of the running example. For readability, nco-arcs are shown as dashed lines.

A basic property of undirected graphs that will be very useful for the sequel, is that all the nodes from the same component of such a graph are reachable from each other. Since co-reachability will play a crucial role, we focus in particular on the components of the sub-graph $(\mathcal{L}, co)$. We denote these components as $(N_i, E_i)$, for $1 \leq i \leq m$. Each component $(N_i, E_i)$ is a connected, undirected graph. It follows that two identifiers are co-reachable from each other iff they belong to the same set $N_j$, for some $1 \leq j \leq m$. Moreover, by construction the sets $N_i$'s are a partition of the set of identifiers showing up in $\mathcal{K}$, and when complete co-reference knowledge is reached, they coincide with the equivalence classes discussed in Section 3, each consisting of the names that an object has in the considered vocabularies. For this reason, each $N_i$ will be called a *name set.*

In our running example (see Figure 4), the graph has 5 components, whose name sets are given by: $\{i_1, j_1, k_1\}$, $\{i_2\}$, $\{j_2\}$, $\{k_2\}$ and $\{k_3\}$.

From a complexity point of view, the computation of the name sets $N_1, \ldots, N_m$ requires a visit of the co-reference graph, thus it can be done in linear time in the size of $co$.

The implementation of `tell` (respectively, `untell`) operations is straightforward, it just reduces to adding (removing) the appropriate arc to (from) the co-reference graph. We therefore focus on `ask` and `cons-ch` operations.

### 5.1 Implementing `ask` operations

The following Proposition states the basic result for the implementation of `ask` operations.

**Proposition 1.** *For any co-reference structure $\mathcal{K} = \langle \lambda, co, nco \rangle$ and identifiers $i, j \in \mathcal{L}$ :*

1. *$j \in$ `ask-co`$(i, \mathcal{K})$ iff $j$ is co-reachable from $i$ in $G$;*
2. *$j \in$ `ask-nco`$(i, \mathcal{K})$ iff either there is a nco-arc from $j$ to $i$ in $G$ or there exists an identifier $j'$ in the same language as $j$ but different from it, such that $j'$ is co-reachable from $i$ in $G$.*

Based on this Proposition, assuming $i \in N_k$, we have:

$$\texttt{ask-co}(i, \mathcal{K}) = N_k$$
$$\texttt{ask-nco}(i, \mathcal{K}) = \{j \in \mathcal{L} \mid (i, j) \in nco\} \cup \{j \in V_j \mid j \neq j' \in V_j \text{ and } j' \in N_k\}$$

and we may therefore conclude that both `ask` operations can be implemented efficiently. `ask-nco` requires to enumerate, for every identifier co-reachable from the given one, all the different identifiers from the same vocabulary. This enumeration may be very long, but it is the only one that correctly reflects the non-co-reference knowledge.

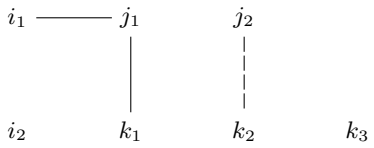


**Fig. 4.** A co-reference graph

### 5.2 Detecting inconsistency

An inconsistency is caused by the same pair to be both in $co^\star$ and in $nco^\star$. In order to devise an algorithm for eliminating the inconsistency, it is crucial to understand under which conditions a co-reference graph represents an inconsistent co-reference structure. To this end, we only need to derive the consequences of Proposition 1.

**Corollary 1.** *For every co-reference structure $\mathcal{K} = \langle \lambda, co, nco \rangle$ and pairs of identifiers $i, j \in \mathcal{L}$, $(i, j) \in co^\star \cap nco^\star$ iff:*

1. *$j$ is co-reachable from $i$ and there is a nco-arc $(i, j)$, or*
2. *$j$ and $j'$ are co-reachable from $i$, where $j' \neq j$ is an identifier in the same vocabulary as $j$.*

It is immediate to verify that the Corollary merely combines conditions (1) and (2) of Proposition 1.

The pairs of identifiers satisfying condition 1 of the last Corollary are therefore given by:

$$C_1 = \{(i, j) \in nco \mid \{i, j\} \subseteq N_k, \text{ for some } 1 \leq k \leq m\}$$

that is all pairs which are known not to co-refer and are co-reachable from one another. Computing $C_1$ simply requires, for each pair $(i, j) \in nco$, to check whether the set $N_k$ where one of $i$ or $j$ belongs, also contains the other one. This is due to the fact that, as already pointed out, the name sets are a partition of the set of identifiers showing up in $\mathcal{K}$, therefore an identifier occurs in exactly one of them. Also notice that there is no need to consider name sets that are singletons.

On the other hand, the pairs of identifiers satisfying condition 2 of the last Corollary are given by:

$$C_2 = \{(i, j) \mid \{i, j\} \subseteq N_k \cap V_h, \text{ for some } 1 \leq k \leq m, 1 \leq h \leq n\}$$

that is all pairs of identifiers which are co-reachable from one another and belong to the same vocabulary. This requires a scanning of each name set, to find 2 identifiers from the same vocabulary. Assuming each identifier carries also the id of the vocabulary it comes from, $C_2$ can be computed in at most quadratic time in the number of identifiers belonging to a name set with cardinality larger than 2.

Since the pairs $(i, j)$ satisfying one of the conditions of the Corollary are the result of the `cons-ch` operation, we have that for all co-reference structures $\mathcal{K}$,

$$\texttt{cons-ch}(\mathcal{K}) = C_1 \cup C_2$$

and we can conclude that this operation can be efficiently implemented too.

### 5.3 Repairing an inconsistency

As already pointed, out, inconsistencies may arise due to incorrect negotiations. Their repair can therefore only be done by the agents re-entering negotiations. In support of this, the knowledge co-reference services tells what are the inconsistent pairs of identifiers,as a result of a `cons-ch` operation.

Let us first discuss inconsistencies found in the set $C_1$ above. These arise from clashes between *co* and *nco*, can be repaired in one of two ways, for any pair $(i, j) \in C_1$ :

1. by removing the $(i, j)$ from *nco* via an `untell-nco`$(i, j)$ operation; in this case, negotiations have brought about that the positive co-reference knowledge is to be trusted; or
2. by making either $i$ or $j$ disappear from the name set $N_k$ where they both belong. Suppose the agent, after negotiation, decides that $i$ is the one which should not be in $N_k$. Now the agent may not know which other identifier(s) in $N_k$ $i$ has been told to co-refer with. But even if he did, removing $i$ from $N_k$ may require taking decisions about other identifiers. For instance, in the co-reference graph shown in Figure 5, if the agent decides to remove $a$ because it clashes with $d$, simply `untell`ing $(a, d)$ does not solve the problem, because $a$ and $d$ continue to co-refer via other arcs. So at least another decision needs to be taken, involving $a$ and $b$, and possibly more.

The same kind of problem arises when solving inconsistencies in the $C_2$ set, which require to remove one or more identifiers from the name set where they belong.

We conclude that, in order to support agents in the repair of inconsistency, there is no particular primitive operation that the service may offer, but the support should take the form of a powerful tool, such as a graph manipulation GUI, for carrying out the task in a most effective way.

## 6 Conclusions

To the end of establishing a necessary service for data integration, we have analyzed the notion of co-reference as it stems from language structures, that is vocabularies of identifiers with reference functions assigning an object to every identifiers. We have then defined the primitive operations of a service for maintaining co-reference knowledge. These operations reflect a functional approach to knowledge representation [13], including `tell` operations, by which users can

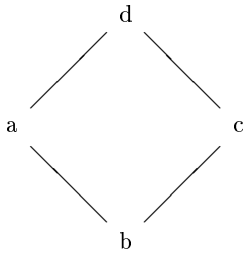


**Fig. 5.** A co-reference graph

tell the service the result of negotiations, and `ask` operations, by which users can extract implicit and explicit co-reference knowledge from the service, based on what they previously told the service. The semantics of these operations has been defined, and an efficient implementation has been derived, based on the co-reference graph.

We plan to expand these results by studying more sophisticate architectures than the centralized one considered in this paper.

## References


1. Karl Aberer, Philippe Cudré-Mauroux, Aris M. Ouksel, Tiziana Catarci, Mohand-Said Hacid, Arantza Illarramendi, Vipul Kashyap, Mecella Massimo, Eduardo Mena, Erich J. Neuhold, Olga De Troyer, Thomas Risse, Monica Scannapieco, Fèlix Saltor, Luca De Santis, Stefano Spaccapietra, Steffen Staab, and Rudi Studer. Emergent semantics principles and issues. In *DASFAA*, page 25 38, 2004.
2. Chartered Institute of Library American Library Association, Canadian Library Association and Information Professionals (Great Britain). *Anglo-American Cataloguing Rules 2004: Binder Inserts (Loose Leaf)*. American Library Association, 2004.
3. Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003),*, 2003.
4. Razvan C. Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *EACL*. The Association for Computer Linguistics, 2006.
5. Lois Mai Chan. *Library of Congress Subject Headings: Principles and Application Fourth Edition*. Library and Information Science Text Series. Libraries Unlimited, April 2005 2005.
6. Nicola Guarino. Formal ontology in information systems. In *FOIS'98*, page 3 15, 1998.
7. Patricia Harpring. Proper words in proper places: The thesaurus of geographic names. *MDA Information, 2(3), 5-12.*, 2(3):5 12, 1997.
8. L. L. Hill and Q. Zheng. Indirect geospatial referencing through placenames in the digital library: Alexandria digital library experience with developing and implementing gazetteers. In *ASIS1999*, 1999.
9. M.Baca et al. J.M.Bower. *Union List of Artist Names - A User's Guide to the Authority Reference Tool, Version 1.0, Getty Art Information Program*. G.K.Hall, New York, 1994.

10. M. Kaiser, H.-J. Lieder, K. Majcen, and Vallant H. New ways of sharing and using authority information. *D-Lib Magazine*, 9(11), 2003.
11. Y. Kalfoglou and W. M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1 31, 2003.
12. Patrice Landry. The macs project : Multilingual access to subjects (lcsh, rameau, swd). *International cataloguing and bibliographic control*, 30(3):46 49, 2001.
13. H.J Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23(2):155 212, 1984.
14. Ken Satoh Mina Akaishi, Koichi Hori. Topic tracer: a visualization tool for quick reference of stories embedded in document set. In *IV2006*, page 101 106, 2006.
15. Bijan Parsia and Peter F. Patel-Schneider. Meaning and the semantic web. In *WWW2004*, page 306, 2004.
16. Manjula Patel, Traugott Koch, Martin Doerr, and Chrisa Tsinaraki. D5.3.1: Semantic interoperability in digital library systems. Deliverable, Project no.507618, DELOS, A Network of Excellence on Digital Libraries, June 2005.
17. Edward T. O'neill Rick Bennett, Christina Hengel-Dittrich and Barbara B. Tillett. Viaf (virtual international authority file): Linking die deutsche bibliothek and library of congress name authority files. In *WLIC2006*, 2006.
18. Dagobert Soergel. The art and architecture thesaurus (aat): a critical appraisal. *Visual Resources*, 10(4):369 400, 1995.
19. National Institutes of Health United States National Library of Medicine. Unified medical language system, 2007. accessed March 6, 2007.
20. Barry Smith Werner Ceusters. Strategies for referent tracking in electronic health records. *Journal of Biomedical Informatics*, 39(3):362 378, 2006.