An algorithm for semantic coordination

P. Bouquet[2,3], L. Serafini[3], S. Zanobini[2] and M. Benerecetti[1],

[1]Dept. of Physical Sciences
University of Naples
Via Cinthia – 80126 Napoli (Italy)

[2]Dept. of Information and Communication Technology
University of Trento
Via Sommarive, 10 – 38050 Trento (Italy)

[3]ITC-IRST – Istituto per la Ricerca
Scientifica e Tecnologica
Via Sommarive, 14 – 38050 Trento (Italy)

bene@na.infn.it, bouquet@dit.unitn.it, serafini@itc.it, zanobini@dit.unitn.it

Abstract

The problem of finding an agreement on the meaning of heterogeneous semantic models is one of the key issues in the development of the Semantic Web. In this paper, we propose (i) a *general algorithm* which implements a new approach, called CTXMATCH, for discovering (semantic) relationships across distinct and autonomous *generic structures* and (ii) a *specific algorithm* specializing the algorithm to the discovering of mappings across *hierarchical classifications*. This approach shifts the problem of semantic coordination from the problem of computing linguistic and/or structural similarities between semantic-based structures (what most other proposed approaches do), to the problem of deducing relations between sets of logical formulas that represent the meaning of concepts belonging to different structures.

1 Introduction

The approach to semantic coordination we proposed in [6, 7] is based on the intuition that there is a huge conceptual difference between coordinating abstract structures (e.g., arbitrary labelled graphs) and coordinating structures labeled with expressions of a language spoken by the community of their users. The latter ones give us the chance to exploit the complex degree of semantic coordination implicit in the way a community uses the language from which the labels are taken.

We believe that at least three distinct levels of semantic knowledge are needed in order to semantically coordinate structures labelled with natural language:

• **Lexical knowledge:** knowledge about the words used in the labels. For example, the fact that the word 'image' can be used to mean a picture or a personal facade;

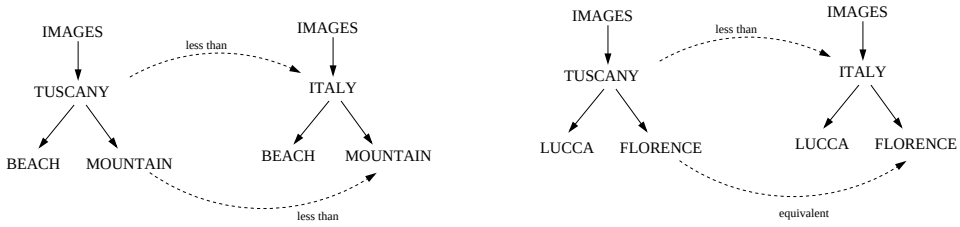
• **Domain knowledge:** knowledge about the relation between senses of labels in the real world or in a specific domain. For example, the fact that Florence is both a city of Italy and of Tuscany;

• **Structural knowledge:** knowledge deriving from the way the labels are arranged in a given structure. For example, the fact that the node `MOUNTAIN` in Figure 1.a can be used to classify images of mountains, and not books.

In [6, 7] we deeply motivate this choice. To summarize these motivations, consider the hierarchical classifications (hereafter HC) in Figure 1 used to classify images in two multi-media repositories. We want to discover the semantic relation between the nodes labelled `MOUNTAIN` in the two HCs in Figure 1.a, and between the two nodes `FLORENCE` in Figure 1.b. Human reasoners understand almost immediately that the relation between the first pair of nodes is "less general than" (after all, the images that one would classify as 'images of mountains in Tuscany' are a subset of the images one would classify under 'images of mountains in Italy'), while that the relation between the second pair of nodes is "equivalent" (in fact, the images that one would classify as 'images of Florence in Tuscany' are the same as the images that one would classify under 'images of Florence in Italy'). Notice that the two relations are different, even though the two pairs of HCs are structurally equivalent. Using the three semantic levels mentioned above, we can account for this difference. Consider the mapping between the two nodes `MOUNTAIN`. *Linguistic knowledge* tells us that the sense of the two labels is the same. *Domain knowledge* tells us, among other things, that Tuscany is a region of Italy. Finally, *structural knowledge* tells us that the intended meaning of the two nodes `MOUNTAIN` refers to images of mountains of Tuscany (left HC), and images of Italian mountains (right HC) respectively. All these facts together allow us to conclude that one node is less general than the other one. We can use a similar reasoning for the two nodes `FLORENCE`. But, exploiting *domain knowledge*, we can add the fact that Florence is both in Tuscany and in Italy (such a relation doesn't hold between mountains and Italy or Tuscany in the first example). This further piece of knowledge allows us to conclude that, despite structural equivalence, the relation is different.

2 CTXMATCH: the general algorithm

The general framework described in Section 1 can be used for discovering relations between any *structures labelled with natural language*. In this section, we introduce the structure and purpose independent part of the algorithm, namely the steps that do not depend on the use nor on the type of structure. This *generic algorithm* must be obviously enriched with *spe-*



(a) (b)

Figure 1: Coordinating HCs

cific structure and purpose dependent functions, i.e. with different functions for each particular type and use of the structures we want to match. In Section 3 we present the specific functions we use to match Hierarchical Classifications, i.e., tree-like structures used for classifying documents.

To make things clearer, imagine the following scenario: an agent *A* (the seeker) has a set of documents organized into a tree–structure. To collect new documents, he can send a query to a provider (an agent *B*). In our approach, the agent can formulate the query using his own structure: for example, imagine that seeker *A* uses the structure on the right-hand side of Figure 1.b to classify his documents. Then, he can select node FLORENCE to formulate the query 'Images of Florence in Italy'. Furthermore, imagine that the provider employs the left-hand structure in Figure 1.b. After receiving the query, he has the following tasks: (i) to interpret the query he receives, (ii) to find semantic relations holding between the query and his structures, and (iii) to return relevant documents (if any). In particular, in this paper we focus on the tasks (i) and (ii).

The algorithm needs two **inputs**:

query *Q***:** A seeker sends a query composed by a node fl in a structure *FS*. It means simply that the seeker wants to find nodes semantically related to the node fl in *FS*;

context *C***:** The context is composed by the three elements of the local knowledge, namely a structure *LS*, a lexicon *LL* and an ontology *LO*. The context is the target of the query[1].

The main goal of the algorithm CTXMATCH is to find the semantic relations between node fl in the query *Q* and all the nodes belonging to the local structure *LS* in the context *C*. For the sake of simplicity, in this paper we focus on the procedure for matching the node fl in the query with a single nodes ll in the context *C*. Therefore, for this simplified version of CTXMATCH, we add a third element in the input: a label ll of the structure *LS*. The **output** of the algorithm will simply be the semantic relation holding between the two nodes.

The algorithm also employs a data–type 'concept' $\langle \phi, \alpha \rangle$, constituted by a pair of logical formulas, where ϕ approximating the *individual concept* represented by a node of a structure and α expressing the relations between the current individual concept and other individual concepts in the structures (*local relevant axioms*). E.g., the formulas associated with the node labeled FLORENCE in rightmost structure in Figure 1.b will approximate the statements 'images of Florence in Italy' (the individual concept) and 'Florence is in Italy' (the local relevant axiom).

Algorithm 1 CTXMATCH(Q,C,ll)

▷ *query* $Q = \langle fl, FS \rangle$ *where* fl *is the foreign term*
FS is the foreign structure

▷ *context* $C = \langle LS, LL, LO \rangle$ *where LS is the local structure*
LL is the local lexicon
LO is the local onltology

▷ *label* ll *is the label of the local node to be matched*

VarDeclarations
context *QC*;
concept $\langle \phi, \alpha \rangle, \langle \psi, \beta \rangle$; ▷ *concepts are pairs of formulas*
relation *R*;

1 $QC \leftarrow \langle FS, LL, LO \rangle$;
▷ *QC represents the virtual query context*
2 $\langle \phi, \alpha \rangle \leftarrow$ BUILD−CXT−MEANING(fl, QC);
3 $\langle \psi, \beta \rangle \leftarrow$ BUILD−CXT−MEANING(ll, C);
▷ *compute the concepts expressed by label* ll *and* fl
4 $R \leftarrow$ SEMANTIC−COMPARISON$(\langle \phi, \alpha \rangle, \langle \psi, \beta \rangle, LO)$;
▷ *R represents the semantic relation between the two concepts*
5 **return** *R*;

In line 1, CTXMATCH first builds the 'virtual' query–context *QC*. The reason of it is that we want the query *Q* to be locally interpreted within the local lexicon and ontology. An important consequence is that the relation returned by the algorithm is **directional**: it expresses *the relation holding between the two nodes from the provider's point of view*. Indeed, the seeker could have different lexicon and ontology and could calculate different relation for the same nodes.

Then, line 2 builds a concept, i.e. a pair of logical formulas, approximating the meaning of the node fl in the virtual context *QC*. Line 3 similarly builds the concept for the node ll in the local context *C*. Finally, line 4 computes the *semantic relation* between the two concepts. The following two subsections describes in more detail this two top-level operations, implemented by the functions BUILD−CTX−MEANING and SEMANTIC−COMPARISON.

2.1 Building the contextual meaning

This step has the task of building the concept expressed by a generic node *t* in a generic context *GC*. Before analyzing the corpus of the algorithm, it's important to focus our attention on the array of senses *SynS*. A synset (set of synonyms) is a set of senses, i.e. of concepts, expressed by an expression

[1] We call context the ensemble of the three levels of knowledge because they express the local representation that an agent has of a portion of the world.

of the natural language[2]. For example the word 'Florence' has, in WORDNET, two senses (i.e. it may express two different concepts): 'city of Tuscany' and 'town of South Caroline'. The array *SynS* records these senses, so that, for example, *SynS*[Florence] is the synset containing the two senses above, while *SynS*[Florence][0] is the first of the two senses.

Let us now look at the algorithm. Line 1 determines the focus of a node t, i.e. the subgraph of the structure T useful to extract the meaning of t. This step is performed essentially for efficiency reasons, as it reduces as much as possible the node space to take into account. Lines 2-3 associate to each node within the focus the synsets found in the Lexicon. Consider the Figure 1.b: the two synsets 'city of Tuscany' and 'town of South Caroline' are associated to the label FLORENCE.

Lines 4-5 try to filter out unreasonable senses associated to t. In our example, 'town of S.C.' is discarded since it is incompatible with the other labels in the focus of t (in fact, node FLORENCE refers clearly to the city in Tuscany – see Algorithm 4).

Algorithm 2 BUILD−CTX−MEANING(GC,t)

▷ *context* $GC = \langle T, L, O\rangle$, *where* T *is a structure*
L is a lexicon
O is an onltology
▷ *label* t *is a generic label*

VarDeclaratrions
sense *SynS*[][] ▷ *array of senses*
structure F
fomula α, η

1 $F \leftarrow$ DETERMINE−FOCUS(t, T);
▷ *the focus* F *is a substructure of* T
2 **for** each label e in F **do**
3 *SynS*$[e] \leftarrow$ EXTRACT−SYNSET(e, L);
▷ *extracts the senses associated to each label in the structure* F
4 **for** each label e in F **do**
5 *SynS*$[e] \leftarrow$ FILTER−SYNSET$(F, O, SynS, e)$;
▷ *unreasonable senses are discarded*
6 $\delta \leftarrow$ INDIVIDUAL−CONCEPT$(t, SynS, F, O)$;
7 $\eta \leftarrow$ EXTRACT−LOCAL-AXIOMS$(F, SynS, O)$;
8 **return** $\langle \delta, \eta \rangle$;

Finally, lines 6 and 7 build the two component of the concept expressed by node t, computing the *individual concept* and the *local relevant axioms*, as we explained in describing Algorithm 1.

2.2 Comparing the concepts

The main task when comparing two concepts is to find the semantic relation holding between them. The algorithm employs the data–type 'deductional–pair': this is an array of pairs $\langle$*relation, formula*$\rangle$, where the *formula* expresses the condition under which the semantic *relation* between the concepts holds. E.g., the deductional–pair $\langle \equiv, \alpha \rightarrow \beta \rangle$ means that if $\alpha \rightarrow \beta$ is valid, then the relation holding between the two concepts is the equivalence ($\equiv$).

Line 1 extracts *global axioms*, i.e. the relations holding between individual concepts belonging to different structures. Consider, for example, the nodes ITALY AND TUSCANY in Figure 1.b: the global axioms express the fact that, for example, 'Tuscany is a region of Italy'. Line 2 builds the array of deductional–pair. It's important to note that the relations, their number and the associated conditions depend on the type of structure to match. In Section 3 we report the pairs relation/condition relevant for matching HCs. Lines 3–6 look for the "correct" relation holding between two concepts. This is done by checking the formulas in each deductional–pair, until a valid one is found[3]. If a valid formula is found, the associated relation is returned.

It's important to observe that the problem of finding the semantic relation between two nodes $t \in T$ and $t' \in T'$ is encoded into a satisfiability problem involving both the formulas extracted in the previous phase, and some further *global relevant axioms*. So, to prove whether the two nodes labeled FLORENCE in Figure 1.b are equivalent, we check the logical equivalence between the formulas approximating the statements 'Images of Florence in Tuscany' and 'Images of Florence in Italy' (individual concepts), given the formulas approximating the statements 'Florence is in Tuscany' and 'Florence is in Italy' (local axioms) and 'Tuscany is a region of Italy' (global axiom).

Algorithm 3 SEMANTIC−COMPARISON($\langle \phi, \alpha \rangle, \langle \psi, \beta \rangle, O$)

▷ *concept* $\langle \phi, \alpha \rangle$
▷ *concept* $\langle \psi, \beta \rangle$
▷ *ontology* O

VarDeclaratrions
formula γ
deductional-pair k[] ▷ *array of pairs* $\langle relation, formula \rangle$

1 $\gamma \leftarrow$ EXTRACT−GLOBAL−AXIOMS(ϕ, ψ, O);
2 $k \leftarrow$ BUILD−DEDUCTIONAL−FORMULAS$(\langle \phi, \alpha \rangle, \langle \psi, \beta \rangle, \gamma)$;
3 **for** each deductional-pair i in k
4 **if** SATISFIES$(\neg k[i].formula)$ **then**
5 **return** $k[i].relation$;
6 **else return** $Null$;

The three functions above constitute the *top-level algorithm*, i.e. the procedure followed to match generic structures labelled with natural language. All remaining functions (see below) are specific to the particular type of structures we need to match.

3 Semantic coordination of Hierarchical Classifications

Intuitively, a classification is a grouping of things into classes or categories. When categories are arranged into a hierarchical structure, we have a hierarchical classification. Prototypical examples of HCs are the web directories of many search engines, for example the Google™ Directory, the Yahoo!™ Directory, or the Looksmart™ web directory. In this section we show how to apply the general approach described in the previous section to the problem of coordinating HCs.

[2] See for example [3] for the use of synsets in a Lexicon.

[3] Note that a formula ϕ is valid exatcly in the case its negation $\neg\phi$ is not satisfi able.

The main algorithm is CTXMATCH, which is essentially the version of CTXMATCH where the input context contains a HC. It returns a relationship between the query node fl and the local node ll. Due to space limitation, we limited the descriptiom to the most relevant functions (see [6, 7] for a more detailed description). In the version of the algorithm presented here, we use WORDNET as a source of both lexical and domain knowledge. WORDNET could be replaced by another combination of a linguistic and domain knowledge resources[4].

HC–specific functions for BUILD-CTX-MEANING

BUILD-CTX-MEANING first needs to compute the focus of the label t and the synsets of each label in the structure. This is done by the functions DETERMINE−FOCUS and EXTRACT−SYNSET, respectively. We only give an intuitive description of these two functions.

Given a node s belonging to a structure S, DETERMINE−FOCUS has the task to reduce S to the minimal one without loosing the capability of rebuilding the meaning associated to the node s. For HC−CTXMATCH we define the focus F of a structure S given a node $s \in S$ as the smallest structure containing s and all its ancestors with their children.

EXTRACT−SYNSET associates to each node all the possible linguistic interpretations (synsets) provided by the Lexicon. In order to maximize the possibility of finding an entry into the Lexicon, we use bot a postagger and a lemmatizator over the labels.

The next function FILTER−SYNSET is applied to each node t of the focus. Its goal is to eliminate those senses associated to a node which seem to be incompatible with the meaning expressed by the node. To this end, it employs three heuristic rules, which take into account domain information provided by the ontology. This information concerns the relations between the senses associated to the node t and the senses associated to the other nodes in the focus.

Intuitively, the situation is as follows. Consider the node FLORENCE in the rightmost structure of Figure 1.b. The function EXTRACT−SYNSET associates to this node the two senses 'town in South Caroline' ('florence#1') and 'a city in central Italy' ('florence#2'). The structure also contains the node ITALY, which is an ancestor of FLORENCE. This node has a sense italy#3 (namely, 'Italy the european state'), for which the relation 'italy#3 hyperonym florence#2' holds, meaning that 'Florence is in Italy'. Therefore, the sense 'florence#1' can be discarded by exploiting knowledge about the sense of an ancestor node. We can then conclude that the term 'Florence' refers to the 'city in Italy' and not to the 'town in South Caroline'. The function ACCESS−ONTOLOGY allows us to discover relations between senses by traversing the ontology O (the WORDNET relations are reported in the left-hand side of Table 1).

Algorithm 4 FILTER−SYNSET$(T, O, SynS, t)$

▷ *structure T*
▷ *ontology O*
▷ *sense SynS[][] array of senses for the labels in T*
▷ *label t*

VarDeclaratrions
relation R_1, R_2, Rel_1, Rel_2 ▷ *initialized to Null*
sense $sense_{t1}, sense_{t2}, sense_y$

1 **for** each pair $sense_{t1} \neq sense_{t2}$ in $SynS[t]$ **do**
2 **for** each ancestor y of t in T **do**
3 **for** each $sense_y$ in $SynS[y]$ **do**
4 $R_1 \leftarrow$ ACCESS−ONTOLOGY$(sense_y, sense_{t1}, O)$;
5 **if** R_1 = 'hyperonymy' **then** $Rel_1 \leftarrow$ 'hyperonymy';
6 $R_2 \leftarrow$ ACCESS−ONTOLOGY$(sense_y, sense_{t2}, O)$;
7 **if** R_2 = 'hyperonymy' **then** $Rel_2 \leftarrow$ 'hyperonymy';
8 **if** $(Rel_1 = Null$ & $Rel_2 \neq Null)$ **then**
9 remove $sense_{t1}$ from $SynS[t]$;
10 $Rel_1 \leftarrow Rel_2 \leftarrow Null$;
11 **for** each pair $sense_{t1} \neq sense_{t2}$ in $SynS[t]$ **do**
12 **for** each descendant y of t in T **do**
13 **for** each $sense_y$ in $sense[y]$ **do**
14 $R_1 \leftarrow$ ACCESS−ONTOLOGY$(sense_y, sense_{t2}, O)$;
15 **if** R_1 = 'hyponymy' **then** $Rel_1 \leftarrow$ 'hyponymy';
16 $R_2 \leftarrow$ ACCESS−ONTOLOGY$(sense_y, sense_{t1}, O)$;
17 **if** R_2 = 'hyponymy' **then** $Rel_2 \leftarrow$ 'hyponymy';
18 **if** $(Rel_1 = Null$ & $Rel_2 \neq Null)$ **then**
19 remove $sense_{t1}$ from $SynS[t]$;
20 $Rel_1 = Rel_2 = Null$;
21 **for** each $sense_{t1}$ in $SynS[t]$ **do**
22 **for** each sibling y of t in T **do**
23 **for** each $sense_y$ in $SynS[y]$ **do**
24 $R_1 \leftarrow$ ACCESS−ONTOLOGY$(sense_{t1}, sense_y, O)$;
25 **if** R_1 = 'contradiction' **then** $Rel_1 \leftarrow$ 'contradiction';
26 **if** $(Rel_1 \neq Null)$ **then** remove $sense_{t1}$ from $SynS[t]$;
27 **return** SynS[t];

Lines 1–10 applies this heuristic to a sense s_n associated to a node t. Formally, it discards s_n if the following two conditions are satisfed: (i) no relation is found between this s_n and any sense associated to some ancestor, and (ii) some relation is found between a sense $s_m \neq s_n$ and some sense associated with an ancestor of t. Lines 11–20 do the same for descendants. Finally, lines 21–26 discard a sense if it is in 'contradiction' with some sense associated to a sibling of t.

The function INDIVIDUAL−CONCEPT builds a formula approximating the meaning expressed by a node t. This is done by combining the linguistic interpretation (the synsets $SynS$ associated to the nodes of the focus) with structural information (T) and domain knowledge (O), in input to the function. A critical choice is the formal language used to describe the meaning. Our implementation for HCs adopts propositional logic, whose primitive terms are the synsets of WORDNET associated to each node.

Lines 1–6 look for some ontological relation between the senses of the siblings and, if anyone is found, the interpretation of the node is refined. For example, imagine we have a node IMAGES with two children EUROPE and ITALY, and that the functions EXTRACT−SYNSET and FILTER−SYNSET associate to the nodes EUROPE and ITALY respectively the senses

[4] It's important to note that WORDNET is not a merged and shared structure, namely a kind of average of the structures to be matched (as in the GAV and LAV approaches). Indeed, it represents the result of linguistic mediation in centuries of use by human speakers. Using WORDNET instead of merged and shared structures, shifts the problem of sharing 'view of the world' to the more natural problem of 'sharing natural language'.

WORDNET relation	axiom
s#k synonym t#h	$s\#k \equiv t\#h$
s#k hyponym t#h	$s\#k \rightarrow t\#h$
s#k hypernym t#h	$t\#h \rightarrow s\#k$
s#k contradiction t#h	$\neg(t\#k \wedge s\#h)$

Table 1: WORDNET relations and their axioms.

europe#3 and italy#1. Since there exists an ontological relation 'europe#3 hyperonym italy#1' (Italy is in Europe) the meaning associated to node EUROPE is not longer europe#3, but it becomes europe#3 $\wedge\neg$ italy#1. In fact we imagine that a user wants to classify under node EUROPE images of Europe, and not images of Italy.

Algorithm 5 INDIVIDUAL−CONCEPT$(t, SynS, T, O)$

▷ *label t*
▷ *sense SynS[][]*
▷ *structure T*
▷ *ontology O*

VarDeclaratrions
formula $\eta = Null$
relation $R = Null, Rel = Null$
path P

1 **for** each $SynS[t][i]$ in $SynS[t][]$ **do**
2 **for** each sibling y of t in T **do**
3 **for** each $SynS[y][k]$ in $SynS[y][]$ **do**
4 $R \leftarrow$ ACCESS−ONTOLOGY$(SynS[t][i], SynS[y][k], O)$;
5 **if** R = 'hyperonymy' **then** $Rel \leftarrow$ 'hyperonymy';
6 **if** $(rel \neq Null)$ **then** replace $SynS[t][i]$ in $SynS[t][]$ with '$SynS[t][i] \wedge \neg SynS[y][k]$';
7 $P \leftarrow$ path from root to t in T; ▷ *Path from root to node t.*
8 $\eta \leftarrow \bigwedge_{e \in P} (\bigvee_i SynS[e][i])$;
9 **return** η;

Lines 7-8 compute the formula approximating the structural meaning of the concept t. This formula is the conjunction of the meanings associated to all of its ancestors (i.e., the path P). The meaning of a node is taken to be disjunction of all the (remaining) senses associated to the node. For example, if you consider the node FLORENCE in the rightmost structure of Figure 1.b, the function returns the formula (images#1 $\vee$ images#2) $\wedge$ italy#3 $\wedge$ florence#2, where (images#1 $\vee$ images#2) means that we are not able to discard anyone of the senses.

Function EXTRACT−LOCAL−AXIOMS extracts the local relevant axioms, i.e. the axioms relating concepts within a single structure. The idea is to rephrase the ontological relations between senses into logical relations. Consider again the senses florence#2 and italy#3 associated to the nodes FLORENCE and ITALY in Figure 1.b. The ontological knowledge tells us that 'italy#3 hyperoym florence#2'. This can be expressed by the axiom 'florence#2$\rightarrow$italy#3'. In HC-CTXMATCH, local axioms are built by translating WORDNET relations into formulas according to Table 1.

HC–specific functions for SEMANTIC−COMPARISON

The top–level function SEMANTIC−COMPARISON calculates the semantic relation between the formulas approximating the meaning of two nodes. In this section we describe the structural dependent functions called by this function: EXTRACT−GLOBAL−AXIOMS and BUILD−DEDUCTIONAL−FORMULAS.

EXTRACT−GLOBAL−AXIOMS works exactly as EXTRACT−LOCAL−AXIOMS. The only difference is that the axioms extracted express relations between concepts belonging to different structures. Consider for example that the two senses tuscany#1 and italy#3 have been associated respectively to nodes TUSCANY and ITALY in Figure 1.b. The ontological relation is 'italy#3 hyperonym tuscany#1', which can be expressed as 'tuscany#1 $\rightarrow$ italy#3'. The rules of translation from WORDNET senses to axioms are the same as for the function EXTRACT−LOCAL−AXIOMS.

In our approach, the problem of finding the relation between two nodes is encoded into a satisfiability problem. BUILD−DEDUCTIONAL−FORMULAS defines the satisfiability problems needed by defining (i) the set R of possible relations holding between concepts and, for each such relation $r \in R$, (ii) the formula which expresses the truth conditions for this relation. Clearly, the set R of possible relations depends on the intended use of the structures we want to map. For HC-CTXMATCH we choose the following set–theoretical relations: $\equiv$, $\subseteq$, $\supseteq$, $\perp$ ($\perp$ means that the two concepts are disjoint).

Relation	Formula
$\perp$	$(\alpha \wedge \beta \wedge \gamma) \rightarrow \neg(\phi \rightarrow \psi)\rangle$
$\equiv$	$(\alpha \wedge \beta \wedge \gamma) \rightarrow (\phi \equiv \psi)\rangle$
$\subseteq$	$(\alpha \wedge \beta \wedge \gamma) \rightarrow (\phi \rightarrow \psi)\rangle$
$\supseteq$	$(\alpha \wedge \beta \wedge \gamma) \rightarrow (\psi \rightarrow \phi)\rangle$

Table 2: The satisfiability problems for concepts $\langle\phi, \alpha\rangle$ and $\langle\psi, \beta\rangle$, with global axioms γ.

Table 2 reports the pairs $\langle$relation,formula$\rangle$ representing the satisfiability problems associated to each relation between concepts we consider, given two concepts $\langle\phi, \alpha\rangle$, $\langle\psi, \beta\rangle$, and the formula γ representing the global axioms. The result of this function is simply an array $k[]$ containing these pairs.

Consider the problem of checking whether FLORENCE in the right-hand structure in Figure 1.b is, say, equivalent to the node FLORENCE in the left-hand structure. Following are the concepts and axioms exteacted by the two strcutures:

$$\text{concept 1:} \quad \text{image\#1} \wedge \text{tuscany\#1} \wedge \text{florence\#2} \quad (1)$$
$$\text{local axiom 1:} \quad \text{florence\#2} \rightarrow \text{tuscany\#1} \quad (2)$$
$$\text{concept 2:} \quad \text{image\#1} \wedge \text{italy\#3} \wedge \text{florence\#2} \quad (3)$$
$$\text{local axiom 2:} \quad \text{florence\#2} \rightarrow \text{italy\#3} \quad (4)$$
$$\text{global axiom:} \quad \text{tuscany\#1} \rightarrow \text{italy\#3} \quad (5)$$

Checking equivalence then amounts to checking the following logical consequence $2 \wedge 4 \wedge 5 \models (1 \equiv 3)$. By the properties of propositional consequence, we can rephrase it as follows: $\models (2 \wedge 4 \wedge 5) \rightarrow (1 \equiv 3)$. It is easy to see that this latter formula is valid. So we can conclude that the relation holding between the two nodes FLORENCE is "equivalence", which is the intuitive one.

In particular, the function SATISFIES checks for the validity of a formula. In our implementation a standard SAT–solver is used for this task.

4 Testing the algorithm

In this section, we report from [5] some results of the first tests on CTXMATCH. The tests were performed on real HCs (i.e., pre-existing classifications used in real applications), and not on *ad hoc* HCs.

Matching Google with Yahoo!. We evaluated CTXMATCH over portions of Google™ and Yahoo!™ Directories looking for overlapping domains. The test was performed on the two sub-hierarchies 'Architecture' and 'Medicine' available in both Google™ and Yahoo!™. The results, measured in terms of precision and recall, are reported in the following table:

		Architecture		**Medicine**	
Relations		Pre.	Rec.	Pre.	Rec.
equivalence	$\xrightarrow{\equiv}$	.75	.08	.88	.09
less general than	$\xrightarrow{\subseteq}$	.84	.79	.86	.61
more general than	$\xrightarrow{\supseteq}$	.94	.38	.97	.35

We observe that the use of domain knowledge allowed us to discover non trivial mappings. For example, an inclusion mapping was found between `Architecture /History/Periods_and_Styles/Gothic/Gargoyles` and `Architecture/History/ Medieval` as a consequence of the relation between `Medieval` and `Gothic` provided by WORDNET. This kind of semantic mappings are very difficult to find using a keyword–based approach.

Product Re-classification. The second test was in the domain of e–commerce. In the framework of a collaboration with a worldwide telecommunication company, the matching algorithm was applied to re-classify the HC of the 'equipment and accessories' office (used to classify company suppliers) into UNSPSC[5] (version 5.0.2). We compare the results of the re-classification using CTXMATCH and the baseline matching process[6]:

	Baseline classification		**Matching classification**	
Total items	194	100%	194	100%
Rightly classified	75	39%	134	70%
Wrongly classified	91	50%	16	8%
Non classified	27	14%	42	22%

Given the 194 items re-classify, the baseline process found 1945 possible nodes, only 75 of which turned out to be correct. The baseline, a simple string-based matching method, is able to capture a certain number of re-classifications, but the percentage of error is quite high (50%) with respect to correctness (39%). With CTXMATCH the percentage of success is significantly higher (70%) and, even more relevant, the percentage of error is minimal (8%).

[5] UNSPSC (Universal Standard Products and Services Classification) is an open global coding system that classifies products and services. UNSPSC is extensively used around the world for electronic catalogs, search engines, e–procurement applications and accounting systems.

[6] The baseline has been performed by a simple keyword based matching which worked according to the following rule: for each item description (made up of one or more words) gives back the set of nodes, and their paths, which maximize the occurrences of the item words.

5 Conclusions and related work

In this paper we presented a new approach to semantic coordination in open and distributed environments. In particular we define in detail (i) a top algorithm (called CTXMATCH) for finding relations between structures labelled with natural language, and (ii) an implementation for finding set–theoretical relationships between nodes of hierarchical classifications (HC-CTXMATCH).

In [6, 7] we compare CTXMATCH with other proposed works, in particular with generic graph matching, CUPID [4], MOMIS [1] and GLUE [2]. We refer to this paper for related work.

References

[1] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.

[2] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of WWW-2002, Hawaii*, 2002.

[3] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, US, 1998.

[4] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.

[5] B. M. Magnini, L. Serafini, A. Doná, L. Gatti, C. Girardi, , and M. Speranza. Large–scale evaluation of context matching. Technical Report 0301–07, ITC–IRST, Trento, Italy, 2003.

[6] P. Bouquet B. Magnini, L. Serafini, and S. Zanobini. A SAT–based algorithm for context matching. In *Proc. of the 4th International and Interdisciplinary COnference on Modeling and Using Context (CONTEXT-03). Stanford University (CA), June 23-25, 2003*, volume 2680 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2003.

[7] P. Bouquet L. Serafini and S. Zanobini. Semantic coordination: a new approach and an application. In *Proc. of the 2nd International Semantic Web Conference (ISWO'03). Sanibel Islands, Florida, USA*, October 2003.