Managing Co-reference Knowledge for Data Integration

Carlo MEGHINI [a], Martin DOERR [b] and Nicolas SPYRATOS [c]
[a] *CNR – ISTI, Pisa, Italy*
[b] *FORTH – ICS, Crete, Greece*
[c] *Université Paris-Sud – LRI, Orsay Cedex, France*

AbstractThis paper presents a novel model of co-reference knowledge, which is based on the distinction of (i) a model of a common reality, (ii) a model of an agent's opinion about reality, and (iii) a model of agents' opinions if they talk about the same object or not. Thereby it is for the first time possible to describe consistently the evolution of the agent's knowledge and its relative consistency, and to formally study algorithms for managing co-reference knowledge between multiple agents if they have the potential to lead to higher states of consistency, independent from the particular mechanism to recognize co-reference. As an example, a scalable algorithm is presented based on monitoring atomic knowledge increments and an abstract notion of belief ranking. The presented approach has a wide potential to study the formal properties of current methods to find co-reference, and to lead to new methods for the global management of co-reference knowledge.

Keywords. Co-reference, knowledge service, algorithms

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 aspect of information integration. Independently if data are brought under a common schema or the sources operate in their own independent ways, and we simply try to connect them into a coherent network of knowledge. The two basic requirements for such a network to be created and managed are the following:

1. Decide whether two data elements residing in different sources refer to the same thing. [15] describes this relationship between data elements as the co-reference relation.
2. Maintain persistent co-reference relation and make it accessible to the individual sources (agents) 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 . There are some methods to assist finding co-reference: 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. Scientists and scholars all over the world maintain card boxes or create dedicated indices in books to store co-reference knowledge. Hypertext links and other "see also" links may indirectly encode co-reference knowledge. These methods suffer either from precision or scalability. OWL introduces the relationship "same as", without exploring the theoretical question in which way two nodes can be distinct and still be "same as" at the same time.

There is no systematic literature about the epistemological nature of co-reference knowledge itself, i.e., what are the relations between the knowldege of different agents and reality, and what are the possible inconsistencies, how can they be detected and reduced. Consequently, there is no widely accepted concept of long-term preservation and management of co-reference knowledge itself. Rather than dealing with ways to identify co-reference, this paper simply assumes that there *are* ways of varying reliability to identify co-reference or non-co-reference relations and deals with the fundamental problem what to do with this knowledge in a distributed environment, in order to increase local and global consistency and the degree of knowledge sharing. To do so, we present a novel model of co-reference knowledge, which is based on the epistemological distinction of (i) a model of a common reality, (ii) a model of an agent's opinion about reality, and (iii) a model of agents' opinions if they talk about the same object or not. Thereby it is for the first time possible to describe consistently the evolution of the agent's knowledge and its relative consistency, and to formally study algorithms for managing co-reference knowledge between multiple agents if they have the potential to lead to higher states of consistency, independent from the particular mechanism to recognize co-reference. As an example, a scalable algorithm is presented based on monitoring atomic knowledge increments and an abstract notion of belief ranking.

In this paper, we only interested in whether or not a co-reference relation is assumed. The idea is that the established 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 potentially billions of identifiers for things, people, places etc. We are not interested in the odds of natural language interpretation. Co-reference clusters, i.e. the set of links being found to connect references to the same object, are normally very small. Most things in the world are rarely referred to. However, some things,

such as Goethe or Paris, may be referred to many many times. We assume that the topology of co-reference clusters, once established, will be similar to that of reference networks of scientific papers. Some sources are preferred as reference, so that a pattern of multiply connected "stars" of different size is the most likely one. Wrong connections between large clusters may cause an immense number of wrong inferences. Therefore efficient ways to narrow down possibly wrong connections and to resolve inconsistencies in a distributed environment are a major concern.

Related work

The is virtually no related work about co-reference knowledge itself. Since the problem becomes more and more urgent in the Semantic Web, just recently a paper appeared that makes a limited proposal to manage co-reference [10]. It fails however to make the necessary epistemological distinctions. A fuzzy notion of context is introduced, and synonyms within a context are "bundled". The authors assume that coreference between different contexts may point to different things, confusing the problem of agreement if something is one or two things with the problem to know if someone speaks about the same thing or not. Therefore, they do not provide a viable method to manage co-reference knowledge . In [17], the authors recognize the fact that standardization and centralization is not necessary to manage 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 items on the Web.

A major motivation for this paper is to describe how the current approaches to support co-reference detection can be deployed more effectively. These approaches 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 (a) compare the attributes of items in different data sources, and (b) estimate the probability that they mean actually the same thing (e.g., it is reasonable to assume that two references to a person with the same name, same birth date and birth place are actually identical). Unfortunately, different sources tend to register different properties for the same things, properties of items may change over time or be unknown. It requires *pre-existing shared* knowledge about the values of the properties being compared. Actually, data cleaning methods mix three different probabilities: (a) that two different items have *accidentally* the same properties, (b) that the same properties are at all registered for the same item in different sources and (c) that these properties are registered in the same way in different sources. These factors limit the reliability of these methods, but since they are scalable, they are efficient to guide manual verification to *find co-reference* based on *further evidence*. We use in this paper the fact that there are methods providing a belief or reliability ranking in a set of co-reference links. Normally, in reactive methods, detected co-reference is *not* preserved beyond the immediate scope and purpose. We claim that these methods would be far more effective if the co-reference links they produce would be globally preserved and published for open access and future, semiautomatic validation.

In *pro-active* methods, sometimes also found under *data cleaning*, 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.

Librarians, scholars and scientists from many disciplines have been heavily investing in so-called authority files or better knowledge organisation systems (KOS) [18], which register names and characteristics of authors [12], historical persons [11], places ([9],[8]), books and other items, as well as categories (Library of Congress Subject Headings[5], the Art & Architecture Thesaurus[20], Unified Medical Language System UMLS[21], 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[19], LEAF[12] for person authorities, and MACS[14] for subject headings, have understood the relevance of *managing and preserving 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 might be 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.[13]), here we address situations that a priori, but not exclusively, apply to particulars. We describe methods that can be applied to managing the connection of multiple KOS by co-reference links.

A variant of the pro-active methods is the *referent tracking* suggested by [22]. 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.

Akaishi et al. [16] describe a statistical information retrieval method, in which they trace related documents via characteristic co-occurrences of terms common to docu-

ments. 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. Non does really manage co-reference knowledge once it is found. In this paper, we present a simple, but effective epistemological model to manage co-reference knowledge. 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. We thereby implicitly show that a central authority and exhaustive description of items is not necessary, as useful it might be. All existing automated methods to detect co-reference may be used to support or enhance the respective agreement processes.

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;
- a *reference function* R_i associating each identifier in V_i with some object in Obj.

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 sets $V_1, \ldots, V_n$ are all distinct and pairwise disjoint; we will let $\mathcal{V}$ stand for the collection 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 $i = 1, \ldots, n$;

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

R_1	
i_1	1
i_2	2

R_2	
j_1	1
j_2	3

R_3	
k_1	1
k_2	2
k_3	3

Figure 1. Elements of a language structure

- *injective,* that is no two identifiers from the same vocabulary denote the same object; in other words, vocabularies are to be understood as, *e.g.authority files* in libraries, each offering "official names" for the domain entities; it is crucial that each such name be unique within the language; of course, any official name can have a number of synonyms which the users of the language may use at their will;
- *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* λ is a 4-tuple$\lambda = \langle Obj, \mathcal{A}, \mathcal{V}, \mathcal{R}, \rangle$ satisfying all the above assumptions. The *vocabulary* of λ 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 some elements of 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 λ, 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 important to realize the difference between co-reference and synonymy: co-reference holds between identifiers of *different languages,* whereas synonymy holds between an identifier in V_i and any number of synonyms from the associated synonym set S_i.

It is easy to see that the co-reference relation is an equivalence relation, which induces a partition $[\mathcal{L}]$ 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

$\approx^r$	
i_1	j_1
j_1	k_1
i_2	k_2
j_2	k_3

$1 \longleftrightarrow \{i_1, j_1, k_1\}$
$2 \longleftrightarrow \{i_2, k_2\}$
$3 \longleftrightarrow \{j_2, k_3\}$

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

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 shown in association with the object they name.

Sharing co-reference knowledge

We view the language structure λ as being the "reality" under study. Initially, each agent A_i knows part of the reality, namely his own vocabulary V_i and reference function R_i as well as synonym set S_i and relation syn_i. The knowledge of the agent increases when he engages in communication with another agent A_j. This communication can be direct (or synchronous) if it takes places when both the agents are present, or indirect (asynchronous) 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

co	
i_1	j_1
j_1	k_1

nco	
j_2	k_2

Figure 3. Tables of a co-reference structure

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;
- `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:

- λ 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 specify a semantics for the operations defined above.

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) 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 . 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:

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

  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 λ 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.

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, ignoring knowledge explicitly told 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: 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.

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 [6]. 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 , each consisting of the names that an object has in the considered vocabularies. For this reason, each graph (N_i, E_i) will be called *name graph* while 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 $\{k_3\}$.

Implementing `ask` *operations*

The following Proposition states the basic result for `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 there exist two identifiers a and b such that $a \in$ `ask-co`$(i, \mathcal{K})$, $b \in$ `ask-co`$(j, \mathcal{K})$ and either a and b belong to the same language, or there is a nco-arc from a to b in G.*

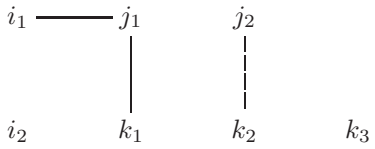

Figure 4. A co-reference graph

Proof: *(1) (→) If j is co-reachable from i in G, then $(i, j) \in co^\star$ by transitivity.*
(←) We have that $co^\star = f^n(co)$ for a finite n (see Section). We prove that for all $1 \leq k \leq n$, $(i, j) \in f^k(co)$ implies that j is co-reachable from i. The proof is by induction on n. For $k = 1$, $f^1(co) = co$. Thus if $(i, j) \in co$ then there is an arc from i to j in G, hence j is co-reachable from i. Suppose the thesis holds for $k = m < n$, let us prove it for $k = m + 1$. $f^{m+1}(co)$ adds to $f^m(co)$ the pairs that are obtained from those in $f^m(co)$ either by symmetry or by transitivity. In the case of symmetry, if $(i, j) \in f^{m+1}(co)$, it means that $(j, i) \in f^m(co)$, which means by the induction hypothesis that i is co-reachable from j; then, for the symmetry of co-reachability, also j is co-reachable from i. In the case of transitivity, $(i, j) \in f^{m+1}(co)$ implies that $(i, h), (h, j) \in f^m(co)$, for some identifier h. By the induction hypothesis, h is co-reachable from i and j is co-reachable from h, which implies that j is co-reachable from i.
The proof of (2) is much simpler and is omitted for reasons of space. □

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.

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 different identifiers $i, j \in \mathcal{L}$, $(i, j) \in co^\star \cap nco^\star$ iff i, j belong to the same name set N_k, for some k, and:*

1. *either N_k contains two identifiers joined by an nco-arc, or*
2. *N_k contains two identifiers from the same language.*

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:

$$C_1 = \{(i,j) \mid i \neq j,\ \{i,j\} \subseteq N_k \text{ and } N_k^2 \cap nco \neq \emptyset,\ \text{for some } 1 \leq k \leq m\}$$

that is all pairs of identifiers which are in the same name set as two pairs that are known not to co-refer. 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. If yes, then all pairs in N_k are in C_1.

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

$$C_2 = \{(i,j) \mid \{i,j\} \subseteq N_k \text{ and } |N_k \cap V_h| \geq 2,\ \text{for some } 1 \leq k \leq m, 1 \leq h \leq n\}$$

that is all pairs of identifiers which are in the same name set as two pairs from the same vocabulary. This requires a scanning of each name set, to find 2 identifiers from the same vocabulary. whenever such a name set is found, all pairs in it are in C_2. Assuming each identifier carries also the identifier of the vocabulary it comes from, C_2 can be computed efficiently.

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.

Repairing an inconsistency

The inconsistencies found in the set C_1 arise from the fact that a pair (i,j) is both in co and in nco. In this case, all pairs of different identifiers in the same name set as i and j are in C_1, but obviously the prime cause of the inconsistency is the pair (i,j); hence, the repairing will focus on this kind of pairs. One of two actions can then be performed:

- 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
- making either i or j disappear from the name set N_k where they both belong. This requires breaking all paths connecting i and j in the name graph (N_k, E_k), by making a number of `untell-co` operations which collectively would *cut* i from j, in fact causing the name graph to split into two graphs.

Analogously, the prime causes of the inconsistencies found in the set C_2, are those identifiers i and j that belong to the same language and also to the same name set. In order to solve these inconsistencies, therefore, one of two actions can be taken:

- to make i and j synonyms of one another, thus using one of the two as a representative of both, and recording this information somewhere[1]; in practice, this would mean to substitute the removed identifier with the other one in all co-reference relationships in which it occurs; or

[1] Synonym lists are commonly associated to authority files.

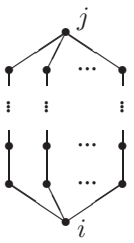

Figure 5. A co-reference graph

- to cut i from j in the co-reference graph.

Clearly, in both cases the second operation is the most difficult one, and the question arises how the co-reference service can support the user in identifying a set of wrong links making up a cut, *i.e.*the co-reference relationships that must be `un-tell`-ed in order to restore consistency.

The service does not have any clue as to what each identifier refers to, and the only optimality criterion that it can follow is to propose to the user *minimal* cuts, thus minimizing the number of negotiations which the user has to carry out. However, this is not a feasible strategy, because there can be an exponential number of minimal cuts. Consider for instance the graph shown in Figure 5: any set including a link from each of the paths connecting i and j is a minimal cut, and their number of such cuts is exponential in the number of paths.

In fact, even identifying *any* minimal cut is a difficult problem, which can be shown to be NP-complete.

For the hardness, Figure 6 shows a reduction from MINIMAL HITTING SET [7] to the problem of saturating the network on the right with a minimal number of links, which is the same as finding a minimal cut between i and j. A MINIMAL HITTING SET instance consists of a finite set S and a collection $\mathcal{C} = \{C_1, \ldots, C_n\}$ of subsets of S. The question is whether there exists a subset X of S of minimal size, containing at least one element of any member C_i of $\mathcal{C}$. The network corresponding to this problem has 3 types of links:

- the links outgoing from i are one-to-one with the element of S, and so are the different nodes they lead to; the capacity of each such links is the outrank of the target node, to make sure all links outgoing from each target node are used;
- the links incoming into j are one-to-one with the members of the collection $\mathcal{C}$, and so are the different nodes they leave from;
- the links in between these, connect each node corresponding to an element x of S with the nodes corresponding to the elements $C_i \in \mathcal{C}$ in which x belong; each link of this and the previous type has capacity 1 to make sure it is used exactly once.

In order to saturate the network, a minimal set of links outgoing from i must be chosen, so that all links incoming into j are saturated. The way the network is built clearly guarantees that such a minimal set of links is one-to-one with a MINIMAL HITTING SET for the initial problem.

Membership in NP can be easily seen by considering that the size m of a minimal cut between two nodes of a graph can be determined in polynomial time in the size of

$S = \{1, 2, 3, 4, 5\}$

$C_1 = \{1, 2, 3\}$

$C_2 = \{2, 3, 4\}$

$C_3 = \{3, 4, 5\}$

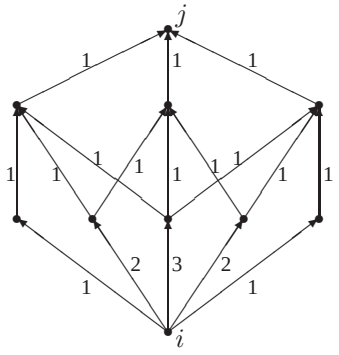

Figure 6. Reduction from MINIMAL HITTING SET

the graph. Given then a candidate solution s to the problem in question, it suffices to determine m and check whether the size of s is the same as m. If yes, then s is a minimal set of links; if not, it is not. All this can be done efficiently, thus determining a minimal set of links is an NP-complete problem.

For these reasons, we envisage an interactive and iterative strategy for cutting two identifiers in the co-reference graph. The strategy is based on the obvious observation that a cut must include a link from any path from i to j. We then envisage a `repair`(i, j) operation, where $(i, j) \in$`cons-ch`, which, when issued by the user, causes the following interaction to take place:

1. The service tries to compute any co-path from i to j.
2. If no such co-path exists, then the repair operation is completed successfully. Clearly at the first iteration a co-path is always found as a consequence of the fact that $(i, j) \in$`cons-ch`.
3. The service shows to the user the links on the obtained co-path which the user has not yet considered (*i.e.*, re-negotiated). At the first iteration no link has been re-negotiated by the user, so all links on the co-path are shown; from the second iteration on, the user may have already negotiated some links, which need not therefore be shown to her again.
4. The user, via negotiations, identifies at least one, possibly many of the shown links as incorrect and `untell-co` each one of them.
5. The services executes each `untell-co` operation issued by the user and iterates.

In order to efficiently execute its part of the above protocol, the co-reference service must be able to quickly determine whether i and j are connected in the co-graph (step 1), and, if so, serve the links on a co-path to the user, discarding the ones already considered in previous iterations (step 3). One way of doing so, is to maintain a *spanning tree* of each name graph (N_k, E_k). A spanning tree of a graph is a minimal set of edges of the graph that connect all nodes. By definition, in a tree there is exactly one path from any two nodes, so the spanning tree is cycle-free and therefore very suitable for the task at hand.

The next Section outlines algorithms and data structures for implementing the co-reference service, based on the results presented so far.

Algorithms and data structures

The basic data structure for co-reference is a table C with four columns, each holding the information about a co-reference pair; in particular,

- the first two columns are used to store, respectively, the first and second element of the co-reference pair;
- the third column stores an identifier of the name set where the co-reference pair belongs;
- the last column stores a binary value v indicating whether ($v = 1$) or not ($v = 0$) the co-arc representing the pair belongs to the spanning tree of the name graph.

The data in the last 2 columns do not represent any entity in the formal model, they are just used for implementation purposes. Non-co-reference information is stored in a two column relation N, each row of which keeps a non-coreference pair.

In what follows, we will sketch the execution of each operation in the interface of the co-reference service, by analyzing the possible scenarios and outlining the actions to be taken. The following invariants regarding the table C can be easily proved, for instance by induction on the size of C :

- *Co-arc uniqueness.* The first two columns are a key of C, in any order (*i.e.*, C has two keys). Moreover, the projection of C on the first two columns is an asymmetric relation. Overall, this means that $(i, j, x, y) \in C$ implies that for no $z \neq x$ and $w \neq y$, $(i, j, z, w) \in C$ or $(j, i, z, w) \in C$. For simplicity, we call a "ij tuple" any tuple that has i and j (in any order) in the first two columns.
- *Weak C_2 consistency.* $(i, j, x, y) \in C$ implies that i and j do not belong to the same language. This does not mean that C is C_2-consistent, but simply that "evident" inconsistencies are avoided by maintaining a synonym list for each identifier. We do not enter into the details of how these lists are implemented.
- *Name consistency.* Language and name set idenfiers are consistent: it cannot happen that the same language identifier is associated with two different name set identifiers in two different tuples of the C table. More technically, there is a functional dependency from either of the first two columns of C to the third. Based on this dependency, we use the notation $\nu(i)$ to indicate the identifier of the name set where i belongs.

For brevity, we only consider `tell` operations, and conclude with a remark on how to efficiently perform the first step of a `repair`, aiming at finding a path from two given identifiers from the same name set.

Figure 7 presents the `tell-co` procedure. `tell-co` takes as input two identifiers i and j and if there already exists an ij tuple in C, it does nothing. Otherwise,

- If both i and j do not occur in C (line 2), then: if i and j belong to the same language, then in order to maintain the weak C_2 consistency, i is chosen to be the official representative of both, having the other as a synonym; if i and j belong to two different languages, then the insertion of the tuple $(i, j, A, 1)$ into C means that a new name set is created, having A as identifier and including both i and j; in addition, (i, j) is made part of the spanning tree of A.

procedure `tell-co`(i, j : **identifiers**)

1. **if** no ij tuple exists in C **then**
2. **if** neither i nor j occur in any tuple in C **then**
3. **if** i and j belong to the same language **then** $syn(i) \leftarrow \{j\}$
4. **else** `insert`$(i, j, A, 1)$ into C where A is a new name set identifier
5. **else if** j does not occur in any tuple in C **then**
6. **if** i and j belong to the same language **then** $syn(i) \leftarrow syn(i) \cup \{j\}$
7. **else** `insert`$(i, j, \nu(i), 1)$ into C
8. **else**
9. **if** i and j belong to the same language **then begin**
10. `replace` $\nu(i)$ by A in C where A is a new name set identifier
11. `replace` $\nu(j)$ by A in C
12. `replace` j by i in C
13. $syn(i) \leftarrow syn(i) \cup \{j\}$
14. **end**
15. **else**
16. **if** i and j belong to the same name set A, **then** `insert`$(i, j, A, 0)$ into C
17. **else begin**
18. `replace` $\nu(i)$ by A in C where A is a new name set identifier
19. `replace` $\nu(j)$ by A in C
20. `insert`$(i, j, A, 1)$ into C
21. **end**

Figure 7. The `tell-co` procedure

- If one of i and j (say j) does not occur in C, there can be two cases: (1) i and j belong to the same language (line 6), in which case j is stored as a synonym of i. (2) i and j belong to two different languages (line 7), then the fact that j co-refers with i is recorded by inserting $(i, j, \nu(i), 1)$ into C, along with the fact that (i, j) is part of the spanning tree of $\nu(i)$.
- If both i and j occur in C, there can be the same two cases: (1) i and j belong to the same language (line 9); in this case, it follows from the weak C_2 consistency of C that i and j belong to two different name sets. Then, in term of the co-graph, i and j are coaleshed into a single node representing one of the two identifiers, having the other one as a synonym. This is implemented by (a) replacing both $\nu(i)$ and $\nu(j)$ by a new name set identifier A in C; (b) replacing j by i in C; and finally (c) storing j as a synonym of i. (2) i and j belong to two different languages; in this case, if i and j belong to the same name set A, then $(i, j, A, 0)$ is inserted into C since i and j are already connected by the spanning tree of A and therefore the newly added co-arc does not have to be on the spanning tree. If i and j belong to two different name sets, then `tell-co` connects two whole name sets, as follows: (1) a new name set identifier A replaces both $\nu(i)$ and $\nu(j)$ in C; (2) $(i, j, A, 1)$ is inserted into C.

Figure 8 presents the `untell-co` procedure. `untell-co` takes as input two identifiers i and j which belong to different languages by the C_2-consistency of C. If no ij tuple exists in C, nothing is done. Else, let (i, j, A, v) be such a tuple, unique by co-arc uniqueness.

procedure `untell-co`$(i, j$: **identifiers)**

1. **if** there exists an ij tuple (i, j, A, v) in C **then**
2. **if** $v = 0$ **then** `remove` (i, j, A, v) from C
3. **else begin**
4. $N_i \leftarrow \{k \mid (i, k)$ is a path in the spanning tree of A without the (i, j) co-arc$\}$
5. $N_j \leftarrow \{k \mid (j, k)$ is a path in the spanning tree of A without the (i, j) co-arc$\}$
6. **if** no tuple (x_i, x_j, X, b) exists in C where $x_i \in N_i$ and $x_j \in N_j$ **then begin**
7. `remove` (i, j, A, v) from C
8. `replace` each tuple (x, y, X, c) in C such that $x, y \in N_i$, by (x, y, A_i, c) where A_i is a new name set identifier for N_i
9. `replace` each tuple (x, y, X, c) in C such that $x, y \in N_j$, by (x, y, A_i, jc) where A_j is a new name set identifier for N_j
10. **end**
11. **else**
12. **for each** tuple $(x, y, A, 0)$ in C such that $x \in N_i$ and $y \in N_j$ **do begin**
13. `remove` $(i, j, A, 1)$ from C
14. `replace` the tuple $(x, y, A, 0)$ in C by $(x, y, A, 1)$
15. **end**
16. **end**

Figure 8. The `untell-co` procedure

- If $v = 0$, then the corresponding co-arc (i, j) is not on the spanning tree of the name set A, hence the tuple is removed from C.
- If $v = 1$, N_i (respectively, N_j) is the set of identifiers reachable from i (j) in the spanning tree of A without the (i, j) co-arc. By definition of spanning tree, N_i and N_j are a partition of the name set where i and j belong. There can be two cases: (1) if no tuple exists in C having x_i and x_j in the first two columns, where $x_i \in N_i$ and $x_j \in N_j$, then (i, j) is the only co-arc connecting i and j; in this case two different name sets are created with the remaining co-arcs in the name set A (lines 7-9). (2) If $(x, y, A, 0)$ is any tuple in C such that $x \in N_i$ and $y \in N_j$, it is replaced by $(x, y, A, 1)$ and the tuple $(i, j, A, 1)$ is deleted from C.

The procedures for operating on non-coreference are quite straightforward. In particular:

- `tell-nco`(i, j) does nothing if an ij tuple exists already in N. Otherwise, the tuple (i, j) is added to N.
- `untell-nco`(i, j) does nothing if no ij tuple exists in N. Otherwise, the tuple (i, j) is removed from N.

Finally, the crucial step in `repair`(i, j) is the identification of the path on the spanning tree leading from i to j. By using a simple backtracking strategy, this requires connecting the co-arcs corresponding to the tuples in C marked with a 1 in the fourth column, starting from those who have an i as one of the two ends, until one is found which as j as one of the two ends.

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 [15], including tell operations, by which users can 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.

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] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT Press, 2nd edition, 2001.

[7] Michael R. Garey and David S. Johnson. *"Computers and Intractability: A Guide to the Theory of NP-Completeness"*. W. H. Freeman and Company, New York, 1979.

[8] Patricia Harpring. Proper words in proper places: The thesaurus of geographic names. *MDA Information, 2(3), 5-12.*, 2(3):5 12, 1997.

[9] 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.

[10] A. Jaffri, H. Glaser, and I. Millard. Uri identity management for semantic web data integration and linkage. Submitted to 3rd International Workshop On Scalable Semantic Web Knowledge Base Systems, Vilamoura, Algarve, Portugal, 2007.

[11] 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.

[12] M. Kaiser, H.-J. Lieder, K. Majcen, and Vallant H. New ways of sharing and using authority information. *D-Lib Magazine*, 9(11), 2003.

[13] Y. Kalfoglou and W. M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1 31, 2003.

[14] Patrice Landry. The macs project : Multilingual access to subjects (lcsh, rameau, swd). *International cataloguing and bibliographic control*, 30(3):46 49, 2001.

[15] H.J Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23(2):155 212, 1984.

[16] 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.

[17] Bijan Parsia and Peter F. Patel-Schneider. Meaning and the semantic web. In *WWW2004*, page 306, 2004.

[18] 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.

[19] 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.

[20] Dagobert Soergel. The art and architecture thesaurus (aat): a critical appraisal. *Visual Resources*, 10(4):369 400, 1995.

[21] National Institutes of Health United States National Library of Medicine. Unified medical language system, 2007. accessed March 6, 2007.

[22] Barry Smith Werner Ceusters. Strategies for referent tracking in electronic health records. *Journal of Biomedical Informatics*, 39(3):362 378, 2006.