Formal and Graphical Annotations for Digital Objects

Nicolas Moreau, Michel Leclère, Michel Chein, Alain Gutierrez

Formal and Graphical Annotations For Digital Objects

Nicolas Moreau, Michel Leclère, Michel Chein, Alain Gutierrez
LIRMM, CNRS - Université Montpellier 2
161 rue Ada
F-34392 Montpellier cedex 5 - France
{moreau, leclere, chein, gutierrez}@lirmm.fr

ABSTRACT

This paper presents graphical tools that facilitate manual building of formal semantic annotations for digital objects. These tools are intended to be integrated in digital object (DO) management systems requiring semantic metadata (e.g. precise indexation, comment, categorization, certification). We define a multidimensional graphical model of semantic annotations that specifically allows contextualization of annotations, and we propose a methodology for building such annotations. The graph-based formalism used (derived from Conceptual Graphs), provides graphical representations that users can easily understand, and are furthermore logically founded. A graphical generic API implementing elements of the $\mathcal{SG}$ family has been developed. CoGUI consists of three specialized tools: a tool for defining a CG ontology, another for creating annotations based on a CG ontology, and finally a tool that uses the CoGITaNT platform, for querying an annotated DO base.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

Keywords

Conceptual graphs, knowledge representation, semantic annotation

1. INTRODUCTION

An enormous amount of documents are available, especially through the web, so it is now essential to have access to tools for adding semantic metadata (also called annotations) to existing documents.

To annotate a (primary) document involves associating a new (secondary) document to the initial one. This very general definition overlooks the great diversity of annotation processes. For instance:

- an annotation process can be automatic or manual or computer-aided;
- an annotation can be a natural language text or a formal expression in some formal language or a blend of both;
- annotations can be added for various reasons, e.g. comments, information, certification, indexation, etc.
- an annotation can be "objective" (e.g. author, date, etc.) or "subjective" (e.g. a point of view concerning the document).

In the present work, complex formal and graphical annotations having a logical semantics are used. Using complex conceptual structures provides precise representations of different comments about various documents. The query language is also built on the annotation language and formal semantics of the answers can be given (the answers are the formulas that entail the query). The principal drawback of this method is that the construction of rich semantical representations requires a substancial amount of human time (for constructing the conceptual vocabulary as well as the annotations themselves). Thus, our method is useful when detailed comments about documents are essential or when automatic indexation is not precise enough for the targeted goals.

An annotating tool basically has to support: reading in documents, decomposing documents into annotatable parts (DOs), creating annotations, linking them to DOs, and storing annotations (cf. for instance, [1], [17], [16], [14], [11], [20], [15]). The tools presented in this paper do not consider all the tasks an annotating tool should manage. We focus on methods and tools that enable an effective and rigorous work for a human annotator. In this paper, the term "annotator" refers to specialists of at least one aspect of the document base who want to describe a feature of this aspect. Our aim is to provide to an annotator notions (e.g. modular ontologies, patterns, prototypes), and graphical tools (implementing these notions) to facilitate building complex annotations acording to a formal model.

Our participation in several projects for the promotion and publication of a multimedia corpus led us to propose a methodology based on an annotation model. The core of such a model is a formal domain ontology, with an *annotation charter* which guides the annotator's work. One goal of

SADPI '07, May 21-22, 2007, Montpellier, France

this charter is to encourage annotators to elaborate complex annotations (*i.e.* semantic networks of concepts), whereas the traditional work of annotators is generally restricted to finding a set of keywords representing the document content. In order to reduce the time needed for constructing such a complex annotation, the annotation process is based on the identification of "conceptual patterns" the annotator has to choose, to glue these patterns together, and to modify them. These patterns are contextualized on parts of the domain ontology so that the tool can automatically restrict, at a given stage, the usable vocabulary. In this way, with a few "clicks", an annotator can construct a rich and original annotation. Another goal is to ensure that the annotations built respect a formal model. Thus, they can be searched by a logically founded search engine and, if they are built by several annotators (and/or from several points of view), they can be jointly processable.

The proposed annotation model was developed through a collaboration with multimedia document indexation experts, who tested our tools in different projects (Opales, Saphir, Logos). Most of the examples given in this paper come from the Saphir project. Saphir is a French national project headed by INA[1], MSH[2], LIRMM, LERIA[3] and NETIA[4], which aims to develop a computer-aided system for hypermedia publications described by intentional specification and rhetoric modeling. In this project, our annotation model and the associated tools include : an editor of annotation models, an annotation editor, and a query/answering tool, all built on CoGUI[5], which is a generic GUI for our graph-based models using the CoGITaNT API [9] as a reasoning server, are included in the annotating tool AudioVisualStudio developed by INA to describe a multimedia corpus.

This paper is organized as follows. Section 2 reviews the formalism we use. In section 3, we introduce the multi-dimensional annotation model and its formalization. Section 4 presents the annotation process and the way to design an annotation charter. Finally, in section 5, we explain how to make use of the annotation base for information retrieval or for query/answering.

2. GRAPH-BASED FORMALISM

The graph-based formalism used in this paper has been developed at LIRMM for 15 years [5]. The main difference relative to the initial Conceptual Graphs model of Sowa [21] is that only primitives allowing graph-based reasoning are represented. This formalism has properties which make it relevant to semantical annotation: it is a kind of semantic network allowing intuitive representation (and thus intuitive interpretation) of the represented knowledge; it allows contextualization of knowledge (*via* nested graphs) and the representation of different kinds of knowledge: annotation, patterns, rules, constraints (*cf.* $\mathcal{SG}$ family [3]); it is logically founded, allowing definition of notions of consistency (for annotations and annotation bases), and deduction from annotations; it has reasoning mechanisms based on graph theory, allowing graphic display of inconsistencies as well as deductions (and use of efficient graph algorithms); finally, it is close to the RDF language, the standard annotation language of the semantic web (*cf.* equivalence properties in [2]), and to Topic Maps, the ISO specification of document descriptions (*cf.* transformations in [4]), thus enabling the semantic interoperability of built annotations.

This section briefly reviews the components of this formalism: simple conceptual graphs (SG) built w.r.t. a vocabulary (a simple ontology), rules and constraints (*cf.* [3] for a study of the $\mathcal{SG}$ family), conjunctive types (cf. [7]), and positive nested graphs (cf. [6]) allowing to contextualize some knowledge to a specific concept.

2.1 Simple Graphs

A *simple conceptual graph* is a labelled bipartite graph. The nodes of a class (called concepts) represent entities, the nodes of the other class (called relations) represent relations between entities. E.g. the graph of Figure 1 can be seen as expressing the following knowledge: "Mary, the woman who is the subject (of the described painting), wears a red dress. She is the mother of the child Jesus".

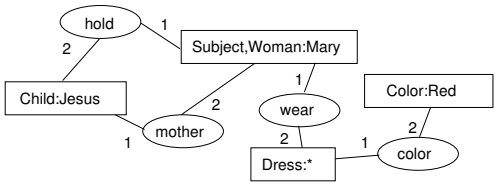

Figure 1: A simple conceptual graph.

Node labels come from a vocabulary called a *support*. A support is a structure $S = (T_C,\ T_R,\ I,\ B,\ \sigma)$, where T_C is a set of concept types and T_R is a set of relations with any arity. T_C and T_R are partially ordered sets. 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. B is a set of banned types. In this formalism, conjunctive types are permitted. However, not all conjunctions of types have a meaning. Banned types represent incompatibilities between concept types: this is a way of expressing that the conjunction of two (or more) types is *banned*. Each banned type is a conjunction of a concept type from T_C. The mapping σ assigns a signature to each relationspecifying its arity and the maximal type for each of its arguments. A support can be seen as a simple ontology, and SGs can be used to encode assertions. SGs can thus be used to annotate a document.

In an SG a concept node is labelled by a couple $t : m$ where t is a concept type (possibly conjunctive) 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 of Figure 1, the node `[Subject,Woman:Mary]` represents the well identified person *Mary*, whereas the node `[Dress:*]` represents a dress. A relation node is labelled 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

[1]Institut national de l'audiovisuel `http://www.ina.fr`.
[2]La maison des sciences de l'homme `http://semioweb.msh-paris.fr/AAR/FR/`.
[3]Informatics lab of Angers university
[4]Editor of management and multimedia content diffusion software `http://www.netia.net`.
[5]`http://www.lirmm.fr/cogui`

and relation nodes as ovals and the order on edges incidental to a *n-ary* relation node are numbered from 1 to n. An 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 labelling nodes and edges.

A graph G is *consistent* w.r.t. a support $S = (T_C, T_R, I, B, \sigma)$ if :

- the labels of the concept nodes (resp. relation nodes) belong to $(T_C \times (I \cup \{*\}))$ (resp. T_R);
- any conjunctive type of a generic concept does not contain a banned type, i.e. less than or equal to a type in B;
- for each individual marker i of G, the conjunction of all types of nodes where i is the individual marker does not contain a banned type, i.e. a type in B;
- the relation nodes satisfy their signatures defined by σ.

The fundamental notion for comparing SGs is a mapping from an 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 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 and may specialize labels. E.g. see Figure 2: there is a projection from Q to G (knowing that `hold < near`).

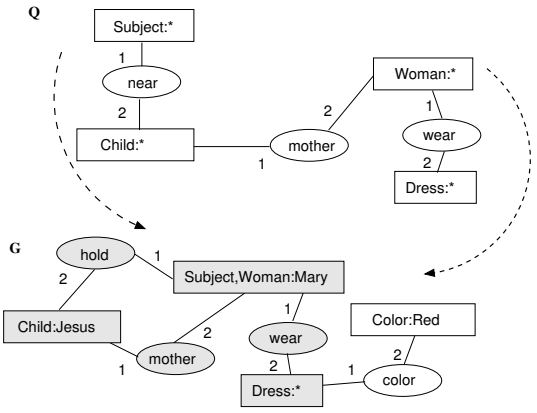

Figure 2: An example of projection from Q to G.

Conceptual graphs are provided with a first-order-logic semantic, defined by a mapping denoted by Φ. For instance, the formula assigned to the graph in Figure 1 is: $\Phi(G) = \exists x(Subject(Mary) \wedge Woman(Mary) \wedge Child(Jesus) \wedge Dress(x) \wedge Color(Red) \wedge mother(Jesus, Mary) \wedge hold(Mary, Jesus) \wedge wear(Mary, x) \wedge color(x, Red))$.

A fundamental result establishes the equivalence between projection and deduction on 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 $\Phi(S)$ ([21] for the soundness and [5] for the completeness).

Completeness is obtained up to a condition on H: H has to be in a normal form, so any individual marker appears at most once in it (i.e. a specific entity cannot be represented by two nodes). A variant of projection can be used to achieve completeness without this restriction [7].

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 an 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 (the graph Q of Figure 3: Q asks G for "*couples of persons* who have a relationship and such that one of them is the subject of the description". The sole answer is `Jesus,Mary` (knowing that `Woman < Person`, `Child < Person` and `child < relationship`). This kind of query is equivalent in expressive power to conjunctive queries in databases.

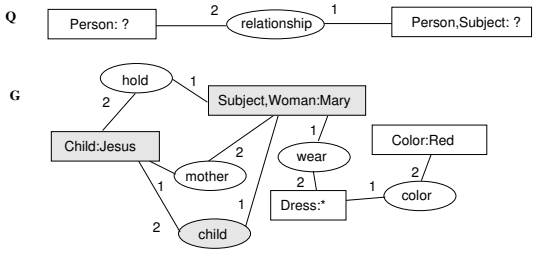

Figure 3: An example of a query.

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

2.1.1 Rules

An inference rule expresses implicit knowledge of form "if *hypothesis* then *conclusion*", where hypothesis and conclusion are both SGs. Figure 4 presents an example of a rule. Each dotted line connects a node in the hypothesis and a node in the conclusion; these nodes are called *connection* nodes. This rule expresses the reverse relationship deduced from the relation "mother": "if a woman is the mother of a person then this person is a child of this woman".

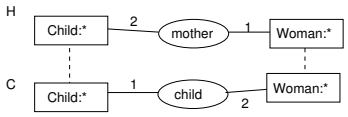

Figure 4: An example of an SG rule.

A rule R can be applied to an SG G if there is a projection from its hypothesis to G. Applying R to G according to such a projection π consists of "attaching" to G the conclusion of R by merging each connection node of the conclusion with the image by π of the corresponding connection node in the hypothesis. The graph G in Figure 3 was obtained from the graph in Figure 1 by application of the rule in Figure 4.

A logical formula can be assigned to a rule. For instance,

the formula associated with the rule in Figure 4 is: $\Phi(R) = \forall x \forall y \ (Person(x) \ \wedge Woman(y) \ \wedge \ mother(x, y) \ \rightarrow Person(x) \ \wedge \ child(y, x) \ \wedge Woman(y))$.

For a knowledge base 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 by rules into account. The knowledge base answers a query Q if an SG G can be derived from F using the rules of $\mathcal{R}$ such that Q can be projected to G. Sound and complete forward and backward chaining schemes (w.r.t. the FOL semantic) have been defined [18].

Finally, note that the query problem is not decidable for general rules. Decidability is obtained for specific rule bases, for instance rule bases that allow to "saturate" the fact base (a base is saturated if rule applications adding new information can no longer be performed); then, answering a query consists of projecting it to the saturated base.

2.1.2 Constraints

Constraints can be either *positive*, expressing knowledge such as: "if *condition* is found, so must *obligation*", or *negative*, expressing knowledge such as: "if *condition* is found, *interdiction* must not". They have the same syntactical form as rules. For negative constraints, one can restrict their syntactic form to an SG – which represents a forbidden graph – by pairwise merging corresponding connection nodes. Figure 5 shows two constraints: the positive constraint C^+ says that "a contemporary painting must be associated with a technic" ; the negative constraint C^- says that "a sketch is never executed with painting oil".

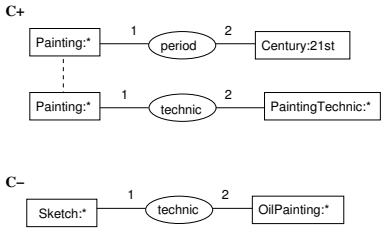

Figure 5: Two SG constraints.

An 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 there is *no* projection from C^- into G.

An 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 only defined 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 SGs derivable from F by $\mathcal{R}$ are consistent with respect to $\mathcal{C}$. With positive constraints, consistency is a more complex notion that we do not present here (see [3]).

2.1.3 Constraints and Difference

Note that constraints enable specification of cardinality constraints. E.g. Figure 6 specifies the functional constraint: "a person has only one mother". Nevertheless two different concept nodes in an 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 an SG. This relation, denoted dif, is antireflexive and symmetrical and its logical translation is $\neq$. dif is pictured by difference links (see for instance the graph in Figure 6: there is a difference link between the nodes with label `Woman`).

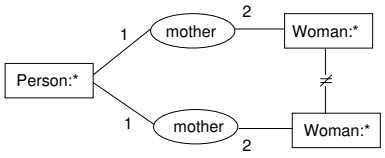

Figure 6: A negative SG constraint for the functionality of the `mother` relation.

Projection (say π from G to H) has to take the dif relations into account: if $\{c_1, c_2\} \in dif_G$ then $\{\pi(c_1), \pi(c_2)\} \in dif_H$. However, even extended in this way, projection is no longer complete with respect to deduction in classical logic. Indeed, as very briefly explained, in classical logic the law of the excluded middle holds true. Applied to equality and difference, this law says that "given two entities, either they are equal or they are different". This leads to a case reasoning that cannot be computed by a simple projection, but rather by an exponential number of projections (in the size of H). Note however, that if we consider a logic in which the law of the excluded middle is not accepted (e.g. intuitionist logic), projection is complete [13]. A third way of dealing with difference is to make the closed-world assumption. In this case, all distinct concept nodes are assumed to represent different entities, thus the law of the excluded middle is not applicable and projection is complete.

2.2 Typed Nested Graphs

Nested graphs [8], equipped with a FOL semantic generalizing the FOL semantic for SGs, have been introduced to specify different levels of description (they are close to Networked RDF Graphs [19] but different since variables can be shared by different graphs). For example, knowledge about a desk and knowledge about what is represented on a photo which is on this desk are not at the same level. Typed nested graphs (TNGs) [6] are nested graphs with additional information about the type of description at each description level.

TNGs are defined recursively from SGs as follows:

1. A typed nested graph is obtained from a simple conceptual graph by adding, to the label of each concept node c, a third field, denoted $Desc(c)$, equal to $**$ ($**$ can be considered as the empty description).

2. Let G be a typed nested graph, let $c_1, c_2, \ldots, c_k$ be concept nodes of G, $n_1, n_2, \ldots, n_k$ be nesting types, and $G_1, G_2, \ldots, G_k$ be typed nested graphs. The graph obtained by substituting (n_i, G_i) in the description $**$ of c_i for $i = 1, 2, \ldots, k$ is a typed nested graph.

In drawings, pairs (n_i, G_i) associated with a concept c are represented by boxes in this concept, labelled by the type n_i and containing the graph G_i. Figure 7 shows a typed nested graph.

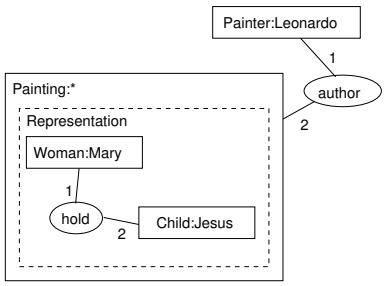

Figure 7: A typed nested graph.

As soon as we consider TNGs, the support has to be extended with T_N, a poset of nesting types: it is a new hierarchy in the vocabulary similar to the hierarchies of concept types or relations.

3. ANNOTATION FORMALISM

The first experiments with annotators highlight the fact that an annotation task is not performed in a single pass over the input document. An annotation is instead performed in a top-down way, which follows a specific objective called an *annotation dimension*. One roughly identifies parts of the input document that are related with the current objective; these parts can be associated in a general description. Then each part is studied with a thinner grain, it is divided into subparts which are associated with more specific descriptions, and so on.

In the Saphir project, two corpus are considered. One corpus concerns TV news annotated along the following dimensions: the sound track, the video track, the text and image inlays, etc. The other corpus concerns researcher interviews. Such an interview is annotated by first considering the theme of the interview, then the rhetoric, which is decomposed into discourse acts, themselves described by using a finer description of the theme and so on.

In order to contextualize annotation, the nested graphs can be combined with modular ontologies. A *modular ontology* is an ontology composed of *modules*. A module is a sub-vocabulary of the global vocabulary which is associated with a nesting type. Modules and nesting types are used to represent *dimensions* of an annotation. The new conceptual graph model called *Modular Nested Graph* (MNG) is defined in the three following subsections.

3.1 Modules

A module is defined as a consistent sub-vocabulary of the general vocabulary. In the sequel, we consider supports without individual markers. Individual markers are only introduced in the so-called graphs of individuals (*cf.* section 4.1.3).

Definition 1. A *module* of a support $S = (T_C, T_R, T_N, B, \sigma)$ is a triple $m = (T'_C, T'_R, T'_N)$, where:

- $T'_C \subseteq T_C$;
- $T'_R \subseteq T_R$;
- $T'_N \subseteq T_N$;
- $\sigma[T'_R] \subseteq T'_C$: each concept type t which belongs to the signature of a relation r in m ($r \in T'_R$) must also belong to m ($t \in T'_C$).

For any module $m = (T'_C, T'_R, T'_N)$ of a support $S = (T_C, T_R, T_E, B, \sigma)$, its underlying support $S_m = (T'_C, T'_R, T'_E, B', \sigma')$ is defined as follows:

- T'_C (resp. T'_R and T'_N) is ordered by restricting the order relation of T_C (resp. T_R and T_N) to elements in T'_C (resp. T'_R and T'_N);
- B' is the restriction of B to the elements in T'_C;
- σ' is the restriction of σ to T'_R.

3.2 Dimension

The module notion is used to define modular ontologies and annotation dimensions.

Definition 2. A *modular ontology* is a triple $O = (S, M, D)$, where $S = (T_C, T_R, T_N, B, \sigma)$ is a support, M is a set of modules defined w.r.t. S, and D is a mapping from T_N to M. A *dimension* is a pair $d = (n, m)$, where $n \in T_N$ is a nested type, and $m \in M$ is the module $D(n)$ representing the point of view that n designates.

The partial order on nesting types induces a specialization relation between dimensions. A dimension $d' = (n', m')$ is said to be more specific than a dimension $d = (n, m)$ if $n' \leq n$. The hierarchization of dimensions leads to constraints on the associated modules: a type used in a specific dimension must be interpretable in a more general dimension. Let us suppose that ($m = (T_C, T_R, T_N)$ and $m' = (T'_C, T'_R, T'_N)$. Then, two kinds of constraints can be considered:

- inclusion constraints: $T'_C \subseteq T_C$, $T'_R \subseteq T_R$, and $T'_N \subseteq T_N$;
- specialization constraints: for every $t' \in T'_C$, there is $t \in T_C$ such that $t' \leq t$, and the same conditions hold for the relationtypes and the nesting types. The specialization constraint is more flexible than the inclusion constraint because it allows the presence, in a specific module, of a concept type which does not appear in the more general module.

3.3 Modular Nested Graph

An MNG G is a TNG defined with respect to a modular ontology $O = (S, M, D)$ in the following way:

1. the root graph G_r composed of all the nodes not contained in a nesting box (nodes of level 0) is a graph which is consistent w.r.t. S;
2. for each typed nested graph (n_i, G_i), graph G_i is consistent w.r.t. S_{m_i} the support underlying the module associated by D to the nested type n_i.

4. ANNOTATION METHODOLOGY

The annotation methodology we propose is based on the MNG formalism. One proposes to define a charter for annotators in an annotation model. This charter is then implemented in our annotation tool to guide annotators. Finally, generic mecanisms of CoGITaNT allow querying of the annotation base.

4.1 Annotation Model

An annotation model first has to define the dimensions one wants to use relative to a given document corpus. Such a model is thus based on a modular vocabulary. The designors of these models can attach, at each module of this ontology, some ontological graphs which constrain and guide users in their annotation. In the annotation tool, these graphs act like an annotation *charter*. This charter will be exploited by users, helping them to not always describe universal knowledge (rules), constraining them in excluding several forms of annotations (constraints), guiding them in the annotation process (patterns), or facilitating repetitive operations (prototypes). We have developed a software tool dedicated to the construction of modular ontology and ontological graphs. This tool is a specific instance of CoGUI (cf. Figure 8).

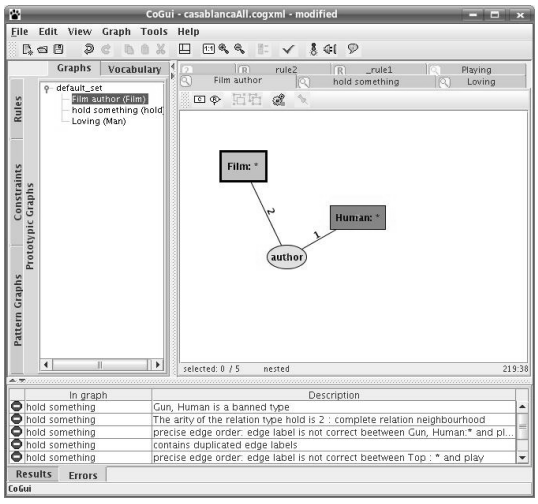

Figure 8: Construction of the model (example of a prototypical graph).

4.1.1 Modular Ontology Construction

The first step in the definition of an annotation model is the creation of a modular ontology. In this paper, we did not develop the methodology to elaborate an ontology (see [10] for a methodology adapted to CGs), but our tool is designed for graphical editing of the global support. To design the modules, the user chooses a selection of concept types, relation types and nesting types in the global vocabulary which can be constructed before this selection or at the same time. Selection of relation types can create some problems related to signature constraints. The user has three choices in such a situation:

- he/she removes the relation from the module;
- he/she adds all concept type parameters of the relation signature in the module;
- he/she adds (in the module and in the general vocabulary) a new relation defined as a specialization of the first and chooses, for these parameters, more specialized type concepts in such a way that they belong to the module.

Dimensions offer a first level of annotation charter in restricting the usable vocabulary in each dimension. The next step is to specify ontological graphs which can be associated with all or only some specific modules.

Rules allow to describe knowledge which is applied to all documents in the base. Rules are useful to simplify work of annotators in defining in intension some types of knowledge. Using these rules, the system can enrich annotations (e.g. a given relation is symmetrical or transitive). Since an annotator knows the rules, he/she can reduce the entered annotation to a sufficient part; the application of rules guarantees the completion.

Constraints are used to define absurd or forbidden knowledge, and can also be used to restrict the signature of a relation at a precise generality level in a given dimension. Constraints allow to define some mandatory parts in annotations and limit edition errors.

4.1.2 Conceptual Patterns

The second annotation charter level is composed of "conceptual patterns" that annotators can select, glue and modify. The identification and use of such a pattern has several advantages:

- to help annotators to develop a richer description than a simple list of concepts. An annotator starts this work from a generic graph selected with respect to the annotation objective, and just has to complete (specialize or modify) it;
- to avoid the repetitive work of reconstruction of similar graphs from one annotation to another. This reduces the time required to annotate and annotation tasks by accelerating the annotation process and avoiding a certain weariness of annotators;
- to achieve relative unity of the annotation base by guiding annotators toward a common "annotation genus";
- to facilitate the training of annnotation work by a novel annotator through patterns which permit him/her to appropriate the pre-defined annotation genus.

In using these patterns, the designer of the annotation model can formalize an annotation scenario: in each dimension, he/she identifies one or several initial annotation patterns and some extension points from which annotators can extend the document description. Two kinds of patterns have been defined: "pattern graphs" associated with the nesting type and the prototypic graphs assoicated with concept types or relations.

A *pattern graph* is an MNG associated with a dimension d which represents a starting point when describing a document with respect to dimension d. It pools general and frequent notions that arise when describing a document with respect to d. A pattern graph has to respect the vocabulary of the module the dimension is associated with. This constraint ensures that the use of a pattern does not upset the annotation consistency. One can associate several pattern graphs to one dimension d; e.g. when several annotation objectives fit the same dimension. Figure 9 is an example of a pattern graph, used in the Saphir project, associated with the *thematic* dimension. It is used to describe the use of a language.

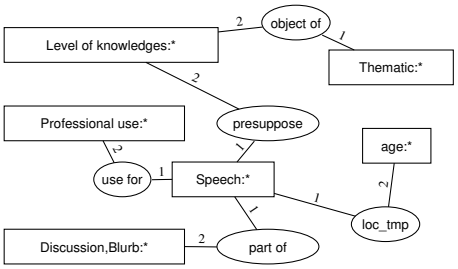

Figure 9: A pattern graph associated with the *Rhetoric* dimension.

A *prototypical graph* is an SG which can be associated with a concept type or with a relation. A prototypical graph defines usual contexts of a type. A prototypical graph has a formal parameter, a special node that we call the *head*, with the same type (or relation) as the type from which it is the prototype. Annotation graph nodes having a type from which a prototypic graph is defined are potential extension points that can be merged with the prototype head. A concept type or relation can have several prototypes which represent different ways to describe an entity or relation. As prototypical graphs are not related to dimensions, the annotator will only have access to prototypes whose vertices are all in the vocabulary of the current dimension during the annotation process. Figure 10 shows an example of a prototypical graph for the concept type *linguistic sign system*.

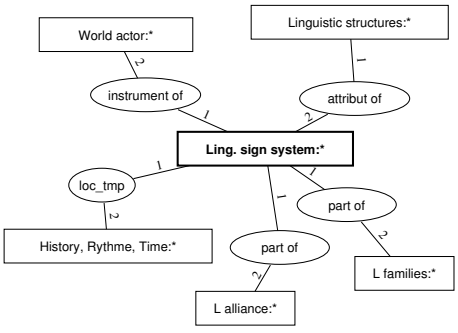

Figure 10: Prototypical graph for "linguistic sign system".

4.1.3 Graphs of individuals

Finally, at the third annotation charter level, one introduces some encyclopedic facts. The role of these facts is to ensure basic knowledge about the domain (or one dimension); it is useless to repeat this in each annotation. Especially, it allows the user to introduce some ontological individuals and to define the knowledge about them.

A graph of individuals is an SG which owns at least one individual marker. The set of ontological individuals is computed from all the individual graphs. Any occurrence of these individuals must have a type that is coherent with the type defined in the graphs of individuals. During the annotation process, if an ontological individual is used, then the graph of individuals which has introduced it can be merged to the annotation graph where it appears. Figure 11 presents an example of a graph of individuals which defines the individual *Mary*. This individual marker will have the type *Woman* in the vocabulary.

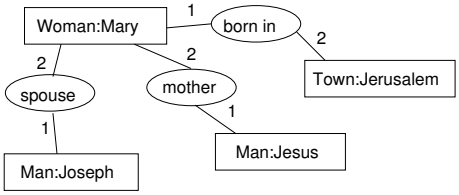

Figure 11: Graph defining the individual *Mary*.

4.2 Structuring model

In order to annotate a DO corpus, we propose an universal structuring model based on the decomposition of resources in sub-resources. This model must allow the user to focus on significant parts of a resource, in order to precisely describe a part of it or to retrieve relevant information when querying the annotation base.

Any resource of the corpus can be decomposed into sub-resources, which in turn are decomposable. For example, a video may be structured in *segments*, a picture in *details*, and a text in *paragraphs*. An hypermedia, like a web page, can be structured by elements it contains (text, video, music, etc.), which in turn are structured as described above (see Figure 12).

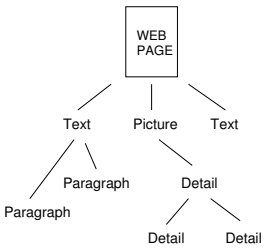

Figure 12: Example of web page structuring.

Such a structure is a tree in which each node is a resource that can be annotated. An *annotation descriptor* is composed of: a *reference* to the resource, *meta-data* (*e.g.* author, date of creation, user access permissions, keywords to classify the resource, etc.), and a *semantic description* of the resource: a *semantic annotation*. A similar model is presented in [12].

In the context of Saphir, the corpus of resources is essentially composed of audiovisual documents from INA's archives,

and interviews from MSH. The structuring process of DO consists of a video segmentation. This is made by the Saphir Visual Studio software developed by INA for the project (*cf.* Figure 13).

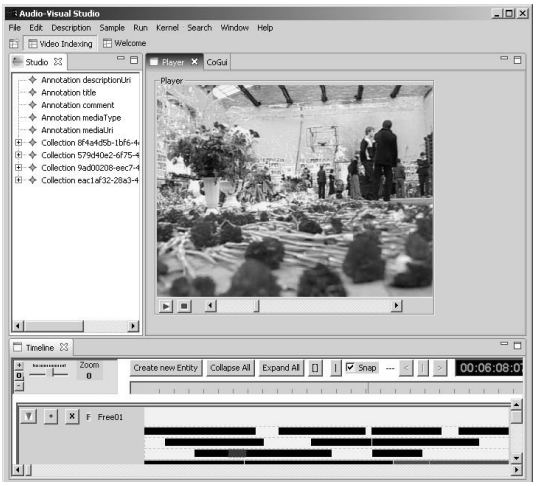

Figure 13: Software of DO structuring Saphir Visual Studio.

4.3 Design of annotations

To obtain a generic and reusable semantic annotation tool, one has to develop it independency from resource annotation descriptors. We make the assumption that any structuring tool uses identifiers to manage its annotation descriptors and reuse these identifiers to refer to resources (via their annotation descriptor) in our semantic annotations. The semantic annotation tool we develop thus has to be integrated in any structuring workbench of corpus of documents.

A semantic annotation is thus a MNG defined on a modular ontology which represents the different annotation dimensions (e.g. in Saphir: thematic, rethoric, pragmatic, etc.). In the Saphir project, the rooted graph of MNGs is only composed of an isolated individual concept node with the identifier of the corresponding descriptor as individual marker (and possibly the kind of resource (e.g. segment, text, image...) as type). The semantic description itself is thus given by way of graphs nested in this node. Figure 14 presents an example of such a Saphir semantic annotation (`#id` identifies a specific descriptor (and thus a resource) in the structuring system of the corpus).

4.3.1 Scenario of annotation

Once the model is defined, the user can create annotations. In the Saphir project, our tool is bundled in the Saphir Visual Studio segmentation tool , developed by INA (figure 15)

An annotation graph is built after several steps which usually follow this process:

1. The user selects an annotation descriptor. A conceptual graph composed of an unique concept, named the head of the annotation, is generated. This concept is typed with *Annotation* and its individual marker is the id of the descriptor. In the Saphir project, descriptors are typed and this type is used as the type of the head concept (e.g. Segment, Picture, Detail, Text, etc.). Thus, the user can build his/her annotation by inserting several nested graphs in this concept. The graph of figure 14 is an annotation example.

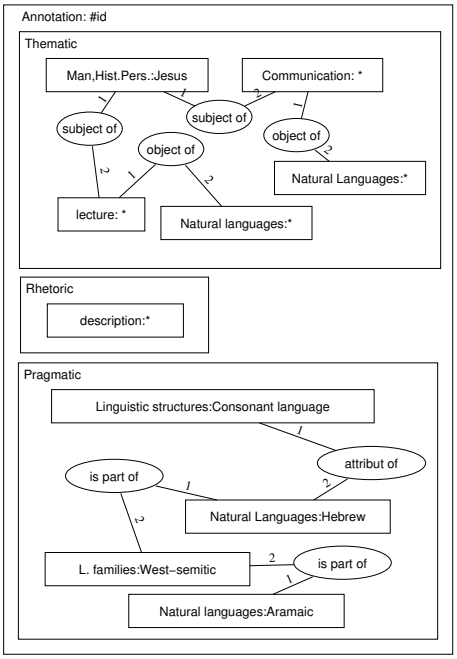

Figure 14: A semantic annotation with three dimensions.

2. The user chooses an annotation dimension. The tool automatically creates the corresponding nesting box and "zooms" within the dimension, hiding the whole annotation to let the user focus on the dimension. The tool also computes the vocabulary associated with the dimension to show the user only relevant information. The user can view the whole annotation at any time if necessary.

3. The user can select a pattern graph to start an annotation process. At annotation creation, or at a dimension activation, the tool shows a list of patterns corresponding to the dimension. The annotation process is not constraining, it is just a guide. The user can specify, generalize or even delete vertices of the inserted pattern.

4. The user can select prototypic graphs. During annotation, the user can request to complete the description of a vertex with one of his/her utilisation contexts. The tool automatically adds information on the prototypic graph in the annotation, assimilating the current vertex with the head of the prototype. Like pattern graphs, applied prototypic graphs can be modified by the user.

4.3.2 Verification and validation

Once the user has built an annotation, he/her has to validate it. A test of consistency of the annotation wrt the modular

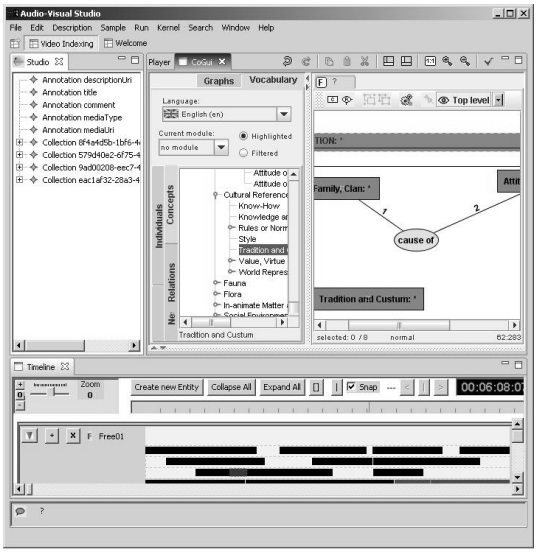

Figure 15: An annotation made with our tool embedded in Saphir Visual Studio.

ontology is made, and if it induces errors, a list of error messages is shown. A graphic error visualisation mechanism allows the user to modify the annotation. Then the tool verifies the respect of constraints using CoGITaNT.

When several users work on the same base, two classical problems concerning the individual markers are arise. There is a problem of *redundancy* when different individuals describe the same entity. This can lead to *silence* when querying the base. Another issue is the *incoherency* problem that occurs when the same individual is used to describe two different entities. This can lead to *noise* when querying the base.

Our tool verifies that the set of types that caracterize an individual marker does not produce a banned type.

5. EXPLOITATION OF THE ANNOTATION BASE

The tool can be used to retrieve information from a base of annotations constructed with our model. Requests can focus on the knowledge represented within the annotation, e.g. one wants to know the names of the actors who play heroes in the DO corpus. Requests can also focus on the resources themselves, e.g. one wants to know all the videos concerning a given theme.

Requests can retrieve documents whose annotation contains specific knowledge (more precisely, are specializations of the query), or can extract particular knowledge from an annotation.

5.1 Form of requests

An elementary request (G, B) is composed of a base of annotations B and a request graph G. Answers to a request are computed using the previously defined projection operation. More precisely, the answers to a request are computed from the set of annotations (in the base) which are specializations of the request.

A user may want two types of information:

The knowledge retrieved by the request. In this case, he/she is interested in the projection image. This information is relevant because data returned by the request may be more specialized than in the request. Particulary, the request can return individual markers corresponding to generic markers in the request, and these individual markers can be the information sought by the user. For example, the user may want "all heroes who hold a revolver".

A user can have a different goal, he/she may want the list of resources that contain a given information. For example, in the Saphir Project, one of the goals is to make a publication by intention of an hypermedia, so the user needs a list of video candidates which match the users' needs to build the final document. In this case, the system returns the individual markers, i.e. the identifiers, of the head nodes of the annotations which are specializations of the request.

5.2 Boolean requests

Elementary requests can be combined by boolean operators. The conjunction, disjunction, and negation can be used.

Let us denote by $ans(G, B)$ the results of the request graph G on the base B.

Conjunction, disjunction and negation of requests are defined as:

- Disjunction: $ans(G_1 \vee G_2, B) = ans(G_1, B) \cup ans(G_2, B)$
- Conjunction: $ans(G_1 \wedge G_2, B) = ans(G_1, B) \cap ans(G_2, B)$
- Negation: $ans(\neg G, B) = B \backslash ans(G, B)$

If a request graph is restricted to a set of concepts, then boolean requests can be assimilated to a boolean keyword information search.

6. CONCLUSION

This work introduces basis of a methodology for formal annotation based on the definition of a multi-dimensional annotation model providing guidance, assistance and validation of a manual annotation work. This methodology is used on CoGUI, our graphical interface for edition, validation and exploitation of knowledge graphs.

This proposition now has to be validated through comprehensive experimentation involving real exploitation of built annotations. The Saphir Project, which aims to produce multimedia publications based on annotations of a corpus, is a first case of experimentation.

We are currently working on the individual markers management problem which arises when annotations are made by several annotators. Experimentations show that annotators often use individual markers. Thus, it is important to ensure that the same individual entity is well identified by a single individual marker and, conversely, that an individual

marker is not used to designate two different entities. We propose to base the validation of an annotation on a *confidence* notion which expresses the level of knowledge of a user on knowledge associated with an individual marker.

Otherwise, we are working on a mecanism of query/answering allowing the *fusion of dimensions* in a more general dimension to compute approximate answers.

7. REFERENCES

[1] O. Aubert, P.-A. Champin, and Y. Prié. First international workshop on semantic web annotations for multimedia,15th world wide web conference. In *Proc. of SWAMM 2006*, page 12.

[2] J.-F. Baget. Simple Conceptual Graphs Revisited: Hypergraphs and Conjunctive Types for Efficient Projection Algorithms. In *Proc. of ICCS'03*, volume 2746 of *LNAI*. Springer, 2003.

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

[4] O. Carloni, M. Leclère, and M.-L. Mugnier. Introducing graph-based reasoning into a knowledge management tool: An industrial case study. In *IEA/AIE*, pages 590–599, 2006.

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

[6] M. Chein and M.-L. Mugnier. Positive nested conceptual graphs. In *International Conference on Conceptual Structures*, pages 95–109, 1997.

[7] M. Chein and M.-L. Mugnier. Types and Coreference in Simple Conceptual Graphs. In *Proc. ICCS'04*, LNAI. Springer, 2004.

[8] M. Chein, M.-L. Mugnier, and G. Simonet. Nested graphs: A graph-based knowledge representation model with FOL semantics. In *Principles of Knowledge Representation and Reasoning*, pages 524–535, 1998.

[9] CoGITaNT. Conceptual graphs integrated tools allowing nested typed graphs. `http://cogitant.sourceforge.net`, 1997.

[10] F. Furst, M. Leclère, and F. Trichet. Operationalizing domain ontologies: a method and a tool. In *16th Europeean Conference on Artificial Intelligence (ECAI)*, pages 318–322, 2004.

[11] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics*, 2(1):227–250, 2005.

[12] M. Koivunen and R. Swick. Collaboration through annotations in the semantic web. `http://www.w3.org/2001/Annotea/Papers/SAbook/annotea.html`, 2001.

[13] M. Leclère and M.-L. Mugnier. Simple Conceptual Graphs with Atomic Negation and Difference. In *Proceedings of ICCS'06*, pages 331–345, 2006.

[14] L.Hollink, A.Th.Schreiber, B. Wielemaker, and B.Wielinga. Semantic annotation of image collections. In *In Proceedings of the KCAP'03 Workshop on Knowledge Markup and Semantic Annotation*, Florida, USA, October 2003.

[15] P. Martin and P. Eklund. Embedding knowledge in web documents. In *Proceedings of the 8th Int. World Wide Web Conference*, pages 1403–1419.

[16] K. Petridis, S. Bloehdorn, C. Saathoff, N. Simou, V. Tzouvaras, S. Handschuh, Y. Avrithis, Y. Kompatsiaris, S. Staab, and M. G. Strintzis. Knowledge representation and semantic annotation for multimedia analysis. *IEE Proceedings on Vision, Image and Signal Processing, Special issue on the Integration of Knowledge, Semantics and Digital Media Technology*, 153(3):253–394, 2006.

[17] L. Reeve and H. Han. Survey of semantic annotation platforms. In *Proceedings of the 20th Annual ACM Symposium on Applied Computing, Web Technologies and Applications track*, Florida, USA, October 2005.

[18] E. Salvat and M.-L. Mugnier. Sound and Complete Forward and Backward Chainings of Graph Rules. In *Proc. of ICCS'96*, volume 1115 of *LNAI*, pages 248–262. Springer, 1996.

[19] S. Schenk and S. Staab. Networked rdf graphs. Research Report RR3/2007, Universitat Koblenz-Landau, 2007.

[20] A. T. Schreiber, B. Dubbeldam, J. Wielemaker, and B. Wielinga. Ontology-based Photo Annotation. *IEEE Intelligent Systems*, May/June(1-2):227–250, 2001.

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