Positive Nested Conceptual Graphs

Michel Chein &[1] Marie-Laure Mugnier
LIRMM (CNRS & University of Montpellier)
email: chein@lirmm.fr, mugnier@lirmm.fr

Abstract. This paper deals with positive (i.e. without negation) nested conceptual graphs (NCGs). We first give a general framework - graphs of graphs provided with morphism - for defining classes of NCGs. Then we define a new class of NCGs - typed NCGs - and we show that known kinds of NCGs can be described very simply as classes of the general framework. All NCG models considered generalize the simple CG model in the sense that they involve objects which are generalizations of simple CGs and reasonings on these objects are based on a graph operation (projection) which is a generalization of that used for simple CGs. Furthermore, the general framework introduced allows one to consider all these models as slight variations of a unique notion. This study has been initiated by applications we are currently involved in.

1. Introduction

The work reported in this paper concerns conceptual graphs (CGs) for representing and reasoning on hierarchically structured knowledge. In these CGs, concept nodes may express complex information represented by CGs. We may call them *positive* nested conceptual graphs, in order to differentiate them from CGs with nested negative contexts. However, for the sake of brevity, we simply call them *nested CGs* (NCGs).

Let us first recall that we develop the CG model as a *graphical* knowledge representation model, where "graphical" is used in the sense of [1], that is a model that "uses graph-theoretic notions in an essential and nontrivial way". Basic objects are *simple CGs* (SCGs), which are connected labelled bipartite graphs (they correspond to the CGs introduced in [2], chap. 3). The fundamental operation for reasonings is *projection*, which is a morphism between these objects. Since it is sound [2] and complete (when graphs have a normal form, [3]) with respect to deduction in first order logic, reasonings on simple CGs can be based on projection computing. Furthermore, projection induces a structural *subsumption* relation on simple CGs. All NCG models considered in this paper generalize this simple CG model, while preserving its properties.

The simple CG model has been implemented on CoGITo [4] [5], a software platform for developing knowledge-based applications for which every piece of knowledge is represented by CGs. But in several applications this model proved to be inconvenient for expressing the knowledge involved. Let us give two examples of applications we are currently involved in. First, a simulation application of the required behavior of human agents in case of a catastrophe in a factory [6] [7]. This behavior is described by CGs which represent situations, pre-conditions and post-conditions of actions to be performed, each of these elements being themselves described by CGs. Second, a study of document retrieval for university libraries [8].

---

[1] & is commutative

The content of a document is partially represented by a CG. At first level, the title, the author and the subject are encoded. At second and deeper levels, the subject is described in further detail. In both cases, knowledge can be naturally structured by hierarchical levels and reasoning must respect these levels. A convenient way of encoding these levels is to put simple CGs within simple CGs. Descending from level to level can be seen as a "zooming" operation [9][10].

This led us to a first NCG model. In these NCGs, concept nodes are labelled with three fields: to the type and referent is added a *description*, which may be either generic (i.e. empty) or a set of NCGs. Projection is recursively defined according to the recursive structure of NCGs. Let us specify that in [11][12] a model close to NCGs is defined (but projection in this latter model is not recursive).

It appeared in applications that a mean of specifying the semantics of nestings was desirable. This led us to the *typed* NCG model. Briefly, besides the posets of concept and relation types we now have a poset of *nesting types*. To each graph of a concept description is associated a nesting type, which specifies the link between this graph and the including concept node. The typed NCG model is presented in this paper.

Several kinds of NCGs have been used by other researchers. Most of them follow more or less Sowa's formalism in [9][10]. Although Sowa does not consider graph operations for reasoning, a projection associated to these graphs can be defined. One obtains a new NCG model. Then a natural question arises: how can we compare these models with respect to their expressive power? More specifically, which model should we choose for extending the CoGiTo platform? In order to answer these questions, we propose a general framework in which various NCG models can be defined and compared. Fundamentally, we deal with objects (categories in the mathematical sense) which are graphs within graphs provided with morphism. We give some examples of known kinds of NCGs and prove that they are specific cases of such objects. A generalization relation over classes can be defined. Briefly said, a class C is more general than a class C' if there is a transformation from C' to C which preserves morphisms (similarly to the notion of problem reduction in complexity theory). This allows us to show that classes of NCGs considered in this paper are equivalent (each one is more general than the others). From a practical viewpoint, this means that any of these models may be chosen for implementation purposes. Several models may be used at an interface level, the choice depending on the application or the taste of the user. Solving a problem involving projection computing can be done by translating this problem into the implemented model, via one of the given transformations, and translating back the answer into the interface model.

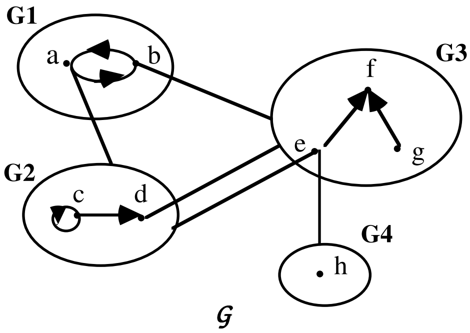
The paper is organized as follows. In next part we define the general framework for defining and comparing models of graphs within graphs, with respect to morphism (or subsumption) computing. Part 3 is devoted to several NCG models. Their equivalence is proven in 3.4. For clarity reasons, these models are first presented without coreference links. Then part 3.5 shows how coreference links can be introduced. In conclusion, we discuss logical semantics of previous NCG models.

2. Graph of Graphs Framework

2.1. Graph of Graphs

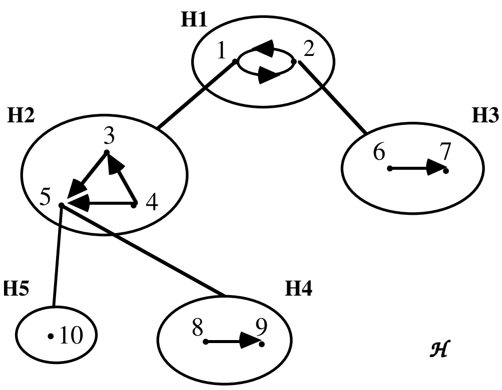
In this part a combinatorial structure, provided with a morphism notion, is introduced. This structure, called a *graph of C-graphs*, is defined from any class *C* of graphs for which a morphism is defined (directed or undirected graphs, multigraphs, labelled graphs, SCGs,...). Let us recall that a *morphism* from a graph *G* to a graph *H* is a mapping, say *f*, from the node set of *G* to the node set of *H*, which preserves adjacency, i.e. for each edge *xy* of *G*, *f(x)f(y)* is an edge of *H*. When *G* and *H* are labelled graphs, there may be additional constraints on labels. For instance, projection on simple CGs is a morphism, which keeps the ordering of edges incident on relation nodes and may decrease node labels (i.e. specialize types and, for concepts, replace the generic marker with an individual one).

Definition 1. *Let C be a class of graphs. A* ***graph of C-graphs*** *(Figure* 1*) is a couple* $G = (\{G_1, ..., G_k\}, W)$ *where :*
(1) *any* G_i *belongs to C, and for all* $i \neq j$, G_i *and* G_j *are disjoint*
(2) *W is a set of triples* (x, G_i, G_j)*, where x is a node of* G_i *and* $i \neq j$.



$\mathcal{G}$

$W=\{(a, G_1, G_2), (b, G_1, G_3), (d, G_2, G_3), (e, G_3, G_2), (e, G_3, G_4)\}$



$\mathcal{H}$

$W=\{(1, H_1, H_2), (2, H_1, H_3), (5, H_2, H_5), (5, H_2, H_4)\}$

Fig. 1. Graphs of graphs, *G* and *H*.

When the class C is implicit we use the term *graph of graphs*. A triple (x, G_i, G_j) can be interpreted in the following way: x is a *complex node* of G_i, which is described by G_j. The *skeleton* of a graph of graph $\mathcal{G}$ is a graph which explicits the underlying structure of $\mathcal{G}$, i.e. relationships among the G_i, and forgets the content of each G_i (Figure 2).

Definition 2. *The* ***skeleton****, $S(\mathcal{G})$, of $\mathcal{G} = (\{G_1, ..., G_k\}, W)$, is a directed labelled graph, whose node set is in bijection with $\{G_1, ..., G_k\}$ and edge set is in bijection with W. There is a directed edge from G_i to G_j labelled by x for each triple (x, G_i, G_j) of W.*

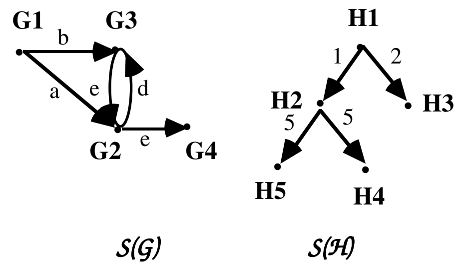
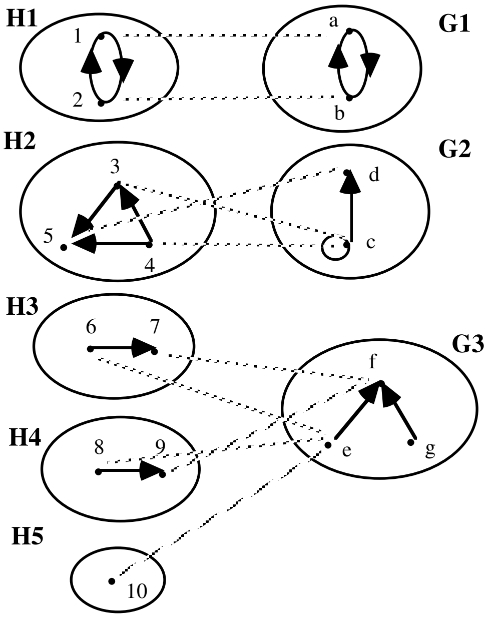


Fig. 2. Skeletons of $\mathcal{G}$ and $\mathcal{H}$ (from Fig. 1).

A morphism from a graph of graphs $\mathcal{G}$ to a graph of graphs $\mathcal{H}$ combines graph morphisms from graphs of $\mathcal{G}$ to those of $\mathcal{H}$, and a graph morphism from the skeleton of $\mathcal{G}$ to the skeleton of $\mathcal{H}$.

Definition 3. *Let $\mathcal{G} = (\{G_1, ..., G_k\}, W)$ and $\mathcal{H} = (\{H_1, ..., H_p\}, Y)$ be two graphs of C-graphs. A* ***morphism from $\mathcal{G}$ to $\mathcal{H}$*** *is a set $\Pi = \{\Pi_1, ..., \Pi_k\}$ of mappings, where for all i, Π_i is a (C-graph) morphism from G_i to $H_{s(i)}$, s being a mapping from $\{1, ..., k\}$ to $\{1, ..., p\}$, which satisfies: if $(x, G_i, G_j) \in W$ then $(\Pi_i(x), H_{s(i)}, H_{s(j)}) \in Y$.*

Figure 3 shows a (graph of graphs) morphism $\Pi = \{\Pi_1, \Pi_2, \Pi_3, \Pi_4, \Pi_5\}$ between $\mathcal{G}$ and $\mathcal{H}$ of Figure 1. Graph morphism induces a preorder (reflexive and transitive relation) over graphs, defined as follows: a graph G subsumes a graph H ($G \geq H$) if there exists a morphism from G to H. Similarly, a graph of graphs morphism induces a subsumption relation over graph of graphs, which is also a preorder.



The 5 graph morphisms are represented with hachured lines.

Fig. 3. A morphism from $\mathcal{H}$ to $\mathcal{G}$ (from Fig. 1).

2.2. Tree of Graphs

Nested CGs are "graphs of simple CGs" with a tree underlying structure, as defined below.

Definition 4. *A graph of graphs is a **tree of graphs** if its skeleton is a rooted tree.*

For instance, the graph of graphs $\mathcal{H}$ of Figure 1 is a tree of graphs. It is useful to give a property which can be seen as a direct definition, by structural induction, of a tree of graphs.

Property 1. *A tree of graphs is a graph of graphs defined as follows:*
(1) *a graph G with an empty set of triples, i.e. ({G}, ∅), is a tree of graphs with root G*
(2) *let A_1, ..., A_k be pairwise disjoint trees of graphs and (x_1, ..., x_k) be a k-tuple of nodes of a graph G (x_1...x_k are not necessarily distinct nodes). Let N be the set of* graphs appearing in the *A_i, added with G. Let W be the set of triples appearing in the A_i, added with triples (x_i, G, root of A_i) for each i. Then (N, W) is a tree of graphs with root G.*

The proof of this property is straightforward. Proving that any graph of graphs whose skeleton is a rooted tree belongs to the class defined by Property 1 can be done by induction on the skeleton depth. The reciprocal is proven by induction on the number of applications of (1) and (2).

Classical notions about trees — such as successor, root, depth, subtree induced by a node — are easily extended to trees of graphs, thus we use them with the same names and without definitions. We also use the representation of a tree by boxes within boxes. More precisely, a box is associated to each node *c* of the root of a tree of graphs. This box contains the disjoint boxes associated to the subtrees induced by the successors of *c*.

Let $\mathcal{A}$ and $\mathcal{A}'$ be two trees of graphs. A ***rooted morphism*** from $\mathcal{A}$ to $\mathcal{A}'$ is a morphism (in the sense of definition 3) which maps the root of $\mathcal{A}$ onto the root of $\mathcal{A}'$. Projection on NCGs is a rooted morphism. Property 1 leads to the following property which can be seen as a recursive definition of a rooted morphism.

Property 2. *Let $\mathcal{A}$ and $\mathcal{A}'$ be two trees of graphs. Let $D(\mathcal{A})$ and $D(\mathcal{A}')$ be the complex node sets of the roots of $\mathcal{A}$ and $\mathcal{A}'$. A rooted morphism from $\mathcal{A}$ to $\mathcal{A}'$ is a set of mappings $\{\Pi_0\} \cup \{\cup \{\Pi_c\}_{c \in D(\mathcal{A})}\}$ where:*

(1) Π_0 is a morphism from the root of $\mathcal{A}$ to the root of $\mathcal{A}'$

(2) for all c in $D(\mathcal{A})$, $\Pi_0(c) \in D(\mathcal{A}')$. Let $G_1, \ldots, G_k$ be the successors of c in $\mathcal{A}$. Then $\Pi_c = \{\Pi_1, \ldots, \Pi_k\}$, where each Π_i is a rooted morphism from the subtree induced by G_i to a subtree induced by a successor of $\Pi_0(c)$.

This property can be used for building an algorithm which computes a morphism (rooted or not) between two trees of graphs, whose complexity is polynomially related to the complexity of morphism computation on the class *C* [13].

2.3. Generalization Relation

Let us first recall that a *category*, say $\mathcal{C}$, is a class of objects, called $\mathcal{C}$-objects, together with a class of morphisms, called $\mathcal{C}$-morphisms, which are related in the following way (see for instance [14]).

(1) with each couple of objects *a* and *b* is associated a set of morphisms, $\mathcal{C}$*-morph(a, b),* such that each morphism of the $\mathcal{C}$-morphism class belongs to just one $\mathcal{C}$*-morph(a, b)*

(2) if $P \in \mathcal{C}$*-morph(a, b)* and $Q \in \mathcal{C}$*-morph(b, c),* there is a unique element of $\mathcal{C}$*-morph(a, c)*, denoted by *PQ*, called the composition of *P* and *Q*

(3) given $P \in \mathcal{C}$*-morph(a, b)*, $Q \in \mathcal{C}$*-morph(b, c)* and $R \in \mathcal{C}$*-morph(c, d),* $(PQ)R = P(QR)$

(4) to each object corresponds a morphism id_a of $\mathcal{C}$*-morph(a, a)* called the identity morphism such that, for any $P \in \mathcal{C}$*-morph(a, b)* and $Q \in \mathcal{C}$*-morph(b, a)*, $id_aP = P$ and $Qid_a = Q$.

It is straightforward to check that ordinary graphs provided with morphism, simple CGs provided with projection, graphs of graphs provided with morphism, nested CGs provided with projection (see part 3) are categories. In order to compare two models of nested CGs, or more generally two categories, we use the problem reduction notion of complexity theory (e.g. [15]). The following definition is obtained by considering the problem of the existence of a morphism between two objects, which is also the problem of subsumption checking.

Definition 5. *Let $\mathcal{C}$ and $\mathcal{D}$ be two categories.* ***$\mathcal{D}$ is more general than $\mathcal{C}$*** *if there is a mapping α from $\mathcal{C}$-objects to $\mathcal{D}$-objects, such that for any couple (a, b) of $\mathcal{C}$-objects one has: $\mathcal{C}$-morph(a, b) $\neq \emptyset$ if and only if $\mathcal{D}$-morph($\alpha(a)$, $\alpha(b)$) $\neq \emptyset$.*

In other words, $\mathcal{D}$ is more general than $\mathcal{C}$ when for all $\mathcal{C}$-objects a and b, a subsumes b if and only if $\alpha(a)$ subsumes $\alpha(b)$. If $\mathcal{D}$ is more general than $\mathcal{C}$, an algorithm for subsumption checking in $\mathcal{C}$ can be built using an algorithm for subsumption checking in $\mathcal{D}$ and an algorithm for computing α. Furthermore, if α is polynomially computable (i.e. for any object c of $\mathcal{C}$, $\alpha(c)$ can be computed with a worst-case complexity polynomial with respect to the size of c) and if there exists a polynomial algorithm for subsumption checking on $\mathcal{D}$, then a polynomial algorithm can be built for subsumption checking on $\mathcal{C}$. The generalization relation between categories is a preorder. Thus, it leads to an equivalence relation.

Definition 6. *Two categories are* ***equivalent*** *if each one is more general than the other one.*

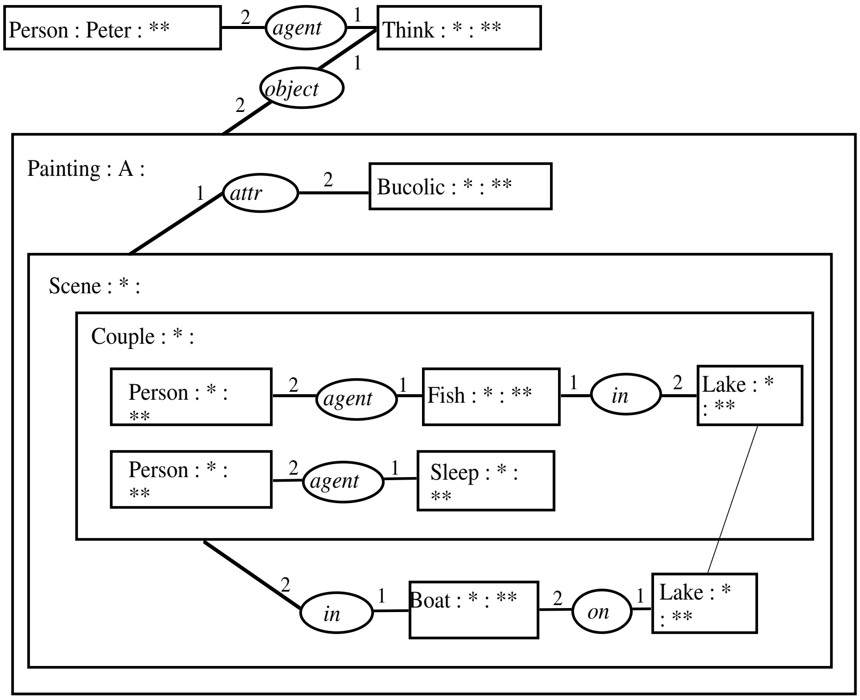
As far as we know, the generalization relation between categories has not been studied yet, neither the equivalence relation which is weaker than the classical notion of isomorphism between categories.

3. Nested Conceptual Graphs

All kinds of (positive) nested CGs we found in papers can be defined as trees of C-graphs where C is a class of simple CGs (e.g. [10], [16], [17], [12]; more generally, see ICCS proceedings). This is not surprising since trees precisely are the underlying structure of boxes within boxes. As the case may be, trees of SCGs may have labelled edges and some constraints on their structure. Projection between NCGs can be defined as a rooted morphism between trees of C-graphs, where C is a class of simple CGs. A subsumption relation on NCGs is induced by projection: a NCG G subsumes a NCG H if and only if there exists a projection from G to H. Using results on trees of graphs, it can be proven that subsumption checking on NCGs (without explicit coreference links, see part 3.5) is not more difficult than subsumption checking on simple CGs, in the following meaning: given an algorithm A for projection checking (or computing) on the class C of simple CGs, an algorithm for projection checking (or computing) on the corresponding class of NCGs can be built, whose complexity is polynomially related to the complexity of A. Furthermore, the generalization relation over categories (definition 5) provides a mean of comparing NCG classes. Thus, the notions introduced in the preceding part define a unified framework for formalizing and comparing NCG classes provided with a subsumption relation. In what follows, we give three examples of NCG classes.

3.1. Untyped NCGs

A class of NCGs has been introduced in [18]. In order to distinguish this class from the others, we call it *untyped NCG* in this paper. Labels of concept nodes are triples *(t, m, d)*. *t* and *m* respectively represent the type and the referent of the concept node, as in simple CGs. The third field represents a description of the concept node. Its value may be **, called the generic description, or a set of untyped NCGs. Therefore, an intuitive semantics of *(t, m, d)* is *"m* denotes an object with type *t* and *partial internal description d"*. For instance, the untyped NCG of Figure 4 may encode the knowledge "Peter is thinking about the painting A, which represents a bucolic scene: a couple in a boat on a lake, one person being fishing, the other one being sleeping".



The coreference link between the "Lake" nodes is ignored in this part. See part 3.5

Fig. 4. An untyped NCG

The graph inside the node A is not supposed to fully characterize the painting, that is why it is called a "partial" description. This description is also called "internal" because it corresponds to a "zoom" (following Sowa's idea in [10]) on the node A, which behave like a *glass box*. In contrast, an information such as "the painting A is from Renoir" could rather be considered as an external information. It would be added to the NCG outside the node A, which in this case becomes a *black box*.

Let us recall that a *support* [3] encodes basic ontological knowledge. Among other elements, it possesses a concept type set T_C, a relation type set T_R, and a set of individual markers I. T_C and T_R are partially ordered by an A-Kind-Of relation, denoted by ≤. T_C has a greatest element, denoted by *Universal.* T_R is partitioned into sets of relation types with the same arity.

Definition 7. ***Untyped NCG*** *can be defined as follows.*

(1) *A basic untyped NCG is obtained from a SCG 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, and a trivial bijection exists between basic untyped NCGs and SCGs).*

(2) *Let G be a basic NCG,* $(c_1, c_2, ..., c_k)$ *a k-tuple of nodes of G, not necessarily distinct, and* $G_1, G_2, ..., G_k$ *NCGs. The graph obtained by substituting* G_i *to the description ** of* c_i *for i = 1, ..., k is an untyped NCG.*

An untyped NCG G can be denoted by $G = (R, C, E, l)$ where R, C and E are respectively relation, concept, and edge sets of G, l is a labelling function of R and C such that for all $r \in R$, $l(r)=type(r) \in T_R$ and for all $c \in C$, $l(c) = (type(c), ref(c), Desc(c))$ with $type(c) \in T_C$, $ref(c) \in I \cup \{*\}$, $Desc(c)$ = ** or is equal to a set of untyped NCGs (in the latter case, c is said to be a complex node). Now, projection can be defined as follows.

Definition 8. Let $G=(R_G,C_G, E_G, l_G)$ and $H=(R_H,C_H, E_H, l_H)$ be two NCGs, with complex node sets D_G and D_H. A **projection** from G to H is a set of mappings $\Pi=\{\Pi_0\} \cup \{\cup \{\Pi_c\}_{c \in D_G}\}$, which satisfies:
(1) for all edge rc of E_G, $\Pi_0(r)\Pi_0(c)$ is an edge of E_H, and if c is the *ith* neighbor of r, then $\Pi_0(c)$ is the *ith* neighbor of $\Pi_0(r)$;
(2) for all node r of R_G, $type(r) \geq type(\Pi_0(r))$;
for all node c of C_G, with $l(c)=(t,m,d)$, let $l(\Pi_0(c))=(t',m',d')$, then: $t \geq t'$, $m \geq m'$ (i.e. a generic marker may be specialized into an individual one), and if $c \in D_G$, say $d=\{G_1, ..., G_k\}$, then $\Pi_0(c) \in D_H$, and $\Pi_c = \{\Pi_1, ..., \Pi_k\}$, where each Π_i is a projection from the NCG G_i to a NCG of d'.

It is straightforward to prove that untyped NCGs are trees of SCGs. The third field of a concept node c, $Desc(c)$ the description of c, is the set of subtrees induced by the successors of c. Figure 5 shows the NCG of Figure 4 represented as a tree of SCGs.

Property 3
An untyped NCG can be defined in an equivalent way as a tree of SCGs. Projection between untyped NCGs is then a rooted morphism between trees of SCGs.

3.2 Typed NCGs

In the untyped NCG model, a nesting does not have a semantics more specific than "the graph inside the concept node represents a partial internal description of the entity represented by the node". In Figure 4 for instance, the same notion is used for representing the nesting of the two persons who *constitute* a couple, the nesting of the couple in the *description* of the scene, and the nesting of the scene *represented* on the painting. It appeared in applications that a mean of specifying the nesting semantics was desirable. A way of solving this problem consists in adding nesting types to the untyped NCGs. A new type set T_N, called the nesting type set, is added to the support. T_N is partially ordered by an A-Kind-Of relation, as the concept and relation type sets, and possesses a greatest element called *Description*.

Definition 9. *Typed NCGs can be defined as follows.*
(1) *A basic typed NCG is a basic untyped NCG.*
(2) *Let G be a basic NCG and (c_1, c_2, ..., c_k) be a k-tuple of non necessarily distinct nodes of G. Instead of substituting a graph to the third field of each c_i, one substitutes a couple (n_i, G_i), where n_i is a nesting type and G_i is a typed NCG. The graph obtained by substituting (n_i, G_i) to the description ** of c_i for i = 1, ..., k is a typed NCG.*

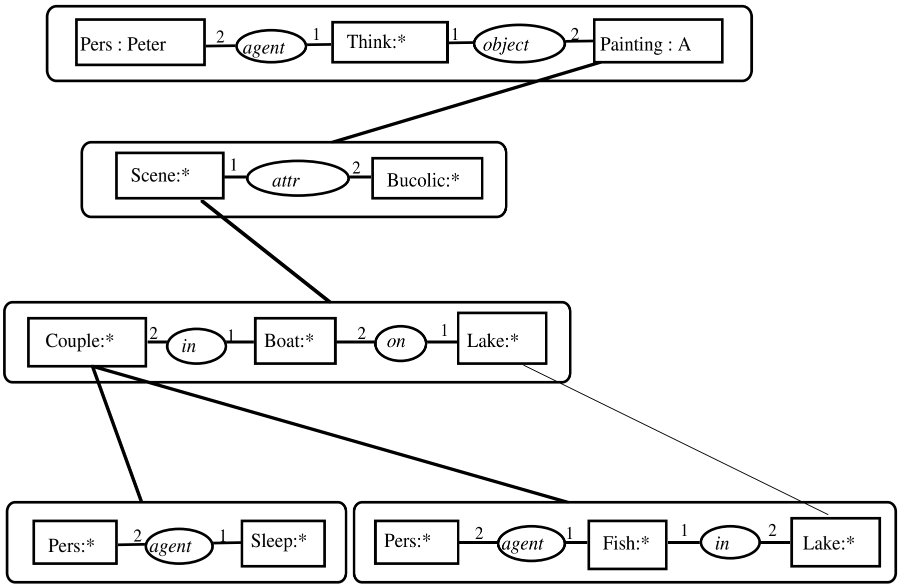
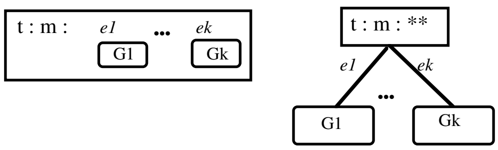


Fig. 5. An untyped NCG represented as a tree of graphs



On the left: a concept node with label *(t, m, d = {(e_1,G_1) ,(e_2, G_2),..., (e_k, G_k)})*
On the right: the tree induced by this node

Fig. 6. Representation of a typed nesting

Property 4. *A typed NCG is a tree of SCGs, where each edge is labelled by a nesting type.*

Figure 7 shows a typed NCG represented as a tree of SCGs with labelled edges. It represents the "painting A" example (see Figure 4 for a comparison with untyped NCGs). *Representation*, *Description* and *Component* are nesting types, with *Representation* ≤ *Description* and *Component* ≤ *Description*.

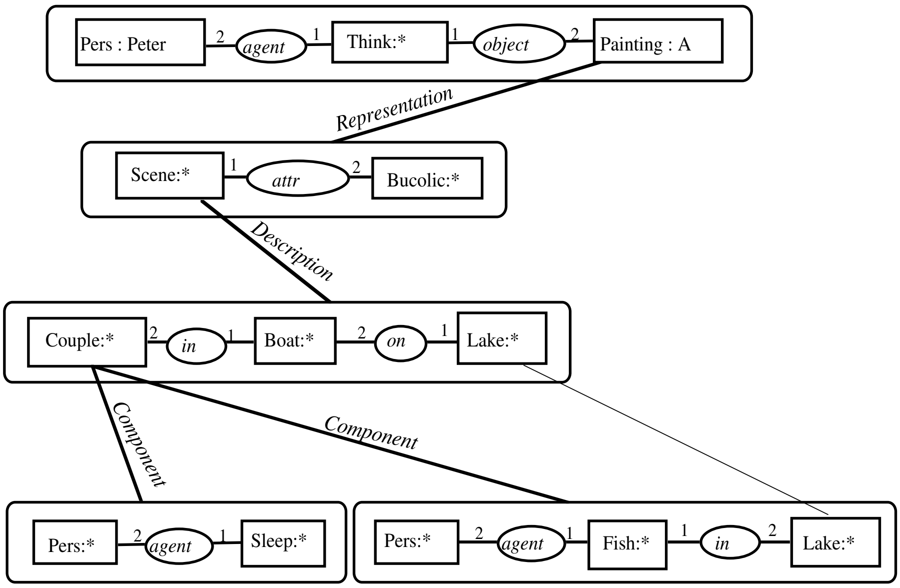


Fig. 7. A Typed NCG represented as a tree of graphs

Definition 10. *A **projection** between typed NCGs is a projection between the underlying untyped NCGs which may decrease the labels of the tree edges.*

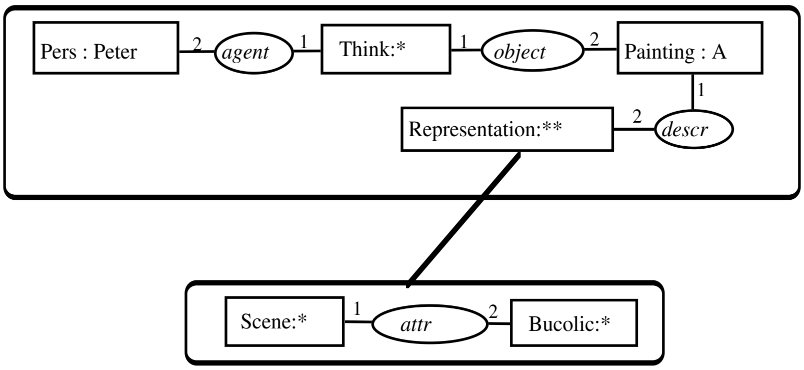
3.3. 2-NCGs

In this paragraph a minor variant of the most used model of NCGs (e.g. [10]) is translated in our framework. In [10] two kinds of concept types are considered, *complex* and *simple* types. Nodes with complex type ("complex nodes") can have a graph for referent, and nodes with simple type ("simple nodes") can only have an atomic marker for referent. In the following formalization, we keep this distinction but any concept node must have a marker, and graphs are not referents but descriptions. Thus, in this model three pieces of information are associated to a complex node: its type, its referent which is a marker, and its description which is a set of graphs, possibly empty (note that this notion of a complex node does not exactly correspond to the notion of a complex node in previous NCGs, which assumes a non-empty description). A simple node only has a type and a referent.
Let us first define 2-SCGs, from which trees of 2-SCGs will be built. A 2-*SCG* is a SCG built on a support, called a 2-support, which is more structured than the usual ones. In a 2-support the concept type set T_C, is composed of two disjoint sets, a simple concept type set, denoted T_1, and a complex concept type set, denoted T_2. T_1 has a greatest element called *Entity*, and T_2 has a greatest element called *Graph*. *Entity* and *Graph* are covered by *Universal*, the greatest element of T_C. No elements of T_1 are comparable with elements of T_2. The set of individual markers I is then

bipartitioned into I_1 and I_2, which are marker sets for simple and complex concept nodes respectively. Two generic markers are considered, * for a simple node and ** for a complex node. These two generic markers are incomparable , * is greater than any element of I_1, and ** is greater than any element of I_2.

Definition 11. *An* ***untyped*** *2-NCG is a tree of 2-SCGs in which only complex concept nodes may have a description. Projection between untyped 2-NCGs is a rooted morphism between trees of 2-SCGs.*

NCGs of [10] can be seen as trees of particular classes of 2-SCGs, in which a support contains a particular binary relation type, say *descr*, with signature *(Entity, Graph)*, such that any relation type with an argument of type T_2 is less than *descr* (and therefore it is binary and its first argument is of type T_1). The second argument may be seen as a partial description of the first one. Note that introduction of *descr* provides another way of specifying nesting semantics. The example of Figure 7 could then be represented as partially shown in Figure 8, where the concept type *Representation* is a specialization of *Graph*. Or, in a dual way, *descr* could be specialized into a relation type *repr* . Then [Painting: A] -1-*(descr)*-2-[Representation: G] would be translated into [Painting: A] -1-*(repr)*-2-[Graph: G].



(Partial representation of the "painting A" example)

Fig. 8. A 2-NCG represented as a tree of graphs

3.4. Equivalence between classes of NCGs

Other classes of NCGs may be built in the same way as in previous part. For instance, *typed 2-NCGs* can be introduced, by adding nesting types to 2-NCGs or, equivalently, labels on the edges of trees. In [13] other classes are considered. An important point is that all these classes can be defined as trees of C-graphs for some class C of simple CGs. It is then easy to provide them with a projection which is a rooted morphism between these trees. The generalization relation introduced in part 2 of this paper allows to show that all these notions are only slight variations of a unique notion.

Theorem. *Typed or untyped NCGs, typed or untyped 2-NCGs, are equivalent classes.*

Proof. Let *UNCG(S), TNCG(S),* 2-*NCG(S)* be respectively classes of *untyped NCGs*, *typed NCGs* and *untyped* 2-*NCGs* on a support S.

Trivially, a graph G of 2-*NCG(S)* can be represented as a graph $\gamma(G)$ of $UNCG(\gamma(S))$, with $\gamma(S)$ is S where the distinction between $T1$ and $T2$ is forgotten. $\gamma(G)$ is obtained from G by replacing labels (t, m) of simple concept nodes with $(t,m,**)$, i.e. by adding an empty description **.

A graph G of *UNCG(S)* can be represented by a graph $\alpha(G)$ of $TNCG(\alpha(S))$, where $\alpha(S)$ is obtained from S by adding a nesting type set restricted to a unique type, *Description*. $\alpha(G)$ is obtained from G by simply labelling all edges of the tree with *Description*.

A graph G of *TNCG(S)* can be represented by a graph $\beta(G)$ of $UNCG(\beta(S))$, where $\beta(S)$ is obtained from S by adding to the concept type set T_C of S a subset T in bijection with T_N and a new maximal type, and by deleting T_N. $\beta(G)$ is obtained from G by transforming labelled edges of the tree into concept nodes with types in T. More specifically, any edge in the tree of G, say from a node c to a graph H and labelled by a nesting type n, is replaced with the following construction: a new graph K with a unique concept node c' of type n is created and two edges are added to the tree, one from c to K and one from c' to H.

A graph G of *TNCG(S)*, with $S = (T_R, T_C, T_N, I)$, can be represented by a graph $\mu(G)$ of $2\text{-}NCG(\mu(S))$, where $\mu(S)$ is built as follows: the simple concept type set T_1 is in bijection with T_C; the complex type set T_2 is restricted to *{Graph}*; the relation type set is in bijection with the union of T_R and T_N, where the *Description* nesting type corresponds to the *descr* binary relation ; the marker set for simple nodes is in bijection with I and there are no individual markers for complex nodes. $\mu(G)$ is obtained from G by transforming each concept node $c = [t : m : d]$ into a simple concept node $c' = [t : m]$, whose neighbors are the images of c neighbors; and, if d is not the empty description, say $d = \{(e_1, G_1), \ldots, (e_k, G_k)\}$, then, for each i a complex concept node $c'_i = [Graph : ** : \mu(G_i)]$ is created and a new relation node of type e_i links c' to c'_i.

A transformation similar to α (resp. β) can be used for transforming an *untyped 2-NCG* into a *typed 2-NCG* (resp. a *typed 2-NCG* into an *untyped 2-NCG*).

It is straightforward to check that transformations γ, α, β and μ preserve morphisms and rooted morphisms. ❑

3.5. Coreference links.

Previous NCG models can be extended with coreference links. Let us formally define coreference links, by way of an equivalence relation, say **coref**, over the set of concept nodes of all SCGs appearing in the tree associated with a NCG. Two distinct concept nodes are said to be coreferent if they belong to the same equivalence class. Concept nodes with the same individual marker necessarily belong to the same equivalence class (they are related with an implicit coreference link). And concept nodes with different individual markers necessarily belong to distinct classes. By definition, a node is coreferent with itself.

There are often constraints on labels of coreferent nodes ensuring that these nodes can be merged. In this case, coreference links internal to a simple CG do not present a fundamental interest, since an equivalent simple CG ("equivalent" in an intuitive meaning, and also according to logical semantics) without coreference link can be built in merging all coreferent concept nodes. But, when a coreference link relates two concept nodes, say *c* and *c'*, from *distinct* simple CGs composing a NCG $\mathcal{G}$, which do not belong to the description of the same concept node of $\mathcal{G}$, the merging of *c* and *c'* is never possible.

Projection on simple CGs provided with coreference links, say Π from *G* to *H*, must preserve the *coref* relation, i.e. for all concepts nodes *c* and *c'* of *G*, if *coref(c,c')* then *coref(*Π*(c),*Π*(c')*. In other words, coreferent concept nodes either have the same image or distinct coreferent images. And the definition of a projection between NCGs provided with coreference links, say Π from $\mathcal{G}$ to $\mathcal{H}$, is added with the same constraint: for all concepts nodes *c* and *c'* appearing in $\mathcal{G}$, if *coref(c,c')* then *coref(*Π*(c),*Π*(c')*.

Note that the coref relation could be introduced in the general framework, as an equivalence relation on the set of nodes appearing in a graph of graphs.

4. Conclusion

All nested CG models considered in this paper generalize the simple CG model, in the sense that they involve objects which are generalizations of simple CGs and reasoning on these objets is based on a graph operation (projection) which generalizes that used for simple CGs. An important point, which we did not discuss, is that of the logical semantics of nested CGs. And the corollary question is whether projection is sound and complete with respect to this semantics.

This point has been studied for untyped NCGs provided with coreference links, with two approaches. In [19] a "non-classical" logic is defined, with a sequent calculus working on these formulas. Projection is proven to be sound and complete with respect to this calculus, i.e. there exists a projection from a NCG *G* to a NCG *H* if and only if the sequent leading from the formula associated to *G* to the formula associated to *H* is derivable. [20] defines a semantics in first-order logic. And projection (from *G* to *H*) is proven to be sound and complete with respect to implication (when *H* has a normal form). Equivalence of NCG classes allows transfer of these semantics to other NCG classes. For 2-NCGs, this transfer is “natural” since the transformation from a 2-NCG to an untyped NCG is trivial. On the other hand, for typed NCGs a new study seems to be more appropriate.

There are possible links between NCGs and contextual reasoning [21], although contextual reasoning is not our goal here. But both logical semantics mentioned above do not seem to be close to one of the logical formalizations of contextual reasoning (e.g. [22], [23]). However, a more in-depth comparison has to be done.

Acknowledgments

We would like to thank G. Kerdiles and E. Salvat for their comments on a previous version of this paper, and specially G. simonet for her suggestions that greatly improved clarity of this work.

References

1. L. K. Schubert, Semantic Networks are in the eye of the Beholder, in *Principles of Semantic Networks*, J. F. Sowa (ed), Morgan Kaufmann, 95-108, 1991.
2. J. F. Sowa, *Conceptual Structures — Information Processing in Mind and Machine*, Addison-Wesley, 1984.
3. M. Chein, and M.-L. Mugnier, Conceptual Graphs : fundamental notions, *Revue d'intelligence artificielle*, vol. 6, n°4, 365-406, 1992.

4. O. Haemmerlé, *La plate-forme CoGITo : manuel d'utilisation*, RR LIRMM 95012, 1995, 52 pages.
5. O. Guinaldo, *CoGITo v*3.3 *: module SGBD de graphes conceptuels*. RR LIRMM 96022, 1996, 10 pages.
6. C.Bos, and B.Botella, P.Vanheeghe, *Modelling and Simulating Human Behaviours with Conceptual Graphs*, R. R. Isen, 1996.
7. C.Bos, and B.Botella, Modelling Stereotyped Behaviours in Human Organizations, in *proceedings 7th Workshop on Knowledge Engineering: Methods and Languages*, Milton Keynes, United Kingdom, 1996.
8. D. Genest, *Une utilisation des graphes conceptuels pour la recherche documentaire*. Mémoire de DEA d'informatique, LIRMM & Université Montpellier II, Juin 1996.
9. J. F. Sowa, Conceptual Graphs as a universal knowledge representation, in *Semantic Networks in AI*, F. Lehmann (ed), Pergamon Press, 75-94, 1992.
10. J. F. Sowa, Logical foundations for representing object-oriented systems, *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 5, 1994.
11. B.C. Ghosh, *Conceptual Graph Language*, Ph.D. Thesis, Asian Institute of Technology, Bangkok, 1996.
12. B.C. Ghosh, and V. Wuvongse, Computational Situation Theory in the Conceptual Graph Language, in *proceedings of ICCS'96,* 188-202, Lecture Notes in AI, 1115, Springer Verlag, Berlin, 1996
13. M. Chein, and M.-L. Mugnier, *Quelques classes de graphes emboîtés équivalentes*, R.R. LIRMM 96-063, 1996, 64 pages.
14. P. M. Cohn, *Universal Algebra*, Harper & Row, New-york.
15. M.R. Garey, and D.S. Johnson, *Computers and intractability — A guide to the Theory of NP-Completeness*, W.H. Freeman and Co, 1979.
16. A. Nazarenko, Representing natural language causality in conceptual graphs: the higher order conceptual relation problem, in *proceedings of ICCS'*93, 205-222, Lecture Notes in AI, 699, Springer Verlag, Berlin, 1993.
17. J.P. Dick, Using Contexts to Represent Text, in *proceedings of ICCS'*94*,* 196-213, Lecture Notes in AI, 835, Springer Verlag, Berlin, 1994.
18. M.-L. Mugnier, and M. Chein, Représenter des connaissances et raisonner avec des graphes, *Revue d'intelligence artificielle*, numéro spécial "graphes conceptuels", vol. 10, n°1, 7-56, 1996.
19. A. Preller, M.L. Mugnier, and M. Chein, A Logic for Nested Graphs, *Computational Intelligence Journal*, (to be published, reference number CI 95-02-558).
20. G. Simonet, *Une sémantique logique pour les graphes emboîtés*, R.R. LIRMM 96047, 1996.
21. J. McCarthy, Notes on Formalizing Context, in *Proceedings of IJCAI'*93, 555-560, 1993.
22. R. V. Guha, *Contexts: a formalization and some applications*, PhD Thesis, Stanford University, 1991.
23. S. Buvac, Quantificational Logic of Context, in *Proceedings AAAI'*96, 600-606,1996.