Introducing Reasoning into an Industrial Knowledge Management Tool

Article *in* Applied Intelligence · December 2009

DOI: 10.1007/s10489-007-0103-x · Source: dx.doi.org

CITATIONS
13

READS
166

3 authors:

Olivier Carloni

14 PUBLICATIONS **50** CITATIONS

SEE PROFILE

Michel Leclere

Université de Montpellier

95 PUBLICATIONS **928** CITATIONS

SEE PROFILE

Marie-Laure Mugnier

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LI…

138 PUBLICATIONS **2,327** CITATIONS

SEE PROFILE

Introducing Reasoning into an Industrial Knowledge Management Tool

Olivier Carloni[1,2], Michel Leclère[1], and Marie-Laure Mugnier[1]

[1]LIRMM, CNRS - Université Montpellier 2, 161 rue Ada, F-34392 Montpellier cedex 5 - France, {leclere, mugnier}@lirmm.fr

[2] Mondeca„ 35 boulevard de Strasbourg, F-75010 Paris - France, http://www.mondeca.com, olivier.carloni@gmail.com

Abstract

This paper is devoted to an industrial case study focused on the issue of how to enhance an existing knowledge management tool (ITM) with reasoning capabilities, by introducing a semantic query mechanism as well as validation and inference services. ITM knowledge representation language is based on topic maps. We show that these topic maps (and especially those describing the domain ontology and annotation base) can be naturally mapped to the $\mathcal{SG}$ family, a sublanguage of conceptual graphs. This mapping equips ITM with a reasoning service. We finally present a media monitoring system benefiting from this transformation and combining ITM with the conceptual graph engine CoGITaNT.

1 Introduction

In the last decade, there has been a massive flow of knowledge into organizations to meet requirements of (1) capitalization of knowledge and know-how of their members in order to build up a corporate memory (i.e. a disembodied representation of their expertise), and (2) implementation of their information systems at the "knowledge level", i.e. exploiting semantics of exchanged information. This evolution from information systems to knowledge systems has given rise to true knowledge engineering, which aims at eliciting knowledge (e.g. the semantics of exchanged information content). This elicitation is performed by way of a knowledge representation language and is controlled by ontologies (domain ontologies for conceptual vocabulary and representation ontologies for language constructs) which are semantic referentials that delineate the meaning of symbols used by these languages. This issue arises again within the framework of the semantic web, which endeavors to describe the content of web resources in order to facilitate their access and use.

Mondeca is a software publisher that is developing ITM (Intelligent Topic Manager), a knowledge management tool based on these principles. The core of this software is a three-level knowledge base (see. Figure 1):

- the highest level is a meta-model of all ITM knowledge bases. It consists of a representation ontology reflexively specifying the semantics of all representations used by ITM. It is common to all customers;
- the intermediate level is composed of models that specify the vocabulary used to describe customer data managed by ITM; it generally comprises a domain ontology describing the conceptual vocabulary used to annotate the content of the customer's resources, some (representation) ontologies defining primitives related to specific data structures enabling ITM services (primitives for representing the thesaurus or the logical organization of documents);
- the lowest level contains the instances: annotations describing the content of information managed by ITM (data, documents), terminological resources (thesaurus) used to index this information, description of the logical organization of documents, etc. This level is managed by the customer.

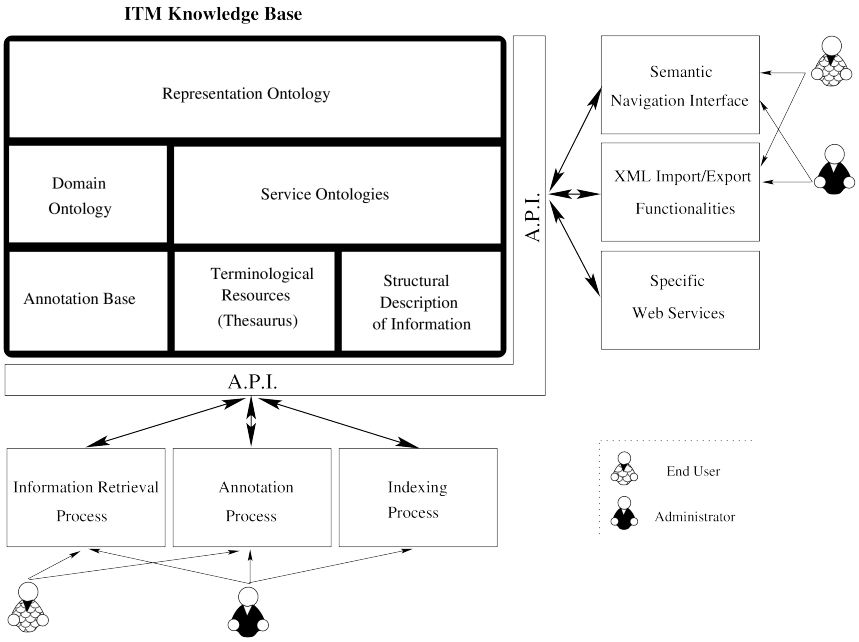


Figure 1: Workspaces and modules in ITM

ITM provides many services based on its knowledge base: indexing of documents from the thesaurus, creation of annotations controlled by the domain ontology, semi-automatic building of document annotations, semantic navigation through the annotation base, searching the annotation base by a querying mechanism. However, although it performs complex knowledge management tasks, ITM does not offer reasoning services. Reasoning capabilites are typically needed when one wants to ensure the consistency of the knowledge base after adding a new annotation, to enrich the annotations by implicit knowledge, or to provide advanced querying mechanisms. Enabling different kinds of reasoning on the knowledge represented in ITM knowledge bases is the focus of a collaboration project between the LIRMM and Mondeca.

The representation language in ITM is based on the Topic Map (TM) paradigm in which knowledge is described by *topics*, representing entities of the modeled domain, and *associations* connecting topics and identifying the *role* played by each topic in the association [HP02]. A TM can be seen as a labeled graph (or hypergraph), as described in [AMRV02] or [BN01], where the vertices come from topics and associations and an edge between a topic and an association indicates the role played by the topic in this association (cf. section 2). The TM paradigm has not been provided with formal semantics.

Formal semantics of representations used by ITM thus have to be defined in order to specify the desired kinds of reasoning. The closeness of the TM language to the conceptual graph model [Sow84] and especially to the $\mathcal{SG}$-family [BM02] – a TM is naturally transformed into a simple conceptual graph (SG) – led us to map ITM representations into this family. By assimilating TMs to SGs it is possible: to provide ITM with a formal semantics since the $\mathcal{SG}$ family is logically based; to incorporate inference rules as well as constraints in ITM bases since the $\mathcal{SG}$ family manages this kind of knowledge; to use reasoning schemes of the $\mathcal{SG}$ family, which are graph-based (thus operate directly on the knowledge defined by the user) while being sound and complete with respect to the logical deduction; to benefit from efficient algorithms developed for these graph operations and based on combinatorial techniques; and finally to tap the GPL library CoGITaNT, which is dedicated to developing applications based on conceptual graphs and implementing the $\mathcal{SG}$ family [Gen97].

The two main contributions of this paper are the detailed definition of a mapping from TM to CG, and an industrial case study about how to equip a complex knowledge management system with reasoning capabilities, while maintaining "upward compatibility".

There are two possible ways of exploiting the mapping, either directly in ITM or in an external service. First, as the mapping is reversible, it allows transfer of the reasoning mechanisms of the target language to topic maps. ITM could thus be directly provided with a logically based query mechanism. With some syntactic extensions, constraints and inference rules could also be represented in ITM. Secondly, note that ITM is a large data repository with classical management functions enhanced with some modules performing specific tasks using only specific parts of its repository. For instance, the module dedicated to information retrieval uses the thesaurus part of the repository. A reasoning module

concerns only the parts of the repository that contain data at the user's knowledge level. Currently, only parts of the domain ontology and annotation base could be exploited by such a module. Furthermore, the ITM repository aims at keeping any added knowledge regardless of its status w.r.t. a treatment performed by a specific module. Especially, it should be possible to keep in the ITM repository some pieces of knowledge that are considered to be inconsistent by the reasoning module (because, for instance, they are used by another module to evaluate the quality of the semi-automatic annotation process). The preferred solution is thus to transfer (globally or incrementally) adequate parts from ITM (after mapping it into CG) into an external CG inferential service. This service maintains a CG copy of a consistent piece of the transferred part and can thus answer queries, validate new annotations, or enrich annotations from implicit knowledge. In this paper, we present the architecture of this system and its use in a media monitoring application.

The sequel of this paper is organized as follows. In section 2 the TM language and the three-level architecture of ITM are presented. Section 3 introduces the $\mathcal{SG}$ family, a sublanguage of conceptual graphs. Section 4 is devoted to the mapping, the kind of reasoning it enables, and the gains for ITM. An application to media monitoring illustrating a real-world use of this mapping is presented in section 5, as well as related works. Finally, Section 6 outlines future extensions.

2 ITM - A Knowledge Management Tool

ITM uses the standard Topic Map language (TM) to represent and store all kinds of knowledge. This language allows to structure a set of resources (adressable documents) with topics that gather resources sharing common properties, and associations that define relationships between topics [ISO00]. As it possesses an XML syntax [Top01], it is a candidate for being used within the framework of the semantic web. Unlike its direct rivals, RDF, RDFS and the OWL family, it has not been equipped with formal semantics and, as far as we know, no reasoning mechanism has been proposed for it.

2.1 Topic Maps

A TM is a network of topics linked by associations. Associations are typed and characterize the role played by each linked topic. Topics can be identified by *names* and characterized by *occurrences* (kinds of attribute-value pairs). Two specific binary associations are standardized for any TM: the *class-instance* association and the *superclass-subclass* association [Top01]. To facilitate the reading of TM in this paper, we represent the *class-instance* association as a topic label (see the function *type* below). We focus in this section on the part of the TM standard used by ITM.

We formally define a TM as a bipartite labeled graph
$tm = (T, A, E, type, name)$ where T is the set of topic nodes, A is the set of association nodes and $E \subseteq A \times T$ is the set of edges. *type* and *name* are labeling

functions from $T \cup A \cup E$ into L, where L is the set of labels. *type* assigns to each node and edge a label in L, which denotes the class for a topic, the type for an association and the role name for an edge. The partial injective mapping *name* identifies some topics with a name. See the TM in Figure 2 for instance: the topic of class `Company` and name `Oracle` plays the role `acquiring` in an association of type `Acquisition`, which links it to two other topics.

Each topic can be further described by a set of occurrences expressing relevant information about the topic. Each occurrence is composed of a value v of L_V, a data type d of D (called *physical type* in ITM) and the name l of the topic that types the occurrence (called *logical type* in ITM). We denote by *occ* the mapping from T into $2^{L_V \times D \times L}$ that assigns to a topic its occurrence set, where D is the data type set and L_V is the set of values for types in D. For example, the topic with name `Oracle` in Figure 2 has three occurrences: a value of data type `Integer` as `Capital` and the values `www.oracle.com` and `www.oracle.fr` of data type `URL` as `web site`. These occurrences can be interpreted as the capital of the company Oracle and the web addresses for further information.

Let us point out again that some elements of the TM standard are not formalized here because they are not used by ITM. However, previous definition as well as the mapping from TM to CG can be extended to include them (see part 4).

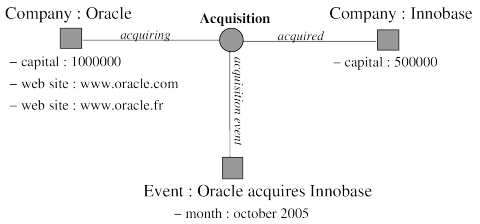
Notations. $Name$ is the extension of the mapping *name* to a TM: it assigns to a TM the subset of L labels which are images of its topics by the mapping *name*. Ref is the mapping which assigns to a TM the subset of all L labels returned by the functions *type* and *occ*.

A TM tm is said to be *self-content* if each label returned by *type* and *occ* is assigned to a topic by *name*, that is $Ref(tm) \subseteq Name(tm)$.

2.2 ITM Knowledge Bases

An ITM knowledge base is a self-content TM structured in three levels by the *Class-instance* association (as illustrated by Figure 3):

- the *meta* level describes the reflexive meta-model of ITM knowledge;
- the *model* level is composed of models which are described using the primitives of the meta level. This level generally comprises a domain ontology that specifies the vocabulary used to represent the content of information managed by the base, as well as several "service" ontologies that specify the semantics of representations enabling ITM services (primitives for representing the thesaurus or the logical organization of documents for instance);
- finally, the *instance* level contains the instances of the base, which are "controlled" by a model of the intermediate level: annotations describing the content of information managed by the base, terminological resources



$T = \{t_1, t_2, t_3\}$ $A = \{a_1\}$ $E = \{e_1 = a_1 t_1, e_2 = a_1 t_2, e_3 = a_1 t_3\}$

$L_V = \{v_1 =$ "1000000", $v_2 =$ "500000", $v_3 =$ "$www.oracle.fr$", $v_4 =$ "$www.oracle.com$",

$v_5 =$ "$october\,2005$"$\}$

$L = \{l_1 =$ "$Oracle$", $l_2 =$ "$Innobase$", $l_3 =$ "$Oracle\,acquired\,Innobase$",

$l_4 =$ "$Acquisition$", $l_5 =$ "$Company$", $l_6 =$ "$Event$", $l_7 =$ "$capital$", $l_8 =$ "$web\,site$",

$l_9 =$ "$month$", $l_{10} =$ "$acquiring$", $l_{11} =$ "$acquired$", $l_{12} =$ "$acquisition\,event$"$\}$

$D = \{d_1 =$ "$Integer$", $d_2 =$ "$String$", $d_3 =$ "$Date$", $d_4 =$ "URL"$\}$

$name = \{(t_1, l_1), (t_2, l_2), (t_3, l_3)\}$

$type = \{(t_1, l_5), (t_2, l_5), (t_3, l_6), (a_1, l_4), (e_1, l_{10}), (e_2, l_{11}), (e_3, l_{12})\}$

$occ = \{\ (t_1, \{(v_1, d_1, l_7), (v_3, d_4, l_8), (v_4, d_4, l_8), \}\ )$

$(t_2, \{(v_2, d_1, l_7)\}\ )$

$(t_3, \{(v_5, d_3, l_9)\}\ )\ \}$

Figure 2: A topic map and its formal definition.

(thesaurus) used to index this information, description of the logical organization of documents, etc.

The model and instance levels are clustered into workspaces dedicated to a specific kind of knowledge (annotations, thesaurus, logical organization of documents...). For each workspace w, there are a TM of the model level, tm_w^{model}, which defines the vocabulary usable to describe the knowledge of the instance level in this workspace, and the associated TM of the instance level.

The following constraints are fulfilled:

- the meta level TM, tm^{meta}, is self-content;
- all topics of the meta and model levels are identified by a name (i.e. the *name* function is total for these TMs);
- the references used in the TMs comply with the level hierarchy (the type labels of the model – resp. instance – level are included in the labels assigned by the $Name$ function of the meta – resp. model – level) and are internal to the workspaces[1]. I.e. for each workspace w, $Ref(tm_w^{model}) \subseteq Name(tm^{meta})$ and $Ref(tm_w^{instance}) \subseteq Name(tm_w^{model})$.

Figure 3 shows part of a knowledge base devoted to competitive intelligence. In order to represent cardinality constraints on roles and occurrences, ITM assigns occurrences to associations and not only topics. The model and instance levels are extracted from the workspace devoted to semantic annotations. They respectively describe a part of the ontology and of the knowledge about companies. The meta level TM, common to all knowledge bases, defines modeling primitives. These primitives are interpreted by ITM and provide the knowledge bases with the following operational semantics:

- an association a of type `Allowed association type` linking three topics t_c, t_a and t_r specifies that every topic of class t_c is allowed to play a role t_r in an association of type t_a. Moreover, an occurrence with type `maximum cardinality` (resp. `minimum cardinality`) and integer value n, associated with a, specifies that an association of type t_a cannot have more (resp. less) than n roles of type t_r. See for instance the model level in Figure 3: an association of type `Allowed association type` allows a topic of class `Company` to play the role `acquiring` in an association of type `Acquisition` at instance level; and the cardinality occurrence in the `Allowed association type` association specifies that the `Acquisition` association cannot have more than one `acquiring` role at instance level;
- an association a of type `Allowed occurrence type` linking t_c and t_o specifies that a topic of class t_c may have an occurrence with logical type t_o. Moreover, an occurrence with type `maximum cardinality` (resp. `minimum`

[1] Let us note however that ITM provides pointers, which are external to the TM language, to relate topics from different workspaces. For instance, a specific term of the thesaurus can be assigned to a resource of the annotation space.

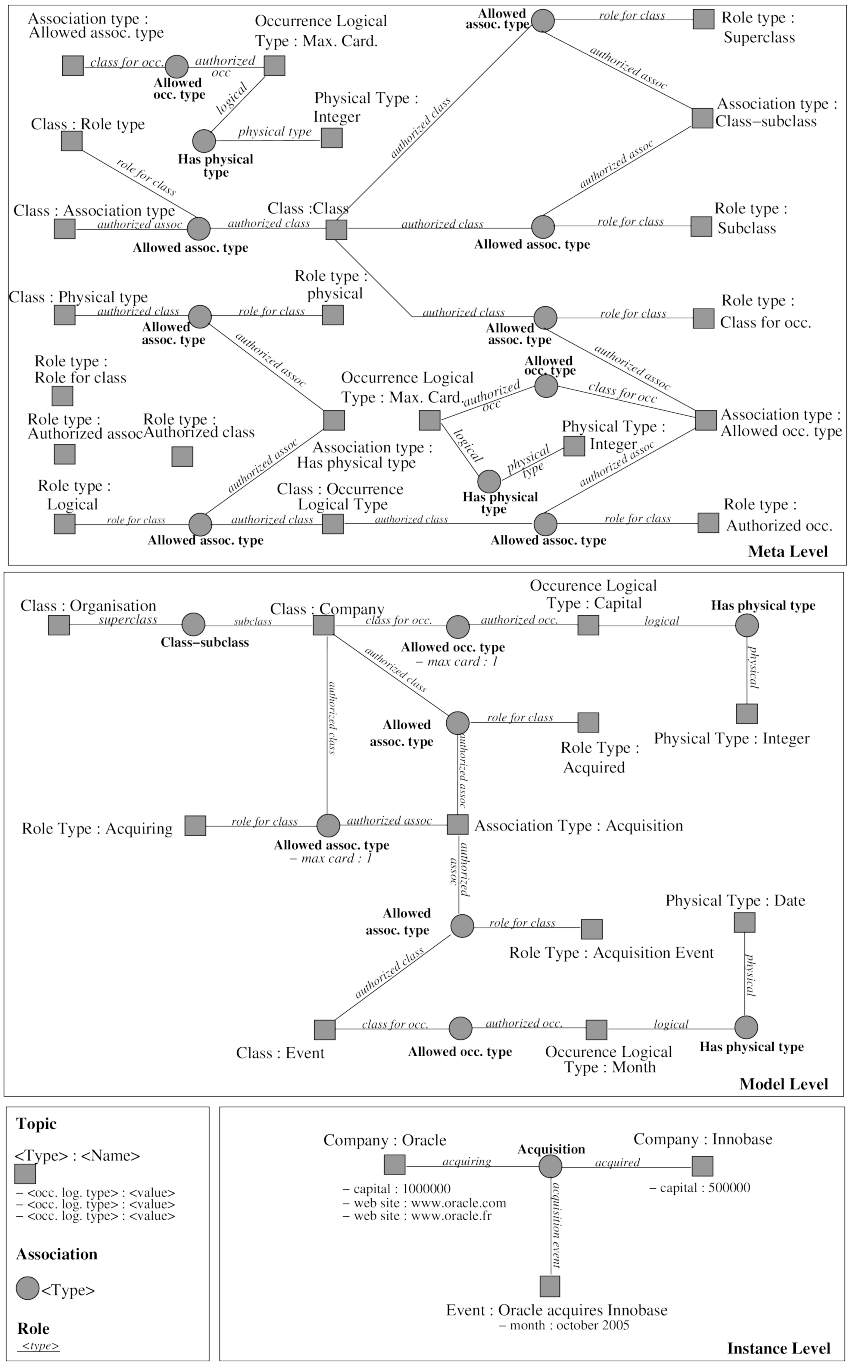


Figure 3: A (partial) knowledge base with its meta, model and instance levels.

`cardinality`) and integer value n, associated with a, specifies that a topic of class t_c cannot have more (resp. less) than n occurrences of type t_o. See for instance the model level in Figure 3: the class `Company` is linked to the logical occurrence type `Capital` by an association `Allowed occurrence type` with a `maximum cardinality` equals to one. This means that any instance of `Company` may possess at most one instance of `Capital`;

- an association of type `Has physical type` linking t_o and t_p specifies that all occurrences with logical type t_o have the physical type t_p. ITM uses the physical type to correctly interpret the value of the occurrence (as an integer, a date, a string or a pointer).
- the association of type `Class-subclass` defines a partial order on topics; the `Allowed association type` and `Allowed occurrence type` associated with classes of topics and their cardinalities are inherited according to this order [2].

3 The $\mathcal{SG}$ Family

Conceptual graphs [Sow84], and especially the $\mathcal{SG}$ family [BM02], were chosen as the knowledge representation and reasoning formalism upon which reasoning tasks are built. Several arguments warrant this choice:

- there is an obvious closeness between topic maps and simple conceptual graphs;
- conceptual graphs have a formal semantics in first-order-logic;
- the $\mathcal{SG}$ family is able to represent different kinds of knowledge : assertions (or facts) as simple conceptual graphs, inference rules and constraints;
- reasoning in the $\mathcal{SG}$ family is based on graph operations. Reasoning tasks operate directly on the knowledge defined by the user and not on their translation into logical formulas. This makes it possible to explain reasoning to the end-user because it can be visualized in a natural way on the pieces of knowledge he/she is familiar with;
- the algorithms that implement the reasoning tasks use combinatorial techniques (namely from graph theory and constraint networks) which make them particularly efficient;
- the $\mathcal{SG}$ family is implemented in the freeware CoGITaNT, an API dedicated to developing applications based on conceptual graphs [Gen97].

This section rather informally presents the components of the $\mathcal{SG}$ family we use in this paper. For further details the reader is referred to [CM92] about simple conceptual graphs and to [BM02] for a study of the whole $\mathcal{SG}$ family.

[2] Ontologies are built and checked using Protégé [Inf07], we do not expand upon this point in this paper.

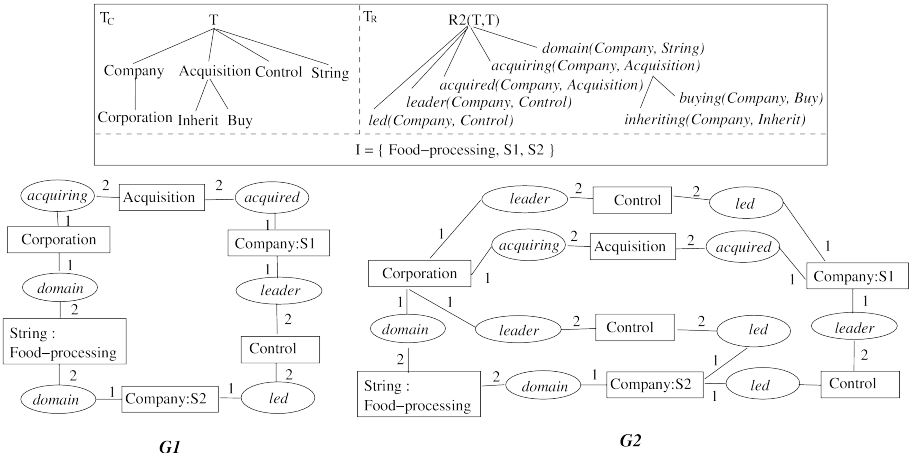


Figure 4: A support and two SGs on this support. The generic marker is not represented.

3.1 Simple Conceptual Graphs

A *simple conceptual graph (SG)* is a bipartite labeled graph: one class of nodes, called *concept* nodes, represents entities and the other, called *relation* nodes represents relationships between these entities or properties of them. E.g. the graph G_1 in Figure 4 can be seen as expressing the following knowledge : "A corporation specialized in food-processing acquires the company $S1$; $S1$ controls the company $S2$, which is specialized in food-processing as well". Node labels come from a vocabulary called a *support*, which can be more or less rich. The support considered here is a structure $S = (T_C,\ T_R,\ I,\ \sigma)$, where T_C is a set of concept types and T_R is a set of relations with any arity (the arity is the number of arguments of the relation). T_C and T_R are partially ordered. The partial order represents a specialization relation ($t' \leq t$ is read as "t' is a specialization of t"). I is a set of individual markers. The mapping σ assigns to each relation a signature specifying its arity and the maximal type for each of its arguments. The support can be seen as a rudimentary ontology and SGs encode assertions called *facts*: they assert the existence of entities and relations among these entities. Figure 4 shows an example of support in which the concept type `Corporation` and the relation `buying` are respectively a specialization of `Company` and `acquiring`. The *signature* of the `acquiring` relation is `(Company, Acquisition)`.

In a SG a concept node is labeled by a couple $t : m$ where t is a concept type and m is a marker. If the node represents an unspecified entity its marker is the generic marker, denoted by $*$, and the node is called a *generic* node, otherwise its marker is an element of I, and the node is called an *individual* node. E.g. in the SG G_1 of Figure 4, the node `[Company:S1]` refers to "the" company $S1$, while the node `[Corporation:*]` refers to "a" corporation. A relation node is

labeled by a relation r and, if n is the arity of r, it is incidental to n totally ordered edges. Classically, concept nodes are drawn as rectangles and relation nodes as ovals and the order on edges incidental to a *n-ary* relation node are numbered from 1 to n. A SG is denoted by $G = (C_G, R_G, E_G, l_G)$ where C_G and R_G are respectively the concept and relation node sets, E_G is the set of edges and l_G is the mapping labeling nodes and edges.

The fundamental notion for comparing SGs is a mapping from a SG to another called a *projection*. In graph terms it is a graph homomorphism. Intuitively a projection from G to H proves that the knowledge represented by G is included in (or implied by) the knowledge represented by H. More specifically, a projection π from G to H is a mapping from C_G to C_H and from R_G to R_H which preserves edges (if there is an edge numbered i between r and c in G then there is an edge numbered i between $\Pi(r)$ and $\Pi(c)$ in H) and may specialize labels. See Figure 4: there is a projection from G_1 to G_2 (which is particular since it is injective and does not change labels). See also the SG Q in Figure 5 (where '?' has to be replaced by $*$) and the SG G_1 and G_2 in Figure 4: there is a projection from Q to G_2 (knowing that `Corporation < Company`) but not from Q to G_1.

Conceptual graphs are provided with a semantics in first-order-logic, defined by a mapping classically denoted by Φ. Concept types are translated into unary predicates and relations into predicates of the same arity. Individual markers become constants. To a support S is assigned a set of formulas $\Phi(S)$ which translates the partial orders on concept types and relations: if t and t' are concept types, with $t' < t$, one has the formula $\forall x(t'(x) \rightarrow t(x))$; similarly, if r and r' are n-ary relations, with $r' < r$, one has the formula $\forall x_1 \ldots x_n(r'(x_1 \ldots x_n) \rightarrow r(x_1 \ldots x_n))$. A SG G is naturally translated into a positive, conjunctive and existentially closed formula $\Phi(G)$, with each concept node being translated into a variable or a constant: a new variable if it is a generic node, and otherwise the constant assigned to its individual marker. The formula assigned to the graph G_1 in Figure 4 is for instance: $\Phi(G_1) = \exists x \exists y \exists z Corporation(x) \wedge Acquisition(y) \wedge Control(z) \wedge Company(S1) \wedge Company(S2) \wedge String(Food\text{-}processing) \wedge acquiring(x, y) \wedge acquired(S1, y) \wedge domain(x, Food\text{-}processing) \wedge domain(S2, Food\text{-}processing) \wedge leader(S1, z) \wedge led(S2, z)$.

The following result of *projection soundness and completeness* is fundamental since it establishes the equivalence between projection and deduction on the formulas assigned to SGs: given two SGs G and H on a support S, there is a projection from G to H if and only if $\Phi(G)$ can be deduced from $\Phi(H)$ and $\Phi(S)$ ([Sow84] for the soundness and [CM92] for the completeness). Completeness is obtained up to a condition on H: H has to be in a normal form, which imposes that any individual marker appears at most once in it (in other words, a specific entity cannot be represented by two nodes). A variant of projection can be used to achieve completeness without this restriction [CM04].

Let us now consider a knowledge base K composed of a support and a set of facts F. Projection yields a basic mechanism to query this base. In its simplest form, a query Q is itself a SG and K answers Q if there is a projection from Q to F. Every such projection can be seen as an answer to Q. More generally,

a query may include distinguished (generic) concept nodes. In this case the answer is restricted to these nodes. Classically, these nodes are marked by a '?' symbol (see Figure 5: Q asks for "*couples of companies* of the same domain such that the first company controls the second company"). This kind of queries is equivalent in expressive power to conjunctive queries in databases.

3.2 Representing Difference

Two concept nodes in a SG might not necessarily represent distinct entities. In order to express difference, a special binary relation can be added over the concept node set of a SG. This relation, denoted by dif, is antireflexive and symmetrical and its logical translation is $\neq$ (i.e. $\neg =$). dif is pictured by difference links (see for instance the graph C_1^- in Figure 6: there is a difference link between the nodes with label `Integer`). We make the classical *unique name assumption*, thus every pair of concept nodes with different individual markers belongs to the dif relation.

Difference is a form of negation, which can be interpreted in several ways. In classical logic the law of the excluded middle intervenes. Applied to equality and difference, this law says that "given two entities, either they are equal or they are different". It leads to a case reasoning, which makes deduction more complex. Some logics, as the intuitionistic logic, do not accept the law of the excluded middle; in these logics difference can be processed without complexity overhead. A third way of dealing with difference is to make the closed-world assumption about facts. In this case, all distinct concept nodes in SG representing facts are assumed to represent different entities, thus the law of the excluded middle needs not to be applied. This assumption is made about topics in ITM bases.

Let us now see how the SG reasoning mechanism can be extended to take difference into account, according to the several ways of understanding negation. First, projection (say π from G to H) is extended to preserve difference, which is expressed by the following condition: if $\{c_1, c_2\} \in dif_G$ then $\{\pi(c_1), \pi(c_2)\} \in dif_H$. If the law of the excluded middle is not considered, projection is complete with respect to deduction. This is the case with intuitionistic deduction or with the closed-world assumption (made in ITM bases). A single projection check is not complete anymore for classical deduction: to obtain completeness, we have to do an exponential number of projection checks (in the size of H). For further details see [LM06].

SGs are used to represent queries and facts but also as building blocks for more complex kinds of knowledge, namely *inference rules* and *constraints*.

3.3 Rules

An inference rule expresses implicit knowledge of form "if *hypothesis* then *conclusion*", where hypothesis and conclusion are both SGs. In Figure 5 two rules, denoted by R_1 and R_2, are represented. Each dotted line connects a node in the hypothesis and a node in the conclusion; these nodes are called *connection* nodes. The nodes of the conclusion that are not connection nodes are colored

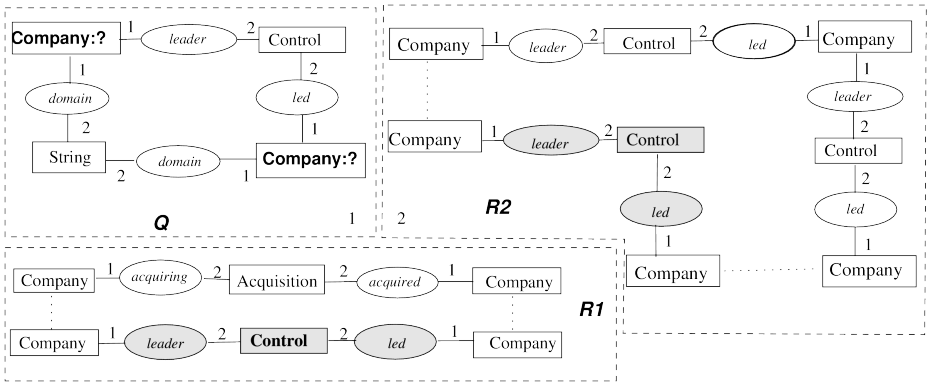


Figure 5: A query and two rules

in gray. The rules express properties of the concept type "control": acquiring a company leads to control it ($R1$: "if a company x acquires a company y then x controls y") and exercising control is transitive ($R2$).

Let R be a rule and F be a SG. R can be applied to F if there is a projection from its hypothesis to F. Applying R to F according to such a projection π consists in "attaching" to F the conclusion of R by merging each connection node in the conclusion with the image by π of the corresponding connection node in the hypothesis. See Figures 5 and 4: if $R1$ is applied to G_1 (using `Corporation` $<$ `Company`) and $R2$ is applied to the resulting SG, one obtains G_2.

The logical formula assigned to a rule R is of form $\Phi(R) = \forall x_1 \ldots x_p\ ((hyp) \rightarrow \exists y_1 \ldots y_q\ (conc))$, where: *hyp* et *conc* are conjunctions of atoms respectively translating the hypothesis and the conclusion, with the same variable being assigned to corresponding connection nodes; $x_1 \ldots x_p$ are the variables assigned to the concept nodes of the hypothesis; $y_1 \ldots y_q$ are the variables assigned to the concept nodes of the conclusion except for the connection nodes. With the rule R_1 in Figure 5, we obtain $\Phi(R_1) = \forall x \forall y \forall z\ (Company(x) \wedge Company(y) \wedge Acquisition(z) \wedge acquiring(x,z) \wedge acquired(z,y) \rightarrow \exists t(Control(t) \wedge leader(x,t) \wedge led(t,y)))$.

Since a knowledge base is now composed of a support, a set of facts (say F) and a set of rules (say $\mathcal{R}$), the query mechanism has to take implicit knowledge coded in rules into account. The knowledge base answers a query Q if a SG F' can be derived from F using the rules of $\mathcal{R}$ such that Q can be projected to F'. Let us consider again Figures 4 and 5: the base containing the fact G_1 and the rules $R1$ and $R2$ answers Q; indeed Q can be projected to G_2, which is derived from G_1 using $R1$ and $R2$. Sound and complete forward and backward chaining schemes have been defined [SM96].

Finally, note that the query problem is not decidable anymore for general rules (i.e. there is no algorithm solving this problem in finite time for all data). Decidability can be obtained with specific forms of rules, for instance rules that

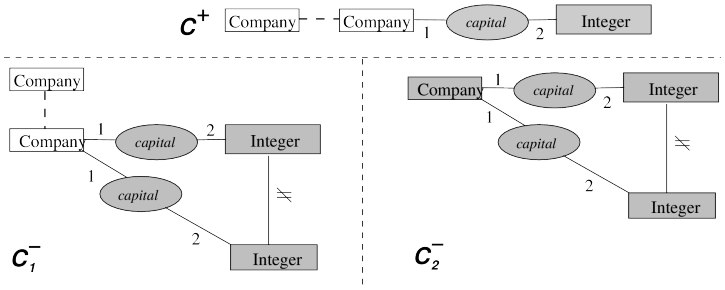


Figure 6: Positive and negative constraints

allow to "saturate" the fact base (a base is saturated if no rule application adding new information can be performed anymore); then, answering a query consists in projecting it to the saturated base. An alternative way of doing consists in using backward chaining instead of forward chaining.

3.4 Constraints

Constraints can be either *positive*, expressing knowledge of form that "if *condition* is found, so must *obligation*", or *negative*, expressing knowledge of form "if *condition* is found, *interdiction* must not". They have the same syntactical form as rules but a different semantics: the first SG defines a condition, the second SG defines a mandatory or forbidden pattern. Figure 6 shows two contraints: the positive constraint C^+ says that "a company must have at least one capital" ("if a company is found, a capital for this company must be found too"); the negative constraint C_1^- says that "a company cannot have two (different) capitals" (" if a company is found, one must not find two different capitals for this company"). C_1^- can be seen as a functional constraint ("a company has at most one capital"). Notice that these examples do not illustrate the expressivity of constraints, as both parts of a constraint can be any SG. But they reflect a typical use of constraints in ITM context, which consists in translating cardinality constraints.

A SG G satisfies a positive constraint C^+ if *each* projection from the condition part of C^+ to G can be extended to a projection of its mandatory part (with corresponding connection nodes having the same image in G). G satisfies a negative constraint C^- if *no* projection from the condition part of C^- to G can be extended to a projection of its forbidden part. Note that if C^- is transformed into a SG by pairwise merging corresponding connection points, the previous definition can be reformulated as: G satisfies C^- if there is no projection from C^- as a SG into G (see Figure 6: C_2^- is equivalent to C_1^-).

A SG G is said to be consistent with respect to a set of constraints $\mathcal{C}$ if it satisfies all constraints of $\mathcal{C}$. When the knowledge base is composed not only of facts but also of rules, these rules have to be taken into account for consistency. The query problem is defined only for consistent bases. A base K composed of

a support, a set of facts F, a set of rules $\mathcal{R}$ and a set of negative constraints $\mathcal{C}$ is said to be consistent if all SG derivable from F by $\mathcal{R}$ is consistent with respect to $\mathcal{C}$. With positive constraints, consistency is a more complex notion that we do not present here (see [BM02]).

4 A Conceptual Graph Inference Engine for ITM

To provide ITM with full query and inference capabilities and consistency control, a query-answering service and a validation/enrichment service have been designed in the $\mathcal{SG}$ family. These services have been implemented using the CoGITaNT library [Gen97]. This section details the mapping from TM to SG which allows ITM to benefit from these $\mathcal{SG}$ services, and presents an overview of these services.

4.1 Translation of an ITM Base into a SG Base

As all knowledge is represented by topic maps in ITM, several kinds of mappings were possible:

1. translate the meta model of ITM (actually the TM formalism) into a support and the lower levels into SGs representing facts. It is the most general way of processing ITM knowledge but it does not allow to exploit the specific features of conceptual graphs (for instance, with this translation, the partial order class-subclass on topics would be represented by relation nodes in SGs, thus it could not be processed automatically by label node comparisons in reasoning);

2. capture the intuitive and operational semantics of TMs by conceptual graph primitives (for instance, the class-subclass association of the meta-model is represented by the native hierarchy of the support). The model level is translated into a support and the instance level into a SG;

3. apply translation 2 but only to the workspace dedicated to annotations, considering that the other workspaces have nothing to do with knowledge representation: indeed, the thesaurus is devoted to linguistic aspects and the other spaces to document organization. This way of doing was chosen.

Let us denote by f the mapping from an ITM base to the $\mathcal{SG}$ family. f essentially encodes the domain ontology into a support and the annotation base into a SG (or a set of SGs). It is illustrated by Figure 7. Moreover, part of the operational semantics of the meta level can be translated into constraints and rules, as explained hereafter.

4.1.1 Building a Support from the Domain Ontology

Let us recall that the domain ontology is a TM of the model level.

1. Three concept types C (for topic classes), AT (association types) and OPT (occurrence physical types) are created as subtypes of a universal type $\top$. To each topic t with *type* `Class`, `Association type` or `Physical type`, f assigns a concept type with label $name(t)$ subtype of C, AT or OPT respectively. The partial order on T_C is induced by the `Class-subclass` association. For instance, the topics `Organisation` and `Company` of the model level are translated into concept types with the partial order $Company \leq Organisation \leq C$.

2. Two relations RT (for role types) and OLT (for occurrence logical types), with signature $\sigma(RT) = (C, AT)$ and $\sigma(OLT) = (C, OPT)$, are created. Each topic t with type `Role type` involved in a ternary association of type `Allowed association type` connecting t to topics c and a becomes a binary relation with label $name(t)$, subrelation of RT; its signature is $\sigma(name(t)) = (f(c), f(a))$. Recall that each topic t with type `Occurrence logical type` is linked by an association with type `Has physical type` to a topic *opt* and by an association with type `Allowed occurrence type` to a topic c; t becomes a relation, subrelation of OLT, with label $name(t)$ and signature $\sigma(name(t)) = (f(c), f(opt))$. For instance, the topic `acquiring` of the model level, linked by `Allowed association type` to `Company` and `Acquisition` becomes a relation *acquiring*, subrelation of TR, with signature $(Company, Acquisition)$.

4.1.2 Building the SG Fact Base from the Annotation Base

Let us recall that the annotation base is a TM of the instance level.

1. Each named topic t (i.e. $name(t)$ is defined) becomes an individual concept node labeled by $type(t) : name(t)$; furthermore $name(t)$ is inserted as an individual marker into I. E.g. the topic `Oracle` of the instance level, with type `Company`, becomes an individual concept node with label $Company : Oracle$.

2. Each unnamed topic t (resp. association a) becomes a generic concept node labeled by $type(t) : *$ (resp. $type(a) : *$). E.g. the association of the instance level, with type `Acquisition`, becomes a generic concept node with label $Acquisition : *$.

3. Each edge e connecting an association a and a topic t becomes a relation node labeled by $type(e)$. Its two incident edges labeled 1 and 2 are resp. linked to $f(t)$ and $f(a)$.Let us point out that it has been arbitrarily chosen to always assign $f(t)$ as the first argument of the relation, and $f(a)$ as its second argument. E.g. the edge carrying the role `acquiring` becomes a relation node with label *acquiring* linked first to the concept node labeled $Company : Oracle$ and second to the concept node labeled $Acquisition : *$.

4. Each occurrence $o = (v, opt, olt)$ in a topic t becomes a relation node labeled by *olt*. Its two incident edges labeled 1 and 2 are resp. linked to a concept node $f(t)$ and to an individual concept node labeled $opt : v$ (this individual node is created only if it does not exist yet). E.g. the ocurrence `1000000` in the topic `Oracle` with logical type `Capital` and physical type `Integer` becomes a concept node with label $Integer : 1000000$ linked to the concept node $Company : Oracle$ by a relation node *capital*.

f is a coding mapping, in the sense that it is *injective* on the set of annotations that can be defined relative to a domain ontology. This property makes it *reversible* and thus enables returning infered knowledge (e.g. an answer to a query, a constraint violation, ...) to ITM.

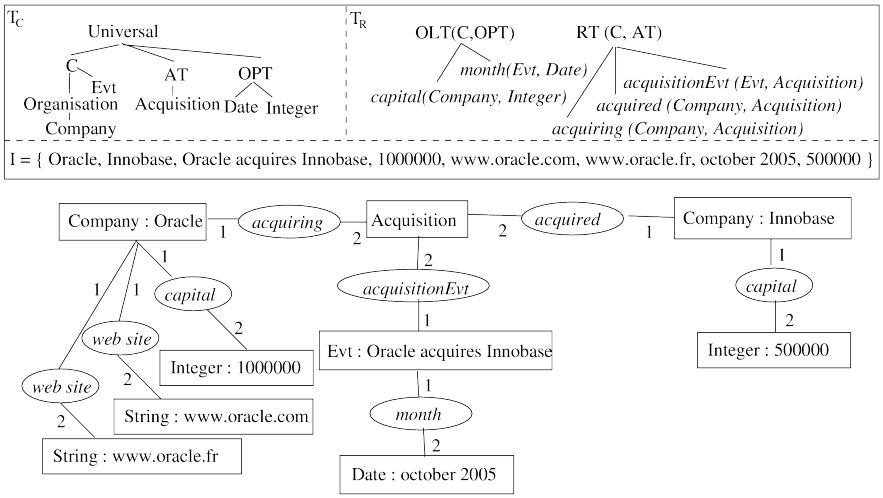


Figure 7: The mapping f applied to the knowledge represented in Fig. 3

4.1.3 Building the Constraint and Rule Sets.

The mapping f maps an ITM base into a conceptual graph base composed solely of a support and SGs. This mapping can be extended to take other constructs of the TM meta-level into account, yielding rules and negative contraints. These features are the following:

- *Maximum and minimum cardinalities* (see section 2.2). ITM minimum and maximum cardinalities are naturally translated into *SG* constraints. E.g. consider the following knowledge coded in ITM: "a topic of class `Company` should have exactly one occurrence of logical type `Capital`". It is translated into a positive constraint "every company has a capital" and a negative constraint "a company cannot have two different values for capital" (see Figure 6).

- *Hierarchical association types.* ITM allows to specify that an association type is hierarchical, i.e. transitive and antisymmetrical. Similarly to the interpretation of cardinalities, transitivity can be translated into a positive constraint and antisymmetry into a negative constraint.

Let us point out that the rules and constraints in the $\mathcal{SG}$ knowledge base do not only come from the above translations. They can also be directly added to the base to take into account knowledge that cannot be expressed with ITM, about the application domain semantics for instance. This point will be illustrated in section 5.

Let us end this section with a remark about the TM standard. As already mentioned in section 2.1, ITM does not use the whole standard. The mapping f could be extended to take other elements of the standard into account. For instance, "scopes" (as described in [BN01]) could be translated into concepts. The fact that an association is within a scope could be expressed by a relation node linking the concept nodes respectively assigned to the association and the scope nodes. To express that an occurrence is within a scope, it would be necessary to code the logical type of the occurrence as a concept rather than a relation, which would allow to link this concept to the concept node representing the scope. More generally, when one has to express properties of a TM element, it is convenient to represent this element as a concept node rather than a relation node, because as many relations as needed can be attached to it.

4.2 Reasoning Services added to ITM

What are the gains for ITM? First, f indirectly provides a formal semantics to ITM, i.e. that of conceptual graphs. Second, it provides ITM with two main features: a declarative query language and an annotation enrichment and validation service which uses inference rules to enrich the annotations and contraints to check their consistency.

4.2.1 Querying

ITM search functions can search substructures of the network restricted to one topic or one association and its neighbors. On the other hand, SGs come with a search mechanism based on projection, which is able to find substructures of *any* shape. By extending the syntax of TMs with the '?' symbol, we obtain a query language for ITM. Note that, due to the reversibility of f, every answer to the query can be translated into an answer to the original TM query problem (with the answer being the projection itself or the subgraph image of the query for this projection).

4.2.2 Enrichment and Validation Service

This service is defined to infer new knowledge from an incoming SG and check if it complies with a set of constraints. It proceeds in three steps. First, it checks the semantic integrity of the new annotation with respect to the support. If

the annotation is declared non-valid, it is rejected: an error report is returned to ITM, so that it corrects or removes the annotation. Otherwise, the system performs the second step: it adds the annotation to the fact base and applies inference rules to deduce new facts. The new deductions may lead to violate some constraints during this phase. In this case, the annotation is removed, inferences are defeased and the constraint violations are returned to ITM. In the other case, the inferred knowledge is added to the fact base and returned to ITM.

5 Experiments and Related Works

This section first presents a real-world use of previous translation in the context of media monitoring. Then it addresses related works.

5.1 PressIndex: an application to Media Monitoring

Media monitoring consists of synthesizing and classifying information from various media for a specific domain, i.e. competitive intelligence in our case, in order to ease decision making in this domain. PressIndex is a project in this area, that covers more than 9500 media sources [ACN06]. The objective is to identify and evaluate facts, rumors and official announcements in order to establish reliable information to enable an analysis on the basis of the facts of the company's competitive environment, and forecasts based on announcements and rumors concerning the evolution of this environment. This is done by automatically detecting in the base predefined patterns of rumors, announcements or facts known to be pertinent or inconsistent in order to help the monitor to establish his/her own evaluation of the relevance of this information. The main identified bottleneck is content aggregation, especially the ability to determine that the same event has appeared in different contexts (as a rumor, then as an announcement and finally as an established fact).

The kernel of the system is an ITM base driven by a semi-automatic annotation service and combined with the $\mathcal{SG}$ validation and enrichment service whose task is to validate and enrich the annotations. The annotation service represents in TM the knowledge automatically extracted from documents, and when there is a lack of information, users may complete this topic map. See Figure 8.

Let us first outline the global process. First of all, the model level of the ITM knowledge, which has been initially created by a knowledge engineer, is mapped into a support, a set of rules and a set of constraints. Note that there are several kinds of constraints and rules which are involved in the process. Some constraints and rules come from ITM (cardinalities are translated into constraints, see end of section 4.1) but most of them have been built by a knowledge engineer directly using the conceptual graph formalism because they were not expressible in ITM. These rules are defined to infer new relations between

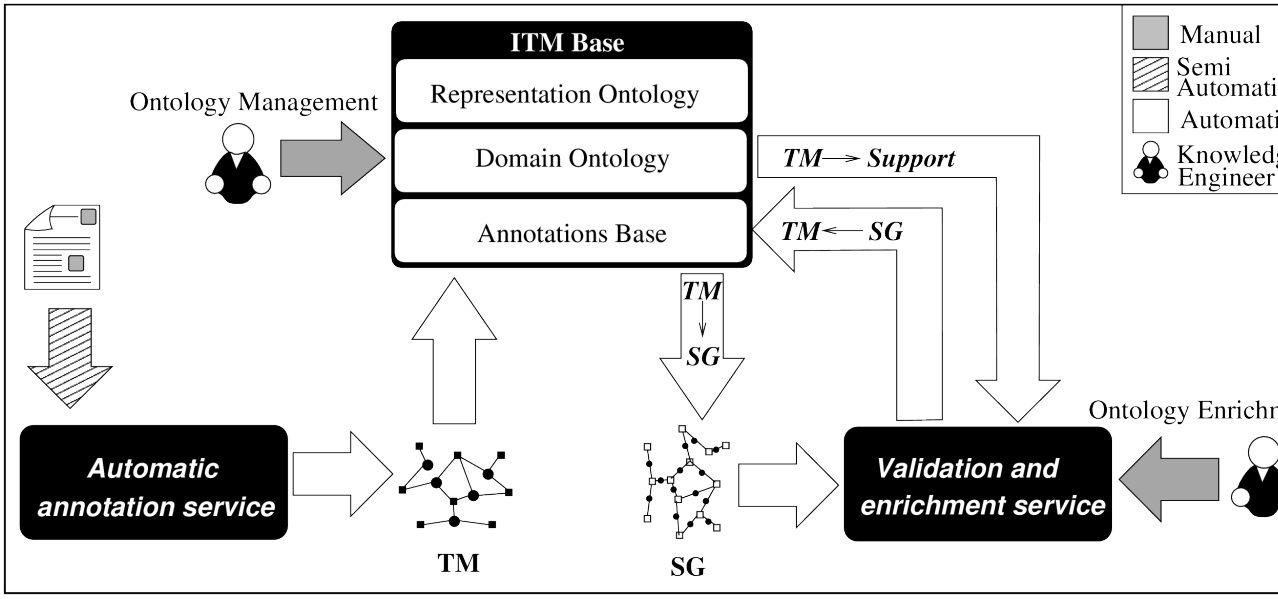


Figure 8: Architecture of the reasoning services.

facts, announcements and rumors, thus enabling the monitor to determine the pertinence of these events. These relations are:

- confirmations or invalidations of rumors from facts and announcements; for instance, the acquisition rumor of a company by an other company may be invalidated by an acquisition announcement of the latter by the former.
- confirmations or invalidations of announcements from facts;
- contradictions between rumors or between announcements.

The domain ontology has been decomposed into three parts concerning, respectively, the description of annotations, the analysis of the monitored domain, and the relevance evaluation of the relations between facts, announcements and rumors. A set of rules and a set of constraints are associated with each of these vocabularies (except for the evaluation vocabulary which only possesses rules).

The support, rules and constraints are loaded by the validation and enrichment service. When an annotation is added to the ITM base, it is transformed into a SG and sent to the $\mathcal{SG}$ service. The base managed by the $\mathcal{SG}$ service corresponds to a copy, enriched by rule applications, of a consistent part of the ITM base. Finally, if inconsistencies are found, the service returns the errors, otherwise it returns to ITM the inferred knowlege likely to interest the monitor.

A first version of the system was presented at the Semantic Web Challenge 2006 [ACN06]. The support generated from the domain ontology contained 195 concept types and 144 relations. 50 rules and 76 constraints were considered; most of them were directly added using conceptual graphs. The experiments were conducted on 12000 documents. The semi-automatic annotation service produced 1000 annotations, each of them containing between 3 and 17 nodes. At the end of the process, the base held 4000 concept nodes and the same number of relation nodes. Nine annotations were estimated to be inconsistent at incoming and 5 led to an inconsistency after inference. During the rule inference process, the 150 company acquisitions extracted from documents yielded 213 control associations and 21 rumor confirmations were detected. The graph in Figure 9 presents the processing time in the $\mathcal{SG}$ server for the i^{th} annotation. The processing time appears to be low and constant, even though the amount of knowledge increases.

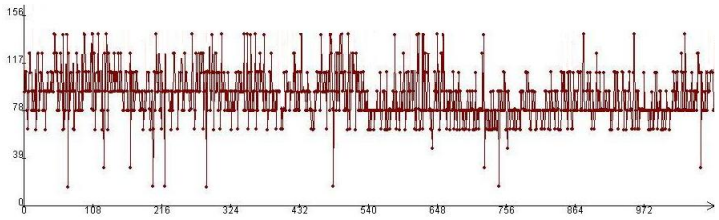


Figure 9: Processing time of the i^{th} annotation received by the $\mathcal{SG}$ service.

5.2 Related Works

To the best of our knowledge, this is the first work connecting topic maps and conceptual graphs. Works listed below are thus only partially related to ours. Their approach is related to ours either because they connect TM to a knowledge representation and reasoning formalism, or because they use a conceptual graph engine to equip another language with effective reasoning mechanisms. All these works take place in the semantic web area and the considered language is RDF.

As interoperability of knowledge-based applications is a main challenge in the semantic web, there have been several proposals for bringing TM and RDF closer. The objective is generally to make TM queriable by an RDF-aware query mechanism (based for instance on the query language SPARQL, currently a W3C candidate recommendation [W3C07]). A first approach consists in defining mappings from TM to RDF. In [Moo01] two kinds of mappings are considered. The first sort, called "modeling the model", represents in RDF the TM model (i.e. the modeling primitives) using RDF constructs. These mappings can be seen as syntax transformations. The second sort, called "mapping the model", tries to map each TM modeling primitive to one or more RDF primitives according to their "natural" semantics. The trouble with this approach is that the translation

looses information (to avoid this loss of information [Moo01] proposes to extend the TM language). In [LD01] a mapping from TM seen as graphs (as defined in [BN01]) to RDF graphs is precisely described. This mapping can be seen as belonging to the "modeling the model" approach. It has the property of being reversible. Another way of rendering TM interoperable with other languages is proposed in [Gar05a]. It is argued that syntax transformations lead to unnatural results, in the sense that the resulting RDF graph is very different from the natural RDF expression of the information represented in the TM. Rather than mapping TM to RDF, it is proposed to unify both languages into a single model. A formal model called "Q" is defined, with reversible mappings from TM and RDF to it. This model is used in [Gar05b] to provide a formal semantics to a TM query language, called "Tolog". A third way of making TM queriable consists in defining a "native" TM query language, as in [BS05]. This paper settles the formal foundations for a TM query language, in which path expression queries allow to extract information from TM. The question of how path expressions relate to formulas of description logics [BCM+03] is left open. More generally, previous works address the issue of querying TM but not the issue of reasoning with TM. Mapping TM to RDF can be seen as a way of providing TM with a semantic entailment notion imported from RDF. In the other works mentioned above, semantic entailment or logical deduction are never considered.

Finally, let us cite Corese, a tool similar to ours concerning the use of conceptual graphs [CDKFZ04]. Corese manages RDF-s knowledge and answers SPARQL queries, by translating them into conceptual graphs (mainly simple conceptual graphs and rules) which are processed by a CG reasoning engine.

6 Conclusion

We have described the main steps to provide the ITM knowledge management tool with graph-based reasoning. The mapping f from topic maps to conceptual graphs is natural and reversible, which enables, on one hand, return of results to ITM and, on the other hand, enrichment of ITM by constraints and inference rules. A first prototype that shows the feasability of this approach has been developed. Basically, it allows transformation by f of ITM knowledge (domain ontology and annotation base, as well as constraints and inference rules) and inputting the results into the CoGITaNT knowledge base, before controlling their consistency and returning the inferred knowledge to ITM. It also provides ITM with a full querying service on the knowledge base. This prototype has been tested in an application layer to implement a media monitoring system [ACN06].

The first of the next two potential improvements involves extending the query language in order to take operators on physical types (e.g. the comparison operator $<$) into account. Especially, numerical quantifications are very common in the industrial sector. The other enhancement deals with knowledge base revision. ITM allows three kinds of operations on its knowledge base: creation, deletion and modification (the latter may be seen as a combination of

deletion and creation). The $\mathcal{SG}$ service currently designed for ITM may state that a piece of knowledge created in ITM is inconsistent. An ITM user can then delete this piece of knowledge, or repair it and submit it again to the $\mathcal{SG}$ service. The problem arises when an ITM user deletes a consistent knowledge part in ITM, since the knowledge that has been possibly inferred using this part has to be removed as well. The next step will therefore involve the development of a revision $\mathcal{SG}$ service able to determine which part of the base has to be deleted.

References

[ACN06] F. Amardeilh, O. Carloni, and L. Noël. PressIndex: a Semantic Web Press Clipping Application. Semantic Web Challenge Papers, 2006.

[AMRV02] P. Auillans, P. Ossona De Mendez, P. Rosenstiehl, and B. Vatant. A formal model for topic maps. In *International Semantic Web Conference*, pages 69–83, 2002.

[BCM+03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook : Theory, Implementation and Applications*. Cambridge University Press, 2003.

[BM02] J.-F. Baget and M.-L. Mugnier. Extensions of Simple Conceptual Graphs: The Complexity of Rules and Constraints. *Journal of Artificial Intelligence Research*, 16:425–465, 2002.

[BN01] M. Biezunski and S. R. Newcomb. A processing model for xml topic maps. http://www.topicmaps.net/pmtm4.htm, 2001.

[BS05] R. A. Barta and G. Salzer. The tau model, formalizing topic maps. In *2th Asia-Pacific Conference on Conceptual Modelling*, pages 37–42, 2005.

[CDKFZ04] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the Semantic Web with Corese Search Engine. In 3^{rd} *Prestigious Applications Intelligent Systems Conference*, pages 705–709, 2004.

[CM92] M. Chein and M.-L. Mugnier. Conceptual Graphs: Fundamental Notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.

[CM04] M. Chein and M.-L. Mugnier. Types and Coreference in Simple Conceptual Graphs. In *12th International Conference on Conceptual Structures*, volume 3127 of *LNAI*, pages 303–318. Springer, 2004.

[Gar05a] L. M. Garshol. Q: A model for topic maps: Unifying rdf and topic maps. In *Extreme Markup Languages*, 2005.

[Gar05b] L. M. Garshol. tolog - a topic maps query language. In *Topic Maps Research and Applications*, pages 183–196, 2005.

[Gen97] D. Genest. Cogitant. http://cogitant.sourceforge.net, 1997.

[HP02] S. Hunting and J. Park. *XML Topic Maps: Creating and Using Topic Maps for the Web*. Addison-Wesley, 2002.

[Inf07] Stanford Medical Informatics. Protégé. http://protege.stanford.edu/, 2007.

[ISO00] ISO/IEC:13250. Topic maps. http://www.y12.doe.gov/sgml/sc34/document/0129.pdf, 2000.

[LD01] Martin S. Lacher and Stefan Decker. On the integration of topic maps and rdf data. In *Extreme Markup Languages*, 2001.

[LM06] M. Leclère and M.-L. Mugnier. Simple Conceptual Graphs with Atomic Negation and Difference. In *14th International Conference on Conceptual Structures*, pages 331–345, 2006.

[Moo01] G. Moore. Rdf and topic maps: An exercise in convergence. In *XML Europe Conference*, 2001.

[SM96] E. Salvat and M.-L. Mugnier. Sound and Complete Forward and Backward Chainings of Graph Rules. In *4th International Conference on Conceptual Structures*, volume 1115 of *LNAI*, pages 248–262. Springer, 1996.

[Sow84] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[Top01] Xml topic maps (xtm) 1.0. http://www.topicmaps.org/xtm/1.0/, 2001.

[W3C07] W3C. Sparql query language w3c candidate. http://www.w3.org/TR/rdf-sparql-query, 2007.