Chapter 9
Nested Conceptual Graphs

Overview

The nested conceptual graph model presented in this chapter is a direct extension of basic or simple conceptual graphs able to represent notions such as internal and external information, zooming, partial description of an entity, or specific contexts. This model also allows reasoning while taking a tree hierarchical structuring of knowledge into account. Nestings are represented by boxes. A box is an SG and, more generally, a box is a typed SG. In full conceptual graphs, a box represents the negation of the graph inside the box. Thus, for differentiating these negation boxes from the boxes used in this chapter, these boxes are usually called "positive" boxes. Nevertheless, since the only kind of boxes considered hereafter are positive boxes, we omit the term "positive."

In Sect. 9.1 different notions representable by nested conceptual graphs are presented. In Sect. 9.2, we introduce *Nested Basic Conceptual Graphs (NBGs)*, whose boxes consist of BGs. *Nested Conceptual Graphs (NGs)*, which extend NBGs with coreference links, are presented in Sect. 9.3. Coreference links can relate concept nodes of the same box (thus boxes become SGs) but also of different boxes. Coreference links in nested graphs are more difficult to manage than in simple graphs. Indeed, since boxes can represent contexts, it is generally irrelevant to merge all nodes of a coreference class into a single node. In Sect. 9.4, we define *graph types*, *typed SGs*, which are SGs with a graph type, and *Nested Typed Conceptual Graphs (NTGs)*, which generalize NBGs by typing the boxes. All of these nested graphs classes are provided with homomorphism. The FOL semantics Φ introduced for SGs is generalized to NTGs in Sect. 9.5 and a homomorphism soundness and completeness theorem is stated. As this semantics is a formula of the positive, conjunctive and existential fragment of FOL, nested and non-nested CGs are somewhat equivalent. Finally, we build a mapping *ng2bg* from nested to non-nested CGs which preserves homomorphisms. This mapping *ng2bg* shows, in another way than through logical semantics, that nested and non-nested CGs have the same descriptive power. It is easy to implement *ng2bg* and this avoids the construction of specific nested graph homomorphism algorithms.

Nevertheless, from a user viewpoint, NTGs are interesting whenever knowledge is intrinsically hierarchical, and when reasonings must follow the hierarchical structure, because in an NTG the hierarchy is explicitly and graphically represented. Nested graphs can also be interesting whenever large graphs have to be manually constructed, as the separation of levels of reasoning increases efficiency and clarity when extracting information.

9.1 Introduction

Let us give a flavor of different knowledge representation situations which are relevant to the conceptual graph model presented in this chapter. Consider, for instance, representing information about a cottage. It is possible to distinguish *internal* from *external* pieces of information about this cottage. The owner's name can be considered as an external piece of information concerning the cottage, whereas the distribution of rooms, the plan of the cottage, is internal information. In information retrieval, the ISBN number of a book can be considered as an external piece of information about that book, whereas the subject of the book is an internal piece of information. In these examples, the internal information can also be called description (of the cottage or book), and more precisely *partial description* of the given entity.

Zooming is a related notion. Let us again consider the cottage example. Having the land registry position of the cottage, one may want to zoom into the cottage, e.g., to determine the cottage plan, which can be considered as internal information about the cottage. Or, at a deeper level, having the cottage plan, one may want to know the furniture distribution in a specific room. In the book example, zooming can consist of obtaining the content of a chapter from a reference to this chapter (e.g., from the number of this chapter).

The knowledge model presented in this chapter can also be related to the *context* notion. An *informational context* can be defined as the (cultural, historical, social, geographical, etc.) surroundings or circumstances of a piece of information that are important to understand the meaning of this piece of information. For instance, in the previous cottage example, the context of the furniture distribution (in a room) is precisely the room having this furniture distribution, the context of the content of a book chapter is precisely this chapter, and so on.

The notion of context is close to the two previous notions of external versus internal pieces of information, and to zooming. Indeed, zooming takes its full meaning when one knows the origin of the zooming, and an internal piece of information takes its full meaning when this origin, which can also be an internal piece of information, is known. In the forthcoming model, only very simple contexts can be represented. Indeed, a context will be represented by a path of (nested) concept nodes. Thus, considering partial description or zooming as an intuitive meaning of what we aim to represent, seems as relevant as the context.

A partial description can be included in another partial description, thus a recursive model is proposed, and the entities are represented by a hierarchical structure. Similarly to the classical notion of "boxes within boxes" used in document processing, we define a model that consists of "graphs within graphs." Briefly said, our model consists of rooted trees of (typed) SGs, and reasoning is based on SG homomorphisms respecting the tree structure.

9.2 Nested Basic Graphs (NBGs)

Let us consider the graph *G* in Fig. 9.1, expressing that "a drawing has been made by the boy Paul." Suppose one wants to add two pieces of information to *G*. First,

Fig. 9.1 A basic conceptual graph

"the drawing is on a table," and secondly "the drawing represents a green-coloured train." These pieces of information can be considered to be of different sorts. The first one can be considered as *external* information about the drawing, i.e., it is a fact concerning the drawing taken as a whole, i.e., a black box, with what is drawn being irrelevant. On the other hand, the fact that a green train is represented on this drawing can be considered as *internal* information, or a *(partial) description*, of the drawing itself. If one wants to build a representation of these facts while keeping this difference of status, then one can add "the drawing is on a table" at the same level as *G*, and the fact that the drawing represents a green train can be put inside the node representing the drawing. This latter piece of information can be obtained by zooming on the drawing node, which is then considered as a glass box. After adding the fact that the drawing in Fig. 9.1 is on a table, and by zooming on the node "Drawing," the nested graph *NG* represented Fig. 9.2 is obtained. It can also be said that the piece of information nested in a node is relevant within the context represented by this node. Thus, the context is represented by a concept node, or more precisely by a path of concept nodes.

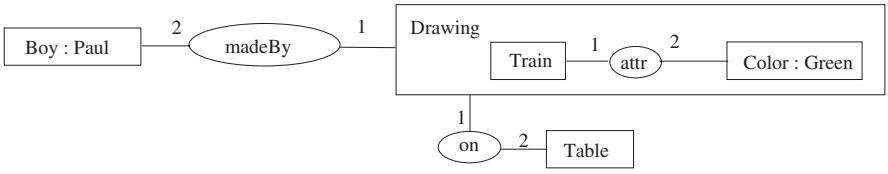

Fig. 9.2 A nested conceptual graph

Drawings of nested graphs can be quite difficult to read. A dynamic device, e.g., a graphical screen, allows nice vizualization of nested graphs by travelling level by level within the graph, i.e., by zooming in and out. For instance, for representing the nested graph in Fig. 9.2, a first image can show the graph without the description of the drawing, a graphical mark indicating that the node corresponding to the drawing has a non-empty description (as in Fig. 9.3). By zooming on a node with a non-

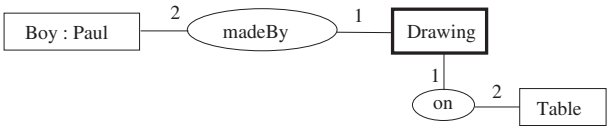

Fig. 9.3 A nested conceptual graph before zooming

empty description, the description graph appears, and the rest of the graph is shaded off (as in Fig. 9.4).

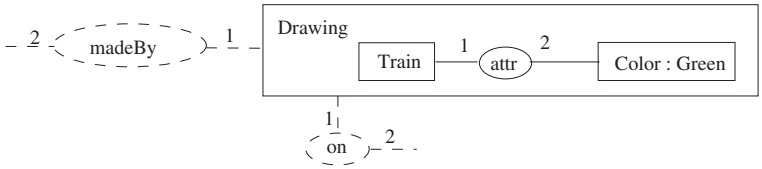

Fig. 9.4 A nested conceptual graph after zooming

We hereafter propose two equivalent definitions of nested graphs corresponding to different viewpoints of the same model. The first definition is a recursive definition obtained by adding a third field, called a *description*, to the concept node labels of a basic conceptual graph (BG), and in the second definition the recursive facet is explicitely represented by a tree. The first definition is well fitted for a graphical user interface since it corresponds to the zoom viewpoint. The second definition facilitates the drawing of nested graphs on a sheet of paper because a nested graph is basically a tree of SGs.

Definition 9.1 (Nested Basic Graph NBG).

- An *elementary* NBG G is obtained from a normal BG H by adding a third field to the label of each concept node c equal to $**$. The set of boxes of G is $boxes(G) = \{H\}$ and the complete concept node set of G is $X_G = C_H$. A trivial bijection exists between elementary NBGs and normal BGs (when no ambiguity occurs we do not distinguish between them).
- Let H be an NBG and D an elementary NBG. The graph G obtained from H by substituting D for the third field $**$ of a concept node c in X_H is an NBG. $boxes(G) = boxes(H) \cup boxes(D)$, and $X_G = X_H \cup X_D$.

- The third field of any concept node $c \in X_G$ is called the *description* of c and is denoted $Descr(c)$.

A BG appearing in the construction of an NBG G, i.e., an element of $boxes(G)$, is called a *box* of G.

An NBG is denoted $G = (C_G, R_G, E_G, l_G)$, where C_G, R_G, E_G are respectively the concept, relation and edge sets of the first elementary NBG, denoted $root(G)$, used in the construction of G. l_G is the labeling function obtained from the labeling function of $root(G)$ by adding a third field with value $Descr(c)$ to the labeling of each concept node $\in C_G$.

We distinguish the set C_G of concept nodes of an NBG G—i.e., the set of nodes of $root(G)$—from the set X_G of all concept nodes appearing in G, i.e., the set of concept nodes of all boxes of G.

We explain in Sect. 9.3 why we consider normal BGs in the previous definition. Recall that we consider normal BGs and normal SGs as identical objects (cf. Sect. 3.5).

It is important to note (for the forthcoming definitions of $Tree(G)$ and X_G) that if a BG or an NBG K is used several times in the construction of a NBG G, we consider that several copies of K (and not several times the graph K itself) are used in the construction of G. Note also that a description is not a type definition (cf. Chap. 8). First, a description applies to a specific concept node, whereas a graph defining a type applies to all concept nodes of this type. Secondly, a type definition provides a characterization of a type, i.e., it describes necessary and sufficient conditions for any object to belong to the type, whereas a description is only a partial information about an object.

Any NBG G has an associated tree $Tree(G)$ whose nodes are in bijection with the boxes of G. Before defining $Tree(G)$, let us introduce the notion of a tree of BGs.

Definition 9.2 (Tree of BGs). A *tree of BGs* is a labeled rooted tree $T=(V_T, U_T, l_T)$, such that:

- V_T, the node set of T, is in bijection with a set of pairwise disjoint normal BGs $\{G_1, \ldots, G_k\}$. For any $i = 1, \ldots, k$, the node associated with G_i is labeled G_i (the set $\{G_1, \ldots, G_k\}$ can be identified with V_T).
- For any arc (G_i, G_j) in U_T, $l_T(G_i, G_j)$ is a concept node c in G_i, and such a labeled arc is also denoted (G_i, c, G_j).
- All labels are distinct, i.e. a concept node c appears at most once as an arc label.

The mapping $Tree$ that assigns a tree of BGs to any NBG is defined as follows.

Definition 9.3 (Tree(G)). Let G be an NBG, the mapping $Tree$ is defined as follows. If G is an elementary NBG, then $Tree(G)$ is restricted to a single node labeled G. If G is the NBG obtained from a NBG H by substituting D to $**$, which is the $Descr$ of the concept node c in H, then $Tree(G)$ is built from $Tree(H)$ by adding a node labeled D successor of the node labeled K, and containing c, in $Tree(H)$. The label of (K, D) is c.

The root of $Tree(G)$ is $root(G)$. Note that for any NBG G, (J,K) is an arc in $Tree(G)$ labeled c if and only if J and K are nodes in $Tree(G)$ (i.e. are boxes of G), c is a concept node in J, and K is the label of the root of the tree associated with $Descr(c)$, i.e., the description graph of c. Otherwise said, the existence of an arc (J,c,K) in $Tree(G)$ expresses the fact that: "J and K are two boxes of G, c is a concept node in J, and K is the label of the root of $Tree(Descr(c))$."
Example. The boxes and tree of the NBG in Fig. 9.2 are represented in Fig. 9.5.

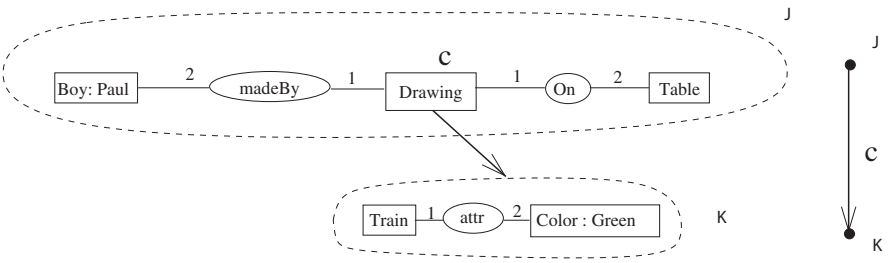

Fig. 9.5 The "tree of boxes" view of the graph in Fig. 9.2

Figure 9.6 presents a more complex example of an NBG. Its boxes are represented in Fig. 9.7, where G_1 is the root, $G_2 = Descr(d)$, $G_3 = Descr(c)$, $G_4 = Descr(e)$, $G_5 = Descr(f)$, $G_6 = Descr(i)$, $G_7 = Descr(h)$, $G_8 = Descr(l)$, $G_9 = Descr(n)$, and its tree is represented in Fig. 9.8.

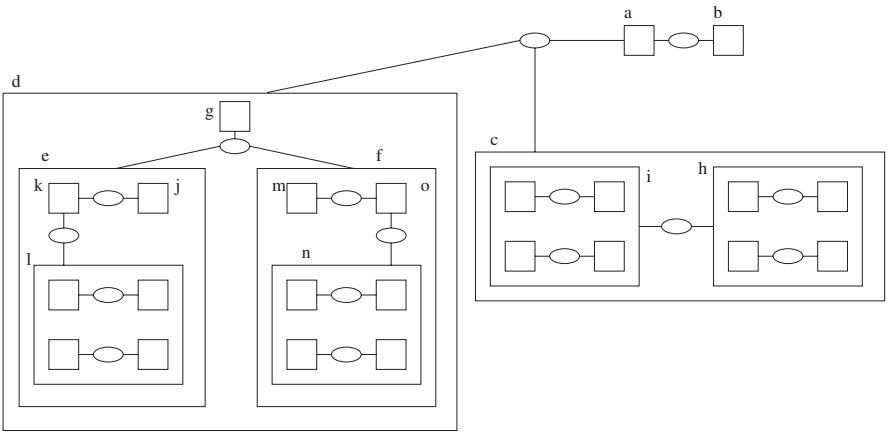

Fig. 9.6 An NBG

The following property is immediate:

Property 9.1. The mapping *Tree* is a bijection from NBGs over a vocabulary $\mathcal{V}$ to the trees of BGs over the same vocabulary $\mathcal{V}$.

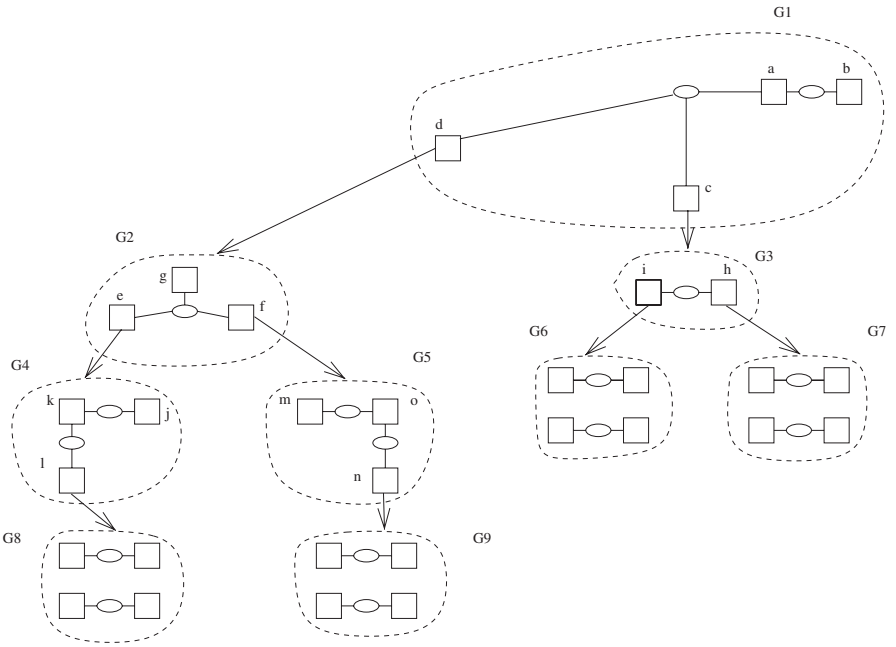

Fig. 9.7 The boxes of the graph in Fig. 9.6

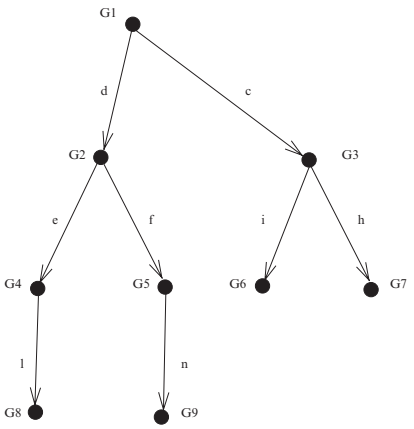

Fig. 9.8 The tree of the graph in Fig. 9.6

Thus, Definition 9.1 and Definition 9.2 can be indiscriminately used.

Definition 9.4 (Depth).

- The *depth* of an NBG G is the depth of $Tree(G)$, i.e., the maximum number of arcs of a path beginning at the root.
- The *depth* of a node x in a NBG G is the depth of the vertex of $Tree(G)$ to which it belongs, i.e., if x is in $root(G)$ then $depth(x) = 0$, and if x is in K and (J,c,K) is an arc of $Tree(G)$, then $depth(x) = depth(J) + 1$.

For instance, in Fig. 9.6 the depth of the nodes m and k is 2, whereas the depth of f and h is 1, and the depth of d and a is 0 (see also Fig. 9.7, where the depth of a node in a box is the number of arcs of a path from the root to the box containing that node).

A *complex concept node* is a node c in X_G with a non-empty $Descr(c)$, i.e., different from $**$. Such a node c is also called a context, and more precisely it is called *the context of* $Descr(c)$.concept!complex

A homomorphism from an NBG G to an NBG H is defined using the tree definition of NBGs. It is composed of an ordinary tree homomorphism π_0 from the rooted tree $Tree(G)$ to the rooted tree $Tree(H)$, and by a set of (BG) homomorphisms π_i from any box G_i of G to the box $\pi_0(G_i)$ of H.

Definition 9.5 (NBG homomorphism). Let G and H be NBGs and $Tree(G) = (V_G, U_G, l_G)$ and $Tree(H) = (V_H, U_H, l_H)$ be their associated rooted trees. An *NBG homomorphism* from G to H is a pair $\pi = (\pi_0, (\pi_{G_1}, \ldots, \pi_{G_l}))$, where $V_G = \{G_1, \ldots, G_l\}$, which satisfies:

- π_0 is a (tree) homomorphism from $Tree(G)$ to $Tree(H)$, i.e., a mapping from V_G to V_H, such that, if (J,K) is in U_G, then $(\pi_0(J), \pi_0(K))$ is in U_H, and $\pi_0(root(G)) = root(H)$,
- $\forall K \in V_G, \pi_K$ is a (BG) homomorphism from (the BG) K to (the BG) $\pi_0(K)$,
- if $l_G(J,K) = c$, then $l_H(\pi_0(J), \pi_0(K)) = \pi_J(c)$.

An example is given Fig. 9.9, where each arrow represents a homomorphism from the BG origin of the arrow to the BG extremity of the arrow.

A homomorphism π from an NBG G to an NBG H naturally induces a mapping (also noted π for simplicity) from the set of all nodes appearing in G to the set of all nodes appearing in H. Let x be a node in a box K of G, $\pi(x) = \pi_K(x)$.

One can extend the homomorphism definition by removing the condition that the root of the source tree has for image the root of the target tree. This extended definition is a bit more general since the root of the first tree can be mapped to any node of the target tree. In this case, only the relative depth is respected, more precisely, if the root of $Tree(G)$ is mapped to a box of depth k in $Tree(H)$ then, for any concept c of G, $depth(\pi(c)) = depth(c) + k$. Let us give an example.

Example. In Fig. 9.10, G_1 and G_2 are two facts and Q_1, Q_2 and Q_3 are three queries. Q_1 represents the question "Is there a train in the context of a thing?" Q_2 represents the question "Is there a train?" and Q_3 represents the question "Is there a train and

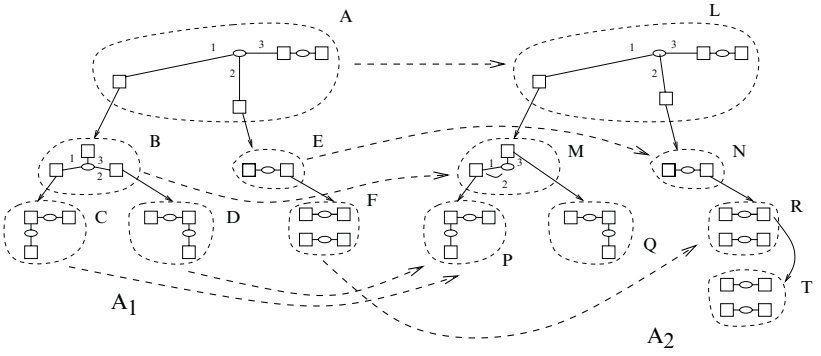

Fig. 9.9 An example of an NBG homomorphism

a boy?" With the definition 9.5, G_1 answers NO to Q_1, YES to Q_2 and YES to Q_3, and G_2 answers YES to Q_1, NO to Q_2 and NO to Q_3. With the extended definition, G_1 answers NO to Q_1, YES to Q_2 and YES to Q_3, and G_2 answers YES to Q_1, YES to Q_2 and NO to Q_3. Both definitions disagree on the answer given by G_2 to Q_2: with the first definition, the answer is NO because there is no train at the first level, and with the extended definition it is YES, and it could be added "in the context of a drawing."

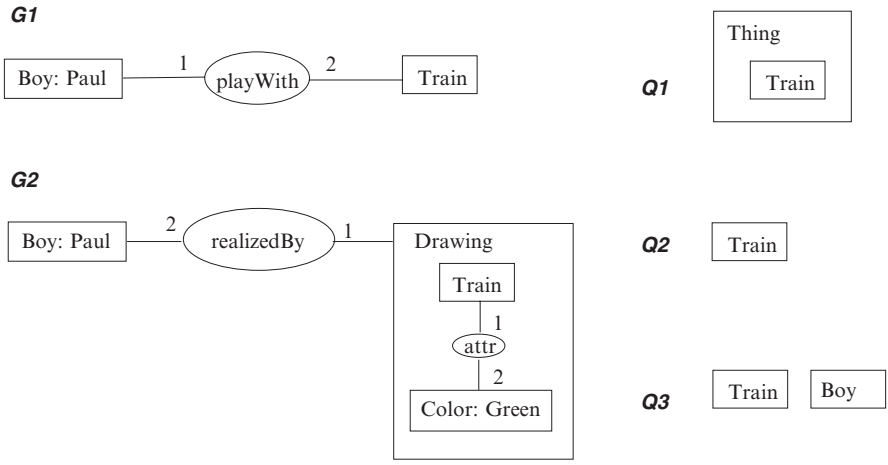

Fig. 9.10 Facts and queries

As NBG homomorphisms preserve the (relative) depth of the nodes, they allow only simple forms of reasoning. Other kinds of reasoning would be useful, especially reasonings mixing knowledge from different levels. Such reasonings can be

defined by first defining transformations of graphs, then by using NG homomorphisms. The decision homomorphism problem for NBGs, NBG-HOMOMORPHISM, is NP-complete (with a BG being a particular case of NBG), and an algorithm for the NBG-HOMOMORPHISM polynomial in the complexity of an algorithm for BG-HOMOMORPHISM can be simply constructed since computing a homomorphism from a tree to a tree (or more generally to any graph) is a polynomial problem (cf. Sect. 7.2.1).

9.3 Nested Graphs (NGs)

In order to express that different nodes appearing in an NBG represent the same entity, one can add coreferences to NBGs, as we did for for BGs (cf. Definition 3.10 in Chap. 3).

The following piece of information about a scientific document that is "an article, whose subject is a wheat food product that is cooked in water, has a result, whose nutritional observation is that the vitamin content of this wheat food product decreases, whose biochemical explanation is that this wheat food product contains hydrosoluble vitamin that is dissolved, and whose nutritional evaluation is that the nutritional quality of this wheat product is deteriorated" can be represented by the graph in Fig. 9.11. In this figure, the coreferent concept nodes are represented by the named variable $*x$.

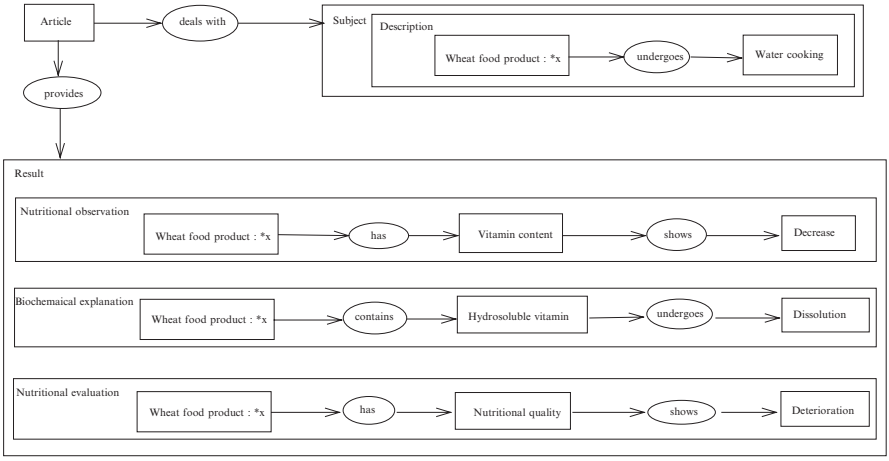

Fig. 9.11 A nested graph with coreferences represented by variables

An abstract example of a nested graph is represented in Fig. 9.12, where the nodes j, m, p, w are coreferent along with the nodes q, r. In this example, the coreference relation is represented by coreference links.

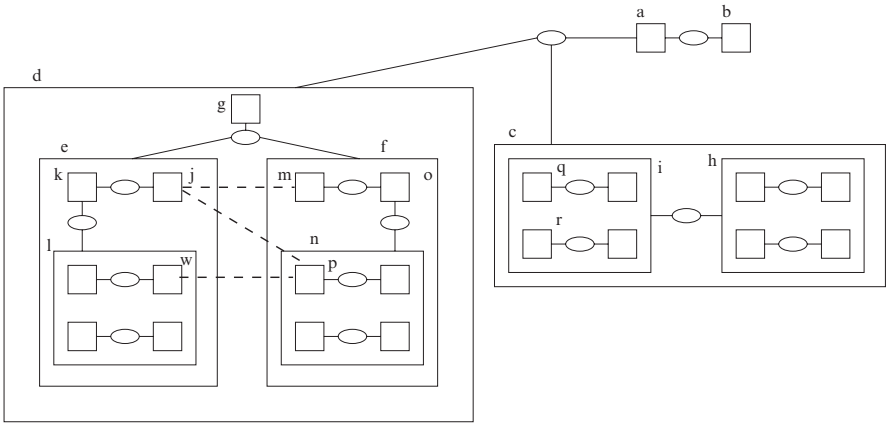

Fig. 9.12 An abstract nested graph with coreference links

Two coreferent concept nodes represent the same entity in nested graphs as in SGs. In SGs we have seen how it is possible to shrink each coreference class into a single node, thus obtaining a graph in normal form. The situation is more complicated for nested graphs. Knowledge is contextualized (by a hierarchical structure) in a nested graph, and merging two coreferent concept nodes can be done only if the contextualization is preserved. Two cases can be considered.

First, let us consider two coreferent concept nodes, say a and b, which are in the same box. These nodes may have descriptions. But, as a and b both represent the same entity (they are coreferent) in the same context (they are in the same box), one can merge a and b into a single node without altering the meaning of the graph. The description of the new node is the disjoint sum of the description of a and of the description of b.

Secondly, let us consider two coreferent concept nodes, say c and d, which are *not* in the same box, thus they are a priori *not* in the same context. The information about an entity in a context may be irrelevant in another context and merging two coreferent concept nodes which are in distinct contexts could entail inconsistencies. For instance, in Fig. 9.13, the concept nodes c and d are coreferent. They represent the same boy Paul but in two different contexts. Merging the two descriptions would state that Paul is dressed up as Zorro while this is stated only on the drawing. Nevertheless, if two coreferent nodes c and d are in the description of a and b, respectively, and a and b are coreferent nodes belonging to the same box, then after merging a and b, c and d become coreferent in the same box. In this case, they can be safely merged.

The simplest way of differentiating coreference links within a box and coreference links between two boxes is to stress that there is no coreference link within a box, i.e., each box is a normal SG. Hence the following definition:

Definition 9.6 (Nested Conceptual Graph (NG)). A *nested conceptual graph (NG)* $(G, coref)$ is an NBG G enriched by an equivalence relation $coref$ on X_G

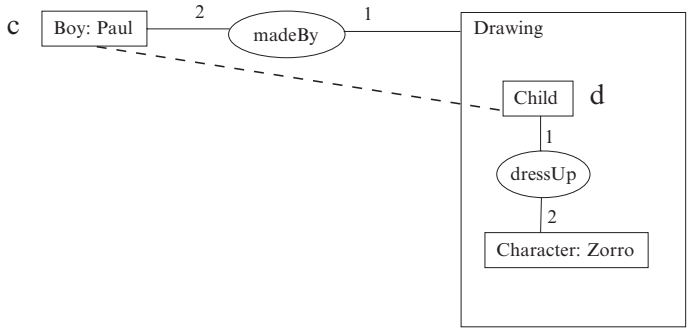

Fig. 9.13 An example of coreferent concept nodes which should not be merged

(the set of all concept nodes in all boxes of G) such that, for each box K in G, $coref$ restricted to K is the trivial equivalence, i.e., if $(c,c') \in coref$ and $c \neq c'$ then c and c' are in two different boxes.

Note that any box of an NG is a normal BG. If the box normality condition is not satisfied, i.e., if one considers a pair $(G, coref)$ where G is a NBG and $coref$ an equivalence relation which does not satisfy the condition in Definition 9.6, then G can be transformed into an intuitively semantically equivalent NG when the following normalization process is successful. One performs from the root of G a top-down (e.g., by a depth-first search or a breadth-first search) normalization of each box as follows. The normalization of a box consists of merging all coreferent nodes (in the box) into a single node, whose description is the disjoint sum of merged node descriptions. A sufficient condition for the normalization process is that each $coref$ class satisfies the condition of a coref class of an SG (cf. Definition 3.10), but this is not a necessary condition.

Definition 9.7 (NG homomorphism). A *(NG) homomorphism* from an NG G to an NG H is defined as a homomorphism of the underlying NBGs on condition that two coreferent nodes of G must have coreferent images in H.

9.4 Nested Typed Graphs

Whenever the universe of discourse can be naturally broken down into independent parts, some knowledge may possibly concern only some of these parts. In such a case, pieces of knowledge can be organized so as to respect the universe of discourse structure. In the same way, when something can be described using different viewpoints, each piece of information corresponding to a given viewpoint can be typed by this viewpoint. Let us give an example about text annotations. Concerning the content of a text, one can make a distinction between the text topic and the way this topic is presented, i.e., the text rhetoric. One can also consider the structure

of the text, the word distribution, or even bibliographical data such as the author's name, the publisher or the number of pages. For each of these aspects, a graph can be constructed, with each graph being labeled by the type of annotation, e.g., topic, rhetoric, structure, bibliographic data, and so on. Thus, one introduces graph types and typed graphs, and also sets of typed graphs.

Graph types are especially useful in nested graphs when one wants to consider different sorts of nesting. A nested typed graph can be considered as an NG in which a complex concept node can have several typed descriptions. In Chap. 13 nested typed graphs are used for representing document annotations. Let us again consider the previous text annotation example. An annotation of the text identified by $T121$ and represented by a concept node c labeled (Text, T121) can be composed of an annotation concerning the text topic and another annotation concerning the text rhetoric. Then, the (global) annotation of the text is a description of c composed of a graph A of type *topic* and a graph B of type *rhetoric*, i.e., the label of c is the triple (Text, T121, $\{$ (`Topic`, A), (`Rhetoric`, B)$\}$).

We define typed SGs (TGs) before considering nested typed graphs. A TG vocabulary $\mathcal{V}$ is an SG vocabulary supplemented by an ordered set of graph types T_G, i.e., $\mathcal{V} = (T_C, T_R, T_G, \mathcal{I})$.

Figure 9.14 presents a tree of graph types.

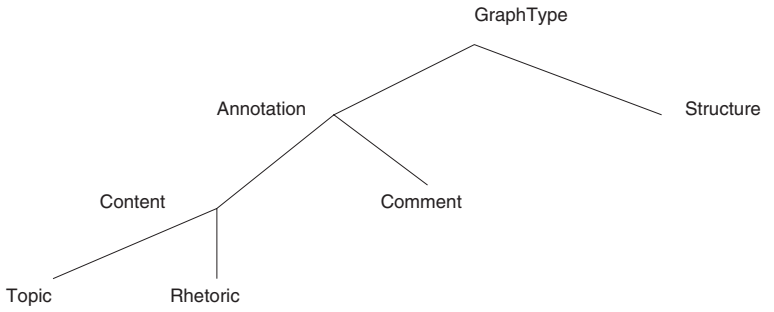

Fig. 9.14 A set of graph types

Definition 9.8 (TG and TG homomorphism). A *typed BG (resp.SG)* on a TG vocabulary $(T_C, T_R, T_G, \mathcal{I})$ is an ordered pair (g, G), where $g \in T_G$ and G is a BG (resp. SG) on $(T_C, T_R, \mathcal{I})$. A homomorphism π from G to H is a *(TG) homomorphism* from (g, G) to (h, H) if $h \leq g$.

Briefly said, a nested typed graph is a nested graph in which the boxes are no longer normal BGs but are typed normal BGs. A BG can be considered as a particular typed BG (there is only one graph type) and, in the same way, an NG can be considered as a particular nested typed graph defined as follows.

Definition 9.9 (Nested Typed Graph (NTG)).

- An *elementary* NTG G is obtained from a typed normal BG (g,H) by adding a third field to the label of each concept node c in H equal to $**$. The set of boxes of G is $boxes(G) = \{(g,H)\}$ and the complete concept node set of G is $X_G = C_H$. A trivial bijection exists between elementary NTGs and typed normal BGs (when no ambiguity occurs we do not distinguish between them).
- Let H be an NTG and $\{(g_1,H_1),\ldots,(g_k,H_k)\}$ a set of elementary NTGs, such that for any $i \neq j \in \{1,\ldots,k\}$, $g_i \neq g_j$. The graph G obtained from H by substituting $\{(g_1,H_1),\ldots,(g_k,H_k)\}$ for the third field $**$ of a concept node c in H is an NTG. $boxes(G) = boxes(H) \cup \{(g_1,H_1),\ldots,(g_k,H_k)\})$, and $X_G = X_H \cup X_{H_1} \cup \ldots \cup X_{H_k}$.
- The third field of any concept node $c \in X_G$ is called the *description* of c and is denoted $Descr(c)$.
- The set X_G of all concept nodes appearing in G is provided with an equivalence relation $coref$, such that for any box K the restriction of $coref$ to K is the trivial equivalence (i.e., if $(c,c') \in coref$ and $c \neq c'$ then c and c' are in two different boxes).

The notions defined for nested graphs (NGs) can be extended to nested typed graphs (NTGs) as follows. First, one can define a tree of typed SGs by substituting TGs for BGs in the definition of a BG tree (cf. Definition 9.2). Secondly, the transformation $Tree$ for nested graphs (cf. Definition 9.3) can be extended to nested typed graphs. Thirdly, the homomorphism definition between NGs can be extended to NTGs.

Definition 9.10 (Tree of typed BGs). A *tree of TGs* is a labeled rooted tree $T = (V_T,U_T,l_T)$, such that:

- V_T, the node set of T, is in bijection with a set of pairwise disjoint typed normal BGs $\{G_1,\ldots,G_k\}$. For any $i = 1,\ldots,k$, the node associated with G_i is labeled by G_i (the set $\{G_1,\ldots,G_k\}$ can be identified with V_T).
- For any arc (G_i,G_j) in U_T, $l_T(G_i,G_j)$ is a concept node in G_i; such a labeled arc is also denoted (G_i,c,G_j).
- For all pairs of arcs (G_i,G_j) and (G_i,G_k) with same label c, the graphs G_j and G_k have a different type.

It is sometimes convenient to label an arc (G_i,c,G_j) not only by c but also by a pair $(c,type(G_j))$. In this case, the condition 3 of the previous definition becomes: All labels are distinct, i.e., if (c,g) and (d,h) are the labels of two distinct arcs, then $c \neq d$ or $g \neq h$ or both.

There are two differences between NTGs and NGs. In NTGs, the boxes are typed, and a concept node has a description composed of a set of graphs, instead of a single graph in NGs.

It is now possible to define a tree of typed BGs associated with an NTG.

Definition 9.11 (Tree(G)). The mapping $Tree$ assigns to any NTG G a tree of typed SGs, denoted $Tree(G)$, which is defined as follows.
If G is an elementary NTG, then $Tree(G)$ is restricted to a single node labeled G.

If G is the NTG obtained from an NTG H by substituting $\{G_1,\ldots,G_k\}$ for $**$, which is the *Descr* of the concept node c in H, then $Tree(G)$ is built from $Tree(H)$ by adding k nodes labeled G_i, $i=1,\ldots,k$ as successors of the node labeled K in $Tree(H)$ containing c. Each arc (K,G_i), for $i=1,\ldots,k$, is labeled by c.

It can immediately be checked that, as in NBGs, the mapping $Tree$ is a bijection from the NTGs on a vocabulary $\mathcal{V}$ to the trees of typed BGs on $\mathcal{V}$. NTG homomorphisms are defined using the tree viewpoint.

Definition 9.12 (NTG Homomorphism). Let G and H be two NTGs with $Tree(G) = (V_G,U_G,l_G)$ and $Tree(H) = (V_H,U_H,l_H)$. A *(NTG) homomorphism* from G to H is a pair $\pi = (\pi_0,\{\pi_{G_1},\ldots,\pi_{G_k}\})$, where $V_G = \{G_1,\ldots,G_k\}$, which satisfies:

- π_0 is an ordinary tree homomorphism from (V_G,U_G) to (V_H,U_H) which maps the root of $Tree(G)$ to the root of $Tree(H)$,
- $\forall K \in V_G$, π_K is a TG homomorphism from K to $\pi_0(K)$,
- $\forall (J,K) \in U_G$, if $l_G(J,K) = c$ then $l_H(\pi_0(J),\pi_0(K)) = \pi_J(c)$),
- two coreferent nodes of G must have coreferent images in H.

Example. The NTG in Fig. 9.15 is obtained from the NG in Fig. 9.11 by replacing the concept types *Description*, *Nutritional observation*, *Nutritional evaluation* and *Biochemical explanation* by graph types. Moreover the root is typed (here by *Annotation*).

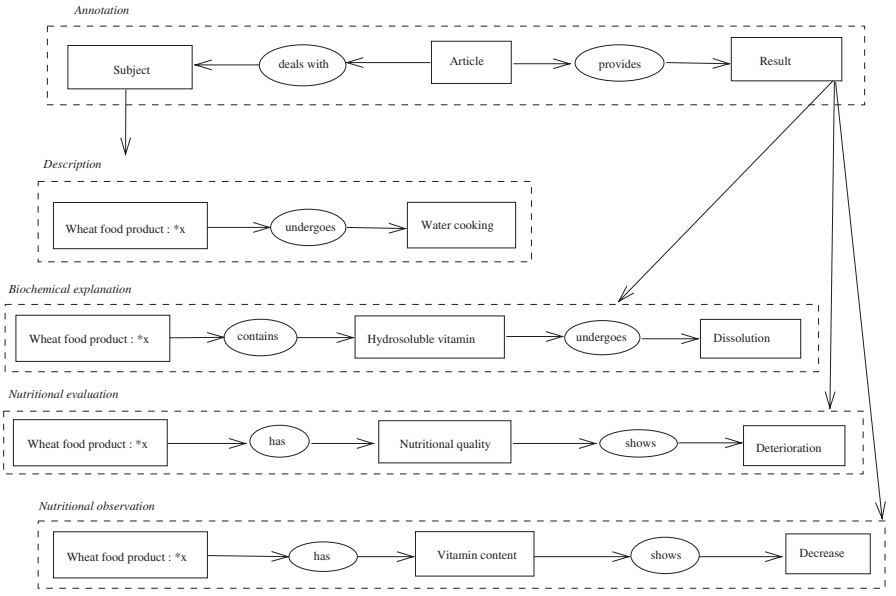

Fig. 9.15 An NTG

Example. In Fig. 9.16, it is assumed that: G is of type g, H of type h, $g \geq h$, and the graph types g_1 and g_2 are greater than or equal to g_3. The mapping π such that:

$\pi(c_1) = d_1$, $\pi(c_2) = d_2$, $\pi(c_3) = \pi(c_4) = d_3$, $\pi(r_1) = s_1$, $\pi(r_2) = s_2$, $\pi(r_3) = s_3$, is a homomorphism from (g,G) to (h,H).

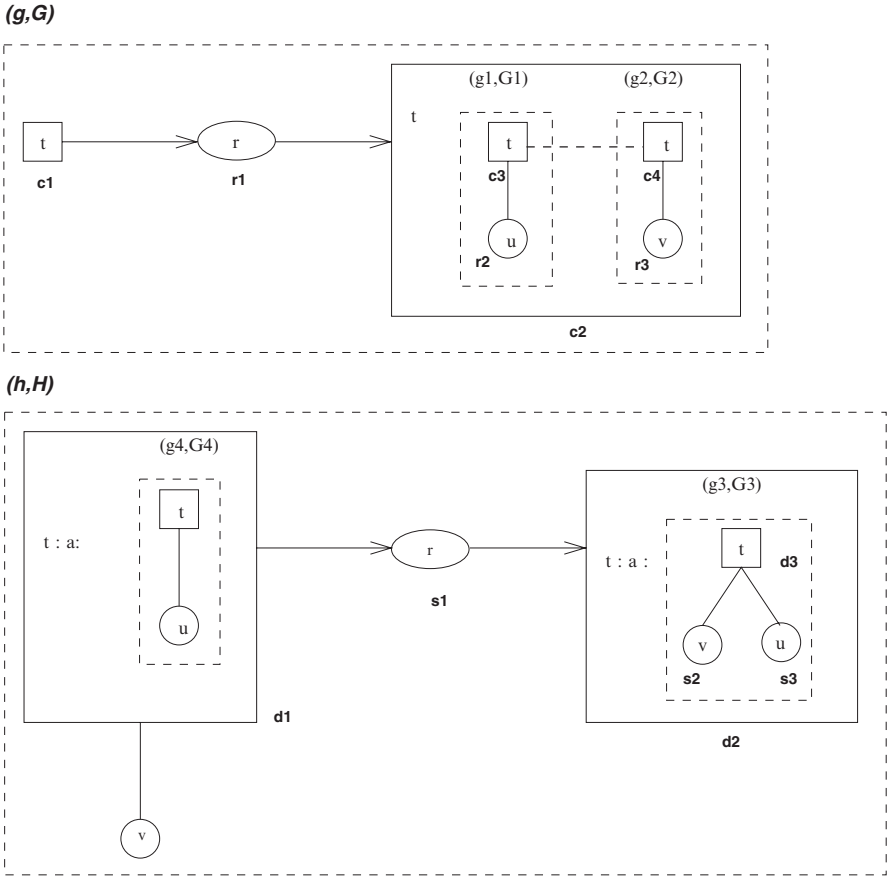

Fig. 9.16 An NTG homomorphism

9.5 The Semantics Φ_N

The FOL semantics Φ_N hereafter defined for NGs and NTGs is an extension of the FOL semantics Φ for SGs (cf. Sect. 4.2.1). The definition of Φ_N for NGs, i.e., untyped nested graphs, is based on two ideas. First, $\Phi_N(G)$ is the conjunction of a formula defining the tree structure of G with formulas associated with all boxes of G. Secondly, the formula associated with a box K is obtained from $\Phi(K)$ by adding an argument which is a variable representing K. The FOL semantics Φ_N for NTGs

is built from the FOL semantics for NGs by adding for each box K of type g an atom $g(y)$ (y being the variable representing K). Thus, we consider only NTGs in this section. If graph types are useless, i.e., if NGs are considered instead of NTGs, the soundness and completeness theorem still holds with Φ_N for NGs.

After defining Φ_N, we prove that the NTG homomorphism notion is sound and complete with respect to Φ_N.

9.5.1 Definition of Φ_N

Let $\mathcal{L}$ be the FOL language associated with the vocabulary $(T_C, T_R, \mathcal{I})$ (cf. Sect. 4.2.1). The FOL language $\mathcal{L}_N$ associated with the vocabulary $\mathcal{V} = (T_C, T_R, T_G, \mathcal{I})$ on which NTGs are built is obtained from $\mathcal{L}$ by the following transformations:

- a new constant a_0 is added to the constants assigned to the individual markers,
- each predicate p of arity $n \geq 1$ in $\mathcal{L}$ is transformed into a predicate, still denoted p, of arity $n+1$,
- a new ternary predicate *descr* is added to $\mathcal{L}$,
- for any $g \in T_G$ a new unary predicate, also denoted g, is added to $\mathcal{L}$ (useless for NGs).

Thus, a binary predicate is associated with each concept type in T_C, and an $(n+1)$-ary predicate is associated with each n-ary relation type. Note that the predicate *descr*, which is used for representing the tree structure of any NTG, belongs to any FOL language $\mathcal{L}_N$ associated with a vocabulary.

The following set of formulas, denoted $\Phi_N(\mathcal{V})$, is associated with $\mathcal{V} = (T_C, T_R, T_G, \mathcal{I})$:
$\forall z \forall x_1 \ldots x_n (t_1(x_1, \ldots, x_n, z) \rightarrow t_2(x_1, \ldots, x_n, z))$, for any t_1 and t_2 concept type in T_C (in this case $n = 1$) or relation type of arity $n \geq 1$ in T_R such that $t_1 \leq t_2$,
$\forall x (g_1(x) \rightarrow g_2(x))$, for all g_1, g_2 in T_G such that $g_1 \leq g_2$ (useless for NGs).

Let G be an NTG or an NG, and let $Tree(G)$ be its associated tree. A set of constants $a_1, \ldots, a_m$ is assigned to the coreference classes containing an individual concept node (the same letter is used to designate an individual marker and its associated constant), and a_0 is assigned to the root box of G. Two disjoint sets of variables are considered. First, a set of variables $x_1, \ldots, x_n$ are assigned to the generic coreference classes, and a set $y_1, \ldots, y_k$ are assigned to the k boxes of G distinct from the root box. Hereafter, we often refer to a box by its associated term, and to a coreference class by its associated term. In an NTG, a concept node c is identified by a pair (u, y), where u is the term (either a variable x_i or a constant a_i, $i \geq 1$) associated with the coreference class of c, and y is the term (either a variable y_i or a_0) assigned to the box containing c.

Example. For instance, in Fig. 9.17, the NG in Fig. 9.5 is represented with its associated variables. x_1 (resp. x_2, x_3) is the variable assigned to the generic concept node of type *Drawing* (resp. *Table*, *Train*), a_0 is the constant assigned to the root box and

y_1 is the variable assigned to the box description of the drawing. The fact that Paul is an individual concept node of type Boy in the root box a_0, is represented by the atom $Boy(Paul, a_0)$. The fact that the concept node x_3 has a *Green* attribute in the box y_1 is represented by the atom $attr(x_3, Green, y_1)$. The fact that in a_0 the concept node x_1 has y_1 for description is represented by the atom $descr(a_0, x_1, y_1)$. Finally, the fact that the type of J (resp. K) is *Proposition* (resp. *Subject*) is represented by $Proposition(a_0)$ (resp. $Subject(y_1)$).

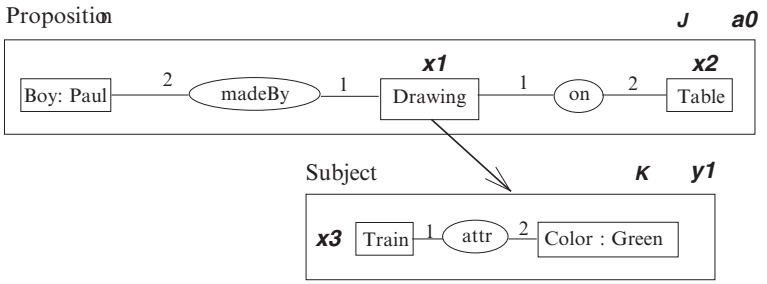

Fig. 9.17 Variables associated with concept nodes and boxes in an NTG

We require the following notation before giving the definition of Φ_N. Let us recall that, given an SG G, $\phi(G)$ is the conjunction of all atoms assigned to nodes of G without quantification, while $\Phi(G)$ is the existential closure of $\phi(G)$ (cf. Definition 4.7).

Definition 9.13 ($\phi(G,u)$). Let G be an SG and let u be a term. $\phi(G,u)$ is the formula obtained from $\phi(G)$ by adding the argument u to each atom in $\phi(G)$.

This transformation can be extended to any FOL formula. For instance, $\Phi(G,u)$ denotes the formula obtained from $\Phi(G)$ by adding the argument u to each atom in $\Phi(G)$.

Definition 9.14 ($\Phi_N(G)$). Let G be an NTG with typed boxes $(g_0,G_0),\ldots,(g_k,G_k)$, (g_0,G_0) being the root box. Let y_i, $i=1,\ldots,k$ be the variables assigned to the G_i-s and x_i, $i=1,\ldots,n$ be the variables assigned to the generic coreference classes. $\Phi_N(G) =$
$\exists y_1 \ldots y_k x_1 \ldots x_n (\phi(Tree(G)) \wedge g_0(a_0) \wedge \phi(G_0,a_0) \wedge \ldots \wedge g_k(y_k) \wedge \phi(G_k,y_k))$, where: $\phi(Tree(G))$ is the conjunction, for any (J,c,K) arc in $Tree(G)$, of the atoms $descr(v_j,u_i,y_k)$, where v_j is the term assigned to the box J, y_k is the variable assigned to the box K and u_i is the term assigned to the coreference class of c.

Example. The formula associated with the NTG G in Fig. 9.17 is:
$\Phi_N(G) = \exists y_1 x_1 x_2 x_3 (descr(a_0,x_1,y_1) \wedge proposition(a_0) \wedge subject(y_1) \wedge$
$Boy(Paul,a_0) \wedge Drawing(x_1,a_0) \wedge Table(x_2,a_0) \wedge madeBy(x_1,Paul,a_0) \wedge$
$on(x_1,x_2,a_0) \wedge Train(x_3,y_1) \wedge Color(Green,y_1) \wedge attr(x_3,Green,y_1))$
The formula associated with the NTG H in Fig. 9.18 is:
$\Phi_N(H) = \exists y_1 x_1 x_2 (descr(a_0,x_1,y_1) \wedge proposition(a_0) \wedge subject(y_1) \wedge Boy(x_2,a_0) \wedge$

$Drawing(x_1,a_0) \wedge madeBy(x_1,x_2,a_0) \wedge Child(x_2,y_1) \wedge Character(Zorro,y_1) \wedge dressUp(x_2,Zorro,y_1))$

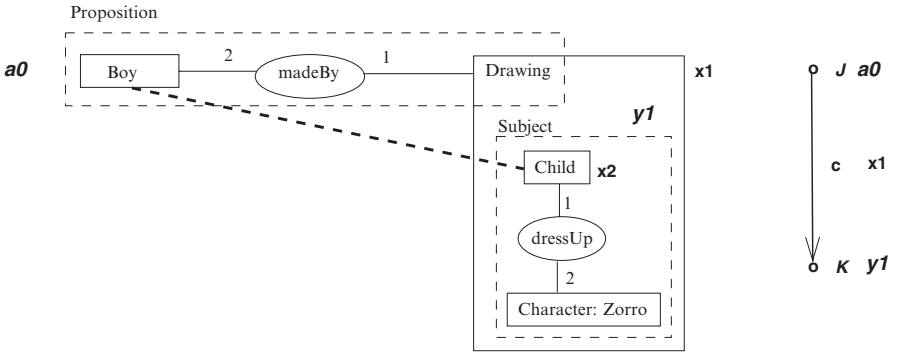

Fig. 9.18 A NTG with coreferent nodes

9.5.2 Soundness and Completeness

Let us show that the homomorphism notion between NGs is sound and complete with respect to the FOL semantics Φ_N. More precisely,

Theorem 9.1. *Let G and H be two N(T)Gs. There is a homomorphism from G to H if and only if* $\Phi_N(\mathcal{V}), \Phi_N(H) \models \Phi_N(G)$.

Note that there is no normality condition for completeness (contrary to the theorem for SGs) since we only consider normal boxes in the definition of nested graphs.

Proof. The proof is based on relationships between NTG homomorphism and $\mathcal{L}$-substitution and on the $\mathcal{L}$-substitution lemma. Ignoring the graph types, one obtains a proof for NGs.

Soundness

Let G and H be two NTGs, $Tree(G) = (V_G, U_G, l_G)$, $V_G = \{G_1, \ldots, G_k\}$, and $Tree(H) = (V_H, U_H, l_H)$, $V_H = \{H_1, \ldots, H_l\}$. Let $\pi = (\pi_0, (\pi_{G_1}, \ldots, \pi_{G_k}))$ be a homomorphism from G to H. Then (cf. Definition 9.12),

1. π_0 is a homomorphism from $Tree(G)$ to $Tree(H)$ which maps the root of $Tree(G)$ to the root of $Tree(H)$,
2. $\forall K \in V_G, \pi_K$ is a homomorphism from K to $\pi_0(K)$,
3. $\forall (J,K) \in U_G$ with $l_G(J,K) = (c,g)$, one has $l_H(\pi_0(J), \pi_0(K)) = (\pi_K(c), h)$, where h is the type of $\pi_0(K)$,

4. if c and c' are two coreferent concept nodes in G, then $\pi(c)$ and $\pi(c')$ are coreferent in H.

Let us prove that there is a $\mathcal{L}_N$-substitution from $\Phi_N(G)$ to $\Phi_N(H)$.
$\Phi_N(G) = \exists y_1 \ldots y_k x_1 \ldots x_m(\phi(Tree(G)) \wedge g_0(a_0) \wedge \phi(G_0, a_0) \wedge \ldots \wedge g_k(y_k) \wedge \phi(G_k, y_k))$, where:
$\phi(Tree(G)) = \bigwedge\{descr(y'_j, u_i, y_p) \mid (y'_j, u_i, y_p) \in Tree(G)\}$, where y'_j is either a_0 or a variable y_j.
$\Phi_N(H) = \exists z_1 \ldots z_l w_1 \ldots w_n(\phi_N(Tree(H)) \wedge h_0(a_0) \wedge \phi(H_0, a_0) \wedge \ldots \wedge h_l(z_l) \wedge \phi(H_l, z_l))$, where:
$\phi(Tree(H)) = \bigwedge\{descr(z'_j, v_i, z_p) \mid (z'_j, v_i, z_p) \in Tree(H)\}$, where z'_j is either a_0 or a variable z_j.

Let σ be the substitution defined as follows:
for any y_i, $i = 1, \ldots, k$, $\sigma(y_i) = z_{f(i)}$, where $z_{f(i)}$ is the variable assigned to the box $\pi_0(G_i)$ of H;
for any x_i, $i = 1, \ldots, m$, $\sigma(x_i) = w_{g(i)}$, where $w_{g(i)}$ is the term assigned to the coreference class of $\pi(c)$, where c is any concept node in the coreference class associated with x_i (the images of coreferent nodes are coreferent nodes, thus $w_{g(i)}$ is well defined).

Let us check that σ is a $\mathcal{L}_N$-substitution from $\Phi_N(G)$ to $\Phi_N(H)$.

- Let $descr(y'_j, u_i, y_p)$ be an atom of $\Phi_N(G)$, then $(J, K) \in U_G$, K (associated with y_p) is the description of a concept node in J (associated with y'_j), which is in the coreference class associated with u_i. Since π is a homomorphism from G to H, $(\pi_0(J), \pi_J(c), \pi_0(K))$ is in $Tree(H)$, and thus $descr(\sigma(y'_j), \sigma(u_i), \sigma(y_p))$ is an atom of $\Phi_N(H)$.
- The atom $h_0(a_0)$ of $\Phi_N(H)$ corresponds to the atom $g_0(a_0)$ of $\Phi_N(G)$.
- Let $g_i(y_i)$ be an atom of $\Phi_N(G)$, g_i the type of G_i and $\pi_0(G_i) = H_{\pi_0(i)}$. Let h be the type of $H_{\pi_0(i)}$ then $h \leq g_i$ and $h(z_{f(i)}) = h(\sigma(y_i))$ is an atom in $\Phi_N(H)$.
- Let $t(u, a_0)$ be an atom of $\Phi_N(G_0)$. It corresponds to a concept node c in G_0 which is in the coreference class associated with u. The atom $t'(u', a_0)$ is assigned to $\pi(c)$, where u' is equal to the term associated with the image by π of the coreference class u. Thus, $t'(u', a_0) = t'(\sigma(u), a_0)$, and $t' \leq t$.
- Let $t(u, y_i)$ be an atom of $\Phi_N(G_i)$. It corresponds to a concept node c in G_i, which is in the coreference class associated with u. The atom $t'(u', z_{f(i)})$ is assigned to $\pi(c)$, where u' is equal to the term associated with the image by π of the coreference class u. Thus, $t'(u', z_{f(i)}) = t'(\sigma(u), \sigma(y_i))$, and $t' \leq t$.
- Let $p(\vec{e}, y_i)$ be an atom of $\Phi_N(G_i)$ corresponding to a relation node r of G_i. The atom $p'(\vec{e}', z_{f(i)})$ is assigned to $\pi(r)$, where $p' \leq p$ and $\vec{e}'$ is the term vector associated with the nodes which are the images by π of the nodes whose term vector is $\vec{e}$, i.e. $\vec{e}' = \sigma(\vec{e})$. Thus, $p'(\pi(\vec{e}), z_{f(i)}) = p'(\sigma(\vec{e}), \sigma(y_i))$ is in $\Phi_N(H)$ and $p' \leq p$. A similar proof can be done for an atom $p(\vec{e}, a_0)$ of $\Phi_N(G_0)$ corresponding to a relation node in G_0. One concludes with the $\mathcal{L}$-substitution lemma.

□

Completeness

Let G and H be two NTGs. $Tree(G) = (V_G, U_G, l_G)$, $V_G = \{G_0, G_1, \ldots, G_k\}$, and $Tree(H) = (V_H, U_H, l_H)$, $V_H = \{H_0, H_1, \ldots, H_l\}$. Let us assume that $\Phi_N(\mathcal{V}), \Phi_N(H) \models \Phi_N(G)$, and, using the $\mathcal{L}$-substitution lemma, let σ be a $\mathcal{L}_N$-substitution from $\Phi_N(G)$ to $\Phi_N(H)$. From σ, we first build a tree homomorphism π_0 from $Tree(G)$ to $Tree(H)$ that maps the root of $Tree(G)$ to the root of $Tree(H)$; then we build a homomorphism π_i from G_i to $\pi_0(G_i)$ for any $i = 0, \ldots, k$.

One takes $\pi_0(G_0) = H_0$. For any $i = 1, \ldots, k$, $\pi_0(y_i)$ is defined by $\sigma(y_i)$. More precisely if $\sigma(y_i) = z_{f(i)}$ then $\pi_0(G_i) = H_{f(i)}$. The substitution σ associates to $g_i(y_i)$ an atom $h(\sigma(y_i))$ with $h \leq g_i$. Thus the type of $\pi_0(G_i)$ is $\leq$ type of G_i.

Let us now check that π_0 is a tree homomorphism from $Tree(G)$ to $Tree(H)$. If $k = 0$, then $Tree(G)$ is restricted to a single node and π_0 is a tree homomorphism. Let us assume that $k \geq 1$. Let us consider an arc (J,K) in $Tree(G)$ with $l_G(J,K) = c$. An atom $descr(y'_j, u_i, y_p)$ in $\Phi_N(G)$ corresponds to this arc and $descr(\sigma(y'_j), \sigma(u_i), \sigma(y_p))$ is an atom of $\Phi_N(Tree(H))$. Thus, $(\pi_0(J), \pi_0(K))$ is an arc of $Tree(H)$ and $l_H(\pi_0(J), \pi_0(K)) = (d, type(\pi_0(K))$ where d is the (single) concept node in $\pi_0(J)$ associated with the term $\sigma(u_i)$ (all boxes are normal). This proves that π_0 is a homomorphism from $Tree(G)$ to $Tree(H)$ (and it maps the root of $Tree(G)$ to the root of $Tree(H)$).

Let us show that σ is an $\mathcal{L}$-substitution from $\Phi(G_i)$ to $\Phi(\pi_0(G_i))$, for $i = 0, \ldots, k$. Let $p(\vec{e})$ be an atom of $\Phi(G_i)$; then there is an atom $q(\sigma(\vec{e}))$ in $\Phi(\pi_0(G_i))$ such that $q \leq p$. Indeed, for any $i = 0, \ldots, k$ the atoms of $\Phi_N(G_i)$ are mapped by σ to atoms of $\Phi_N(\pi_0(G_i))$ with a decrease of the predicate. Thus, the result holds for $i = 0$. For any $i \in \{1, \ldots, k\}$, $p(\vec{e}, y_i)$ is an atom of $\Phi(G_i, y_i)$; therefore it is an atom of $\Phi_N(G)$ and there is an atom $q(\sigma(\vec{e}), \sigma(y_i))$ of $\Phi_N(H)$ with $q \leq p$. $q(\sigma(\vec{e}), \sigma(y_i))$ is an atom of $\Phi(\pi_0(G_i), \sigma(y_i))$. Thus $q(\sigma(\vec{e}))$ is an atom of $\Phi(\pi_0(G_i))$ with $q \leq p$. As σ is an $\mathcal{L}$-substitution from $\Phi(G_i)$ to $\Phi(\pi_0(G_i))$, for $i = 0, \ldots, k$, then with Property 4.7 there is a homomorphism π_i from G_i to $\pi_0(G_i)$ such that for any concept c in G_i associated with the term u in $\Phi(G_i)$, the concept $\pi_i(c)$ is associated with $\sigma(u)$ in $\Phi(\pi_0(G_i))$. The last condition of an NTG homomorphism (cf. Definition 9.12) is satisfied since if $l_G(J,K) = c$ (associated with u), then $l_H(\pi_0(J), \pi_0(K)) = d$ where d is the single concept in $\pi_0(J)$ associated with $\sigma(u)$ and $d = \pi_J(c)$. □

9.6 Representation of Nested Typed Graphs by BGs

In this section, we define an injective mapping *ng2bg* which assigns a BG to a nested graph. We will see that this mapping preserves the homomorphisms.

A BG (or SG) vocabulary $\mathcal{V}' = (T_C, T_R \cup T_G \cup \{cont, descr\}, \mathcal{I} \cup \{a_0\})$ is assigned to a typed graph vocabulary $\mathcal{V} = (T_C, T_R, T_G, \mathcal{I})$, where the elements in T_G are now considered as unary relation types, and *cont*, *descr* and *coref* are three new binary relations and a_0 is a new individual marker.

Let $(G, coref_G)$ be an NTG on $\mathcal{V}$, with $Tree(G) = (V_G, U_G, l_G)$ and $V_G = \{G_1, \ldots, G_k\}$, G_1 being the root. First, for $i = 1, \ldots, k$, each G_i (on the vocabulary $\mathcal{V}$) is transformed into an BG G'_i (on the vocabulary $\mathcal{V}'$) as follows. A new concept node c_i is added to G_i and, if $i = 1$, i.e., G_i is the root of G, then its label is $(\top, a_0)$. Otherwise its label is $(\top, *)$; c_i is linked to a new unary relation node labeled by g_i (the type of G_i). For each concept node c in G_i, a new relation node labeled *cont* is created with c_i as its first neighbor and c as its second neighbor.

Secondly, the tree structure has to be coded by the relation *descr* used as follows. Whenever G_j is the description of c in G_i, i.e., (G_i, c, G_j) is an arc of $Tree(G)$, then a new relation node labeled *descr* is added, which links c in G'_i to c_j in G'_j.

Thirdly, $coref_G$ is translated by relation nodes of type $coref$. For each pair $(c_i, c_j) \in coref_G$, we add $coref(c_i, c_j)$ (as $coref_G$ is an equivalence relation, we also have $coref(c_j, c_i)$, $coref(c_i, c_i)$ and $coref(c_j, c_j)$.

Example. For instance, let G be the NTG in Fig. 9.18, then $ng2bg(G)$ is represented in Fig. 9.19.

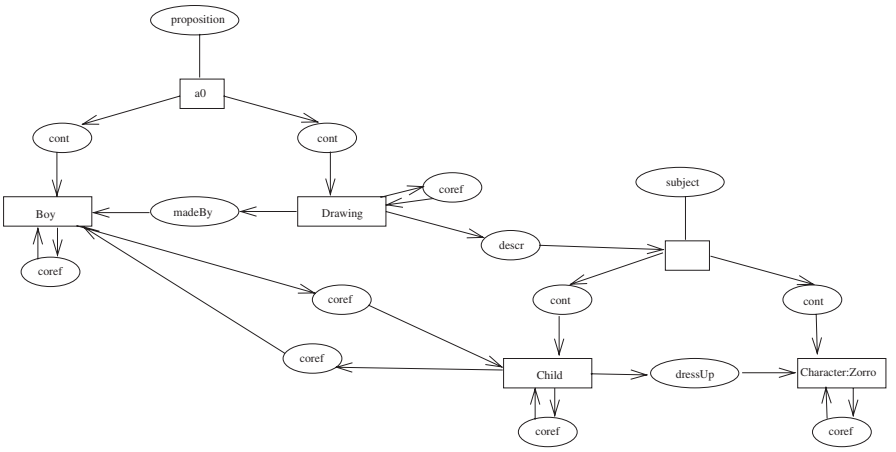

Fig. 9.19 The SG $ng2bg(G)$ associated with the nested typed graph G in Fig. 9.18

Theorem 9.2. ng2bg *is an injective mapping from the set of NTGs on $\mathcal{V}$ to the set of BGs on $\mathcal{V}'$. Furthermore, there is a bijection between the set of (NTG) homomorphisms from G to H and the set of (BG) homomorphisms from $ng2bg(G)$ to $ng2bg(H)$.*

Proof. Let G be an NTG on $\mathcal{V}$, with $Tree(G) = (V_G, U_G, l_G)$ and $V_G = \{G_1, \ldots, G_k\}$, G_1 being the root and $G' = ng2bg(G)$. For $i = 1, \ldots, k$, c_i is the node in G' associated with G_i and G'_i is the subBG of G' corresponding to G_i. Let H be an NTG on $\mathcal{V}$, with $Tree(H) = (V_H, U_H, l_H)$ and $V_H = \{H_1, \ldots, H_l\}$, with H_1 being the root and $H' = ng2bg(H)$. For $i = 1, \ldots, l$, d_i is the node in H' associated with H_i and H'_i is the subBG of H' corresponding to H_i. Let $\pi = (\pi_O, \{\pi_1, \ldots, \pi_k\})$ be a homomorphism

from G to H where π_i is a homomorphism from G_i to $H_{f(i)}$. A homomorphism π' from G' to H' is built as follows (cf. Fig. 9.20).

- For any $i = 1,\ldots,k$ and any node x in G'_i, $\pi'(x) = \pi_i(x)$.
- For any c_i, $i = 1,\ldots,k$, $\pi'(c_i) = d_{f(i)}$.
- For any unary relation node r (of type $l_G(G_i)$) incident to c_i, $\pi'(r)$ is the unary relation node (of type $l_H(H_{f(i)}) \leq l_G(G_i)$) incident to $\pi'(c_i) = d_{f(i)}$.
- Let r be a binary relation node of type *cont* in G' and (c_i, c) its neighbor list. $\pi'(r)$ is the relation node of type *cont* in $H_{f(i)}$ having for neighbor list $(\pi'(c_i), \pi'(c))$.
- Let r be a binary relation node of type *descr* in G' and (c, c_j) its neighbor list. Let us assume that c is in G'_i. $\pi'(r)$ is the relation node of type *descr* in H' having for neighbor list $(\pi'(c), d_{f(j)})$.
- Let r be a binary relation node of type $coref_G$ in G' and (u_1, u_2) its neighbor list. Let us assume that u_1 and u_2 are in G'_i. $\pi'(u_1)$ and $\pi'(u_2)$ are in $H'_{f(i)}$, and there is a binary relation node r' of type $coref_H$ in $H'_{f(i)}$. One takes $\pi'(r) = r'$.

One can check that π' is a homomorphism from G' to H', and this mapping from the (NTG) homomorphisms from G to H to the (BG) homomorphisms from G' to H' is injective and surjective. □

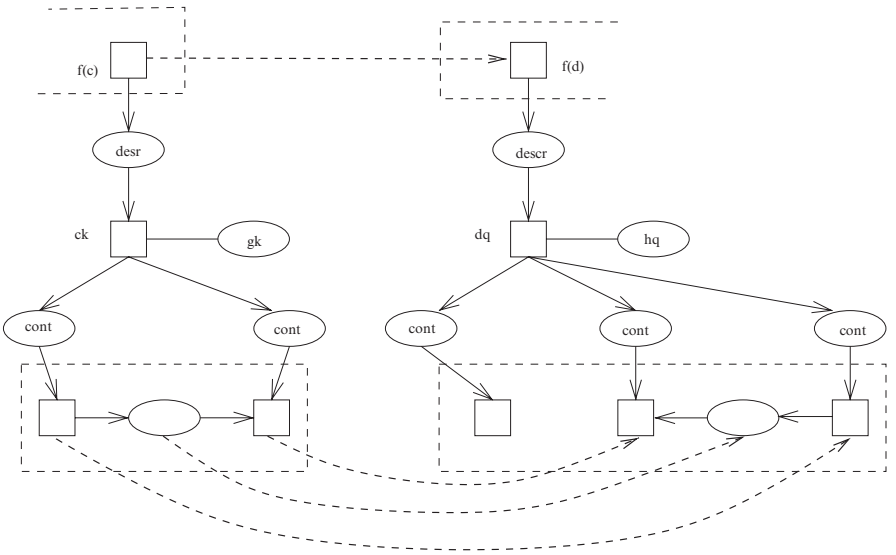

Fig. 9.20 *ng2bg* preserves the homomorphisms (proof of Theorem 9.2)

The previous correspondence allows us to transport notions and properties of SGs to NTGs and, in a way, the nested typed graphs (NTGs) are not more expressive than simple graphs (SGs). However, even if the transformation *ng2bg* is simple, the SG obtained seems more difficult to understand for a human being than the initial NTG.

9.7 Bibliographic Notes

It is generally agreed that knowledge has a contextual component, and different formalizations have been proposed to deal with knowledge contexts (e.g., cf. [BP83], [Guh91] and [McC93]).

History of nested conceptual graphs. Nested conceptual graphs were introduced in [Sow84]. Concept nodes representing a proposition can have a referent that is a graph; the intuitive semantics is that this graph is asserted by, or describes, the surrounding proposition. After that, several variants of nested conceptual graphs have been proposed to represent contexts, knowledge description by increasing level of detail, objects in object oriented programming, or related notions. Natural language processing is a main application domain (e.g., [Sow92] [Naz93] [Dic94]).

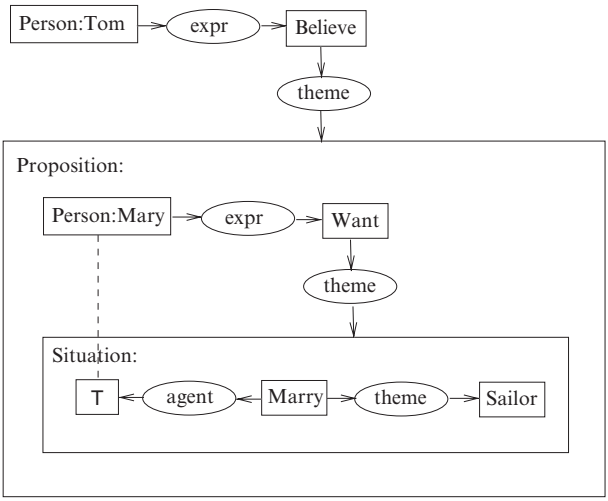

Fig. 9.21 Nested Graph representing a natural language sentence

For instance, the Fig. 9.21 (due to Sowa, e.g., [Sow99])) represents the sentence "Tom believes that Mary wants to marry a sailor." Tom is the experiencer (expr) of the concept [Believe], which is linked by the relation (theme) to a proposition that Tom believes. This proposition is described by another graph, which states that Mary is the experiencer of [Want], whose theme is a situation that Mary hopes will occur. That situation is described by another graph, which says that Mary (represented by the concept [$\top$]) marries a sailor. The coreference link shows that the concept [$\top$] in the situation concept refers to the same individual as the concept [Person: Mary] in the proposition concept.

We introduced a combinatorial structure in [CM97], provided with a homomorphism called "Graph of graphs," and showed that the different kinds of nested conceptual graphs found in the CG literature are instantiations of this generic notion,

and more precisely are trees of specific graphs. Relationships with category theory are also pointed out in this paper. Indeed, a kind of graphs provided with a morphism is a concrete category and transformations like *ng2bg* are functors between two categories. Note however that this framework did not take coreference links into account. It was only concerned with the recursive structure of nested graphs.

The model of nested graphs (without coreference) presented in this chapter was introduced in [MC96] and developed in [CM97]. The latter paper also introduces typed nestings, to specify relationships between the surrounding node and one of its descriptions. Note that, in the model presented in this chapter, we shifted nesting types to graph types. This allows us to type graphs at the first level, and not only as descriptions of a concept node. This model was motivated by two applications in knowledge representation, one concerning simulation (cf. [BBV97]) and the other concerning document indexing (cf. [Gen00]) Since then it has been used in other works (e.g., [GC05], [MLCG07] and [TML06] from which the example of Fig. 9.11 and Fig. 9.15 are extracted). In [CMS98] and [PMC98]) nested graphs with coreference links are processed and two sound and complete logical semantics are defined (see below).

Different logical semantics. Sowa extends the mapping Φ (as defined on SGs) by introducing a special binary predicate *descr* whose first argument is a term and second argument is a formula. For each complex concept node c with referent a non-empty graph D, $\Phi(G)$ contains the atom $descr(e, \Phi(D))$, where e is the term assigned to c. This logical semantics is similar to the semantics of contexts in [Guh91] [McC93], with *descr* playing the same role as the *ist* predicate. Note that the predicate *descr* takes a formula as argument, thus this logical semantics goes beyond FOL.

A non-classical logical semantics, with a calculus based on Gentzen-like sequents, equivalent to NG homomorphism, was presented in [PMC98]. In [Sim98] [CMS98], two alternative semantics in classical FOL are given to NGs. The first one extends Φ (as defined on SGs). For homomorphism completeness, the target NG has to be in so-called *k-normal* form, where k is the depth of the source NG. Every graph can be put in *k-normal* form. The drawback is that putting a graph into *k-normal* form may involve increasing the depth of its tree structure. The second semantics extends Ψ (as defined on SGs, cf. Chap. 4). As for SGs, there is no normality condition on the target graph. None of these two semantics are entirely satisfying because they do not translate the notion of context. With Φ, a concept node is simply represented by the term associated with its coreference class without a term representing its context. Thus properties attached to a concept node can be equivalently attached to any coreferent concept node, even if this node is in another context. With Ψ, the opposite approach is taken: Two coreferent nodes are translated by distinct terms (plus a term translating the fact that they represent the same entity); consequently, properties attached to one node cannot be equivalently attached to the second node, even if these nodes are in the same context. The logical semantics Φ_N defined in this chapter distinguishes between two kinds of coreference, depending on whether it is intra-context or inter-contexts, and thus can be seen as a combination of previous semantics Φ and Ψ. The logical semantics for typed nested graphs is new.

Other works on nested graphs. In [Bag99], a generalization of nested graphs (boxed graphs) is defined in which each box contains a BG, and relation nodes can link concept nodes belonging to different boxes. This extension was mainly done to reify coreference links. The idea is to replace the coreference relation (or to supplement it) by different relation types translating different coreference relations. Each type is provided with rules which describe the properties of this coreference relation. More generally, a boxed graph can be seen as a nested graph with several roots, added with a set of relation nodes linking concept nodes of different boxes (these relation nodes are said to be out of context). The boxed graph homomorphism does not necessarily map roots to roots, and a relation node out of context can be mapped to a node in or out of context, provided that edges are preserved. A transformation from boxed graphs to simple graphs is given, which, as a side effect, gives another proof of equivalence between simple and nested graphs (see also [Bag01] (in French)).

In [Ker01], descriptions are named. However, this framework does not provide operations on nested graphs. A formal declarative semantics is given by an embedding into a nested structure, with an associated notion of truth, from which a notion of deduction on nested graphs is defined. A transformation from these nested graphs to simple graphs is exhibited, which preserves deduction.

In [Pre98a] and [Pre00], the NGs studied are without generic nodes. A semantics (called contextual semantics) in connection with situation theory [BP83] and formal concept analysis [GW99] is proposed. Triadic power context families, i.e., a kind of formal context in the sense of formal concept analysis, are defined. The notion of a standard model built on a triadic power context family and the related notion of semantic entailment are introduced. An NG G_2 is semantically entailed by an NG G_1 if and only if G_2 is valid in the standard model of G_1. Finally, the framework is provided with eight sound and complete elementary rules which are sound and complete with respect to semantic entailment.