Chapter 3
Simple Conceptual Graphs

Overview

Concept nodes representing the same entity are said to be *coreferent* nodes. In basic conceptual graphs (BGs) only individual concept nodes may be coreferent. *Simple Conceptual Graphs (SGs)* enrich BGs with unrestricted coreference. The introduction of this chapter develops the discussion concerning equality started in the previous chapter and presents the *conjunctive type* and *coreference* notions. A new definition of a vocabulary extending the previous one with conjunctive types is given in Sect. 3.2. Section 3.3 defines SGs. An SG is simply a BG plus a coreference relation. The coreference relation is an equivalence relation over the concept node set with the following meaning: All concept nodes in a given equivalence class represent the same entity.

SG homomorphisms naturally extend BG homomorphisms. In Sect. 3.4, generalization and specialization operations defined on BGs are extended to SGs. Normal SGs are introduced in Sect. 3.5. An SG is *normal* if its coreference relation is the identity relation, i.e., each node is solely coreferent with itself. A normal SG can be associated with any SG. In fact, normal SGs and normal BGs can be identified, which emphasizes the importance of normal BGs. The notion of *coref-homomorphism*, which is specific to SGs, is introduced in Sect. 3.6. Instead of mapping concept nodes onto concept nodes as for a homomorphism, a coref-homomorphism maps *coreference classes* onto coreference classes. Relationships between homomorphisms and coref-homomorphisms are studied, and it is shown that, in the presence of a coreference, the intuitive meaning of generalization or specialization operations is better captured by coref-homomorphism than by homomorphism. The normal form of an SG is in a sense the most compact form of an SG. The *antinormal* form studied in the last section can be considered as the most scattered form of an SG: Each relation node is separated from any other relation node, and there are no multiple edges.

3.1 Introduction

The essential difference between *basic* conceptual graphs (BGs) and *simple* conceptual graphs (SGs) is that in SGs several concept nodes *not necessarily individual* can represent the same entity. Nodes representing the same entity are said to be *coreferent*, i.e., they refer to the same entity. Then the first question is: Can any pair of concepts be coreferent?

This question is closely related to the structure and the interpretation of the concept type set. Besides the structure of the ordered concept type set (e.g., a semi-lattice or lattice), an essential point is how the type hierarchy is interpreted. When a type is interpreted as a set of entities, the order over the hierarchy is interpreted as the inclusion. Then, the greatest lower bound (*glb*) of two types, when it exists, can be *lattice-theoretically* or *order-theoretically* interpreted. In the order-theoretic interpretation there is nothing more than the interpretation of the subtype relation by set inclusion. In the lattice interpretation, the interpretation of the glb of two types is then interpreted as the intersection of these sets. E.g., let *Building* and *OldThing* be two incomparable types, and let *HistoricLandmark* be their glb (see Fig. 3.1 left). With both interpretations, every entity of type *HistoricLandmark* is also of types *Building* and *OldThing*. With the lattice interpretation, every entity of both types *Building* and *OldThing* is necessarily of type *HistoricLandmark*. With the order-theoretic interpretation, the glb of two types is simply a subtype, thus a subset of their intersection; an entity of both types *Building* and *OldThing* is not necessarily a *HistoricLandmark* (cf. Fig. 3.1 right).

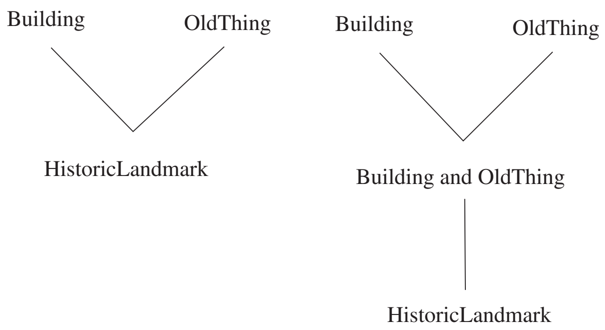

Fig. 3.1 Interpretation of types

The way the concept type set is interpreted has an effect on reasoning.

For instance, let Q be the query [HistoricLandmark:*] ("is there a historic landmark?") and let G be a fact containing the coreferent concept nodes [Building:a] and [OldThing:a], thus indicating that a is an entity which is a *Building* and an *OldThing*. With the lattice interpretation, G should provide an answer to Q while in the order-theoretic interpretation it should not.

The lattice-theoretic interpretation explicitly requires building all types of intersections (for instance, if an entity of types *Building* and *OldThing* is not necessarily of type *HistoricLandmark* one has to build their common subtype *Buildingand-OldThing*, which is a supertype of *HistoricLandmark*, see Fig. 3.1 right). Thus it leads to a number of new and artificial types which can be exponential in the number of useful types.

On the other hand, the drawback of the order-theoretic interpretation is that we lose the possibility of expressing conjunctions of types, e.g., that *HistoricLandmark* is exactly the intersection of *OldThing* and *Building*. This property could be expressed by a rule (cf. Chap. 10), stating that every entity of type *Building* and of type *OldThing* is also of type *HistoricLandmark*. However, coding the conjunction of types in the hierarchy itself is more efficient than using rules, exactly as coding the subtype relation in a type hierarchy is more efficient than considering a flat set of types and representing the hierarchy by rules.

Conjunctive Type

A concept node represents an entity, but an entity can be represented by several coreferent concept nodes. A way of gathering all or some concept nodes representing the same entity is desirable to ease the building and understanding of an SG by a human being.

A simple way to gather these nodes is to merge them into one node. This operation has to keep the meaning of the original SG. Thus if an entity is represented by a node of type t and by a node of type t' there must be a type representing exactly the entities which are of both types t and t'. More generally, we need to express that a type is the conjunction of several types. Any subset of (incomparable) primitive types defines a conjunctive type. The set of conjunctive types is partially ordered by an order extending the order defined on primitive types.

However, not all conjunctions of types have a meaning. Thus we need a way of expressing that the conjunction of two (or more) types is *banned*, or equivalently that they cannot have a common subtype. A classical way of doing this is to add a special *Absurd* type below disjoint types. But this technique is not always precise enough. For instance, t_1, t_2, t_3 being direct supertypes of *Absurd* means that the intersection of any two of them is empty. We cannot express that t_1 and t_2 are disjoint as well as t_1 and t_3, but that there can be an entity of type t_2 and t_3 (e.g. $t_1 = Animal$, $t_2 = Ship$, $t_3 = Robot$, with all being subtypes of *MobileEntity*).

Giving all acceptable (i.e., non-banned) types in extension is not conceivable in practice, so we define the set of acceptable conjunctive types by the primitive type set and the (maximal) banned conjunctive types. In the previous example, the types $\{t_1, t_2\}$ and $\{t_1, t_3\}$ would be banned. Theoretically, the number of banned conjunctive types can be exponential in the number of primitive types but, practically, considering banned types of cardinality two seems sufficient and their number is at most quadratic in the number of primitive types.

Coreference

An individual marker is generally considered as a surrogate or an identifier (in the programming meaning) of an entity. According to the Unique Name Assumption (UNA) common in knowledge representation, it is thus assumed that two distinct individual markers represent distinct entities. Hence nodes with distinct individual markers cannot belong to the same coreference set.

Let us consider two concepts with incomparable types t_1 and t_2. Can these nodes be coreferent without any condition on t_1 and t_2? The answer is positive in many works. As already said, in our opinion important properties required are, first, that *(a)* coreferent concepts can always be merged into a single node, secondly, that *(b)* the meaning of the obtained graph is the same as that of the original one. With a lattice-interpretation of concept types these properties are fulfilled if the types of the coreferent nodes have a glb (distinct from the *Absurd* type), which becomes the type of the new node. With the order-theoretic assumption, and in the absence of conjunctive types, the only way to fulfill these properties is to require that the set of types of the coreferent nodes possesses a minimal element in this set itself. Indeed, let us suppose for instance that there are two coreferent concepts with types *Building* and *OldThing*, respectively. Either these concepts cannot be merged and *(a)* is not satisfied, or they can and the type of the new node is a subtype of *Building* and *OldThing* and the obtained graph is (in general) strictly more specialized than the original one. That is why in works where properties *(a)* and *(b)* are required, it is required that coreferent nodes have the same type. In the framework of conjunctive types proposed here, *(a)* and *(b)* are obtained as soon as the conjunction of the types of the coreferent nodes is not a banned type. Then every SG can be easily transformed into an equivalent normal SG, which is called its *normal form*.

Whether a coreference set contains both generic and individual nodes is less important. Indeed, making a generic node and an individual node coreferent can always be performed by restricting the marker of the generic concept to the individual marker of the other node without changing the meaning of the graph.

3.2 Vocabulary

In a *conjunctive vocabulary*, the set T_C of concept types is built with three components: a set of primitive concept types, an operation of type conjunction and a set of banned conjunctive types. A set of primitive concept types has the same structure as a set of concept types in a vocabulary as defined in Chap. 2, i.e., it is a partially ordered set with a greatest element.

A conjunctive type is a set of n incomparable primitive types $E = \{t_1,\dots,t_n\}$. A conjunctive type is either acceptable or banned. Thus, T_C is the set of acceptable conjunctive types. Let $\{t_1,\dots,t_n\}$ be an acceptable conjunctive type, then every subset of this set is also an acceptable conjunction. Let $\{t_1,...,t_n\}$ be a banned conjunctive type, which does not mean that any subset of this conjunction is banned but that there is no entity with all these types.

Let us assume that incomparability of types in a conjunctive type is not required. Let $\{t_1,...,t_n\}$ be an acceptable conjunctive type with, for instance, $t_i \leq t_j$. Then the conjunction of t_i and t_j has the same meaning as t_i since every entity of type t_i is of type t_j, hence t_j, the greatest type, is useless. For example, let us consider $t = \{Boy, Person, SmallEntity\}$, as $Boy < Person$, t has the same meaning as $t' = \{Boy, SmallEntity\}$. If the conjunction of $\{t_1,...,t_n\}$ is banned, then t_j is useless too, because if there cannot be an entity with types $\{t_1,\ldots,t_n\}$, there cannot be an entity with types $\{t_1,\ldots,t_n\} \setminus \{t_j\}$. For example if $t = \{RacingCar, Animal, Car\}$ is banned, then as $RacingCar < Car$, t has the same meaning has $t' = \{RacingCar, Animal\}$ which is banned too.

Thus we restrict conjunctive types to the set of minimal types of a type set, and if $\{t_1,...,t_n\}$ is a set of types, then its associated conjunctive type is denoted $min\{t_1,...,t_n\}$ or $t_1 \wedge \ldots \wedge t_n$.

Due to the associativity of the conjunction operator, defined as the minimal operator, a conjunction of conjunctions of primitive types is a conjunction of primitive types. Therefore, we consider only conjunctions of primitive types.

The set of all acceptable conjunctions of primitive types can be exponentially bigger than the primitive type set. This is why the set of concept types is not defined in extension but by means of the primitive type set and a set of assertions stating which types are banned. Let us define these notions more formally.

Definition 3.1 (Primitive concept type set). A *primitive concept type set* is an ordered set $(T, \leq)$ with a greatest element denoted by $\top$.

This definition is the same as the concept type set definition (cf. Definition 2.1).

Definition 3.2 (Conjunctive concept type). A *conjunctive concept type* is given by a (non-empty) set of incomparable types $\{t_1,...,t_n\}$. This type is denoted by the set itself or by $t_1 \wedge \ldots \wedge t_n$. Let A be any (non-empty) subset of T, then the conjunctive type associated with A is defined as the set, $min(A)$, of minimal elements of A (a minimal element of A is an element t of A such that $\forall t' \in A, t' \not< t$).

A primitive type t is identified with the conjunctive type $\{t\}$. Conjunctive types are provided with a natural partial order that extends the order defined between primitive types: Given two conjunctive types t and s, t is a specialization of s if every primitive type of s has a specialization (possibly equal to itself) in t.

Definition 3.3 (Conjunctive concept type set). Let T be a set of primitive concept types. $T^{\sqcap}$ denotes the set of all conjunctive types over T. It is provided with the following partial order, which extends the partial order on T: Given two types $t = \{t_1,\ldots,t_n\}$ and $s = \{s_1,\ldots,s_p\}$, $t \leq s$ if for every $s_j \in s$, $1 \leq j \leq p$, there is a $t_i \in t$, $1 \leq i \leq n$, such that $t_i \leq s_j$.

Example. Let t=$BuidingBlock \wedge Cube \wedge Small$ and s=$Toy \wedge Cube$. $BuidingBlock \leq Toy$ and $Cube \leq Cube$ then $t \leq s$.

Property 3.1. $(T^{\sqcap}, \leq)$ is a lattice.

The previous property relies on basic results in order theory. In order theory, a subset of incomparable elements is called an *antichain*, and it is well-known that the set of antichains provided with the previous partial order is a lattice: Each pair of antichains has a greatest lower bound that is the set of minimal elements of the union of the antichains, and a least upper bound which is the set of minimal elements of the intersection of the filters generated by the two antichains. Furthermore, the inf-irreducible elements of that lattice are exactly the primitive types (cf. Appendix).
Example. Figure 3.2 represents an ordered set and the lattice of its antichains; the first set can be seen as the set T of primitive types and the lattice is then the set $T^{\sqcap}$ of conjunctive types over T.

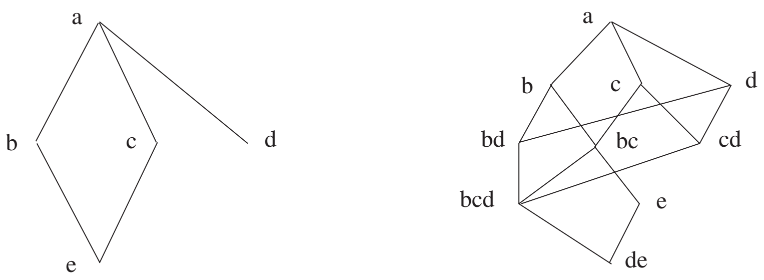

Fig. 3.2 An ordered set and the lattice of its antichains

The property of being a banned conjunctive type is hereditary: If a conjunctive type A is banned, all conjunctive types less than A are also banned. For instance, if $A = \{Car, Animal\}$ is banned then $\{RacingCar, Animal\}$ is banned too.

Definition 3.4 (Banned type set). Let $\mathcal{B}$ denote a set of conjunctive types. An element of $T^{\sqcap}$ is said to be *banned* with respect to $\mathcal{B}$ if it is less than or equal to (at least) an element of $\mathcal{B}$. $\mathcal{B}^*$ denotes the set of all banned types: $\mathcal{B}^* = \{t \in T^{\sqcap} \mid \exists t' \in \mathcal{B}, t \leq t'\}$.

In the ordered sets terminology, $\mathcal{B}^*$ is the union of the ideals generated by the banned types.
Example. Figure 3.3 presents examples of banned and non-banned types. $\{BuildingBlock, Cube, Firetruck\}$ is banned because it is less than $\{BuildingBlock, Car\}$, which is itself banned.

The set of non-banned types, i.e., of acceptable types, is obtained from $T^{\sqcap}$ by removing $\mathcal{B}^*$:

Definition 3.5 (Concept type hierarchy). A concept type hierarchy $T_C(T, \mathcal{B})$, is given by a couple $(T, \mathcal{B})$, where:

- T, the set of primitive concept types, is partially ordered by $\leq$, with a greatest element, denoted by $\top$ and called the universal type,

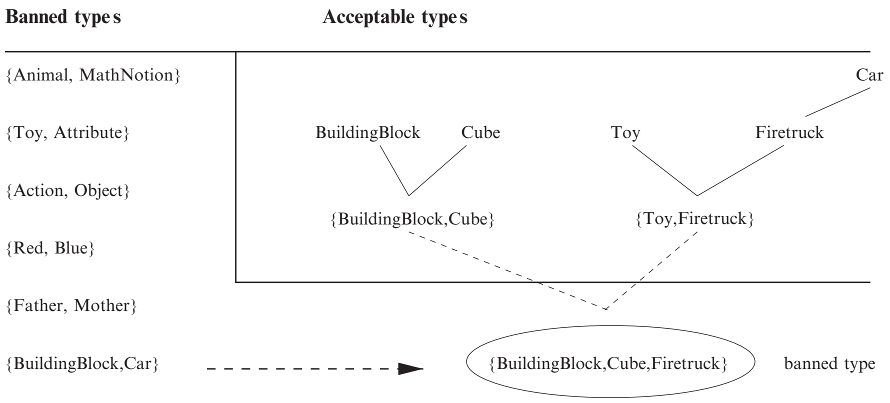

Fig. 3.3 Examples of banned and acceptable concept types

- $\mathcal{B}$, the set of basic banned conjunctive types, is composed of conjunctive types over T,
- $\mathcal{B}$ complies with T, i.e., for all $b \in \mathcal{B}$, there is no type $t \in T$ with $t \leq b$ i.e., $\mathcal{B}^* \cap T = \emptyset$.

$T_C(T,\ \mathcal{B})$ is defined as the set $T^{\sqcap} \setminus \mathcal{B}^*$. $T^{\sqcap}$ is thus partitioned into the acceptable types $T_C(T,\ \mathcal{B})$ and the banned types $\mathcal{B}^*$. Whenever there is no risk of ambiguity, $T_C(T,\ \mathcal{B})$ is simply denoted T_C.

An important particular concept type hierarchy is the case where there are no banned types, i.e., $\mathcal{B}$ is empty. In this case $T_C = T^{\sqcap}$ and T_C is a lattice. In general case, T_C is a sup-semi-lattice, i.e., each pair of types has a least upper bound.

A conjunctive vocabulary contains three parts: A hierarchy of concept types, a hierarchy of relation types, and a set of markers.

Definition 3.6 (Vocabulary). A *conjunctive vocabulary* is a triple $(T_C, T_R, \mathcal{I})$.

- $T_C, T_R, \mathcal{I}$ are pairwise disjoint sets.
- T_C is the *concept type hierarchy* defined by a set T of primitive concept types and a set $\mathcal{B}$ of banned conjunctions.
- T_R, the set of relation symbols, is ordered by a relation $\leq$, and is partitioned into subsets $T_R^1, \ldots, T_R^k$ of relation symbols of arity $1, \ldots, k$ respectively. The arity of a relation r is denoted $arity(r)$. Furthermore, any two relations with different arities are not comparable.
- $\mathcal{I}$ is the set of individual markers.

BGs (cf. Definition 2.2) can be defined on a conjunctive vocabulary and hereafter a conjunctive vocabulary is simply called a vocabulary.

The set of ordered concept labels is defined in the same way as in Chap. 2:

Definition 3.7 (Ordered set of concept labels). The ordered set of *concept labels* is the cartesian product of the ordered sets T_C and $\mathcal{I} \cup \{*\}$.

The introduction of conjunctive types gives a semi-lattice structure to the set of concept labels:

Property 3.2. The ordered set of concept labels on a conjunctive vocabulary is a sup-semi-lattice.

Proof. T_C is a sup-semi lattice since it is obtained from a lattice by removing elements while keeping the least upper bound of all remaining elements (if t and t' are acceptable, then $t \vee t'$ is acceptable too). $\mathcal{I} \cup \{*\}$ is a sup-semi-lattice. One concludes using a (simple) semi-lattice property: The product of two sup-semi-lattices is a sup-semi-lattice (indeed, let (t,m) and (t',m') be two concept labels; it is easy to check that $(lub(t,t'), lub(m,m'))$ is equal to $lub((t,m),(t',m'))$). Thus $T_C \times \{\mathcal{I} \cup \{*\}\}$ is a sup-semi-lattice. □

A set of labels therefore has a least upper bound. An entity of the application domain can be represented by several concepts. In this case the labels of these concepts have to satisfy two conditions. The first condition, already seen in Chap. 2, corresponds to the Unique Name Assumption: Two distinct individual markers cannot represent the same entity. The second condition requires that the conjunction of the types of the concepts representing an entity does not yield a banned type. These two conditions are fulfilled in the following definition.

Definition 3.8 (Compatible concept labels). The concept labels $(t_1,m_1),\ldots,(t_k,m_k)$ are *compatible* if

- there is at most one individual marker, i.e., $|\mathcal{I} \cap \{m_1,\ldots,m_k\}| \leq 1$,
- the conjunctive type $t' = min\{t_1 \cup \ldots \cup t_k\}$ is in T_C.

Property 3.3. The concept labels (t_1,m_1), ..., (t_k,m_k) are compatible if and only if these labels possess a greatest lower bound.

The greatest lower bound of concept labels (t_1,m_1), ..., (t_k,m_k) is the concept label $(t',\ m')$ with $t' = min\{t_1 \cup \ldots \cup t_k\}$ and $m' = min\{m_1,\ldots,m_k\}$, i.e., m' is the generic marker if all m_i are generic markers, otherwise m is the only individual marker appearing in the m_i.

Example. $(Object,*), (Cube,C3), (Toy \wedge Small,*)$ are compatible. Assume that $\{Young,\ Mother\}$ is not banned then $(Young,\ Judy)$ and $(Mother,\ *)$ are compatible, but $(Young,\ Judy)$ and $(Mother,\ Mary)$ are not compatible because Judy and Mary are different individuals.

3.3 Simple Conceptual Graphs (SGs)

Roughly said, an SG is a BG plus a coreference relation. A coreference relation is an equivalence relation over the concept set and two equivalent concepts are called coreferent concepts. The important properties to fulfill are, first, that coreferent concepts can always be merged, secondly, that the formal semantics of the obtained graph is equivalent to those of the original one (cf. Chap. 4).

Definition 3.9 (Compatible concept nodes). A set of *compatible concept nodes* is a set of concept nodes such that the set of their labels is compatible, i.e., these labels possess a greatest lower bound. The *label of a compatible concept set X* is defined as $l(X) = glb(\{l(x) \mid x \in X\})$.

Definition 3.10 (Simple conceptual graph). A *simple conceptual graph* (in short SG) over a vocabulary $\mathcal{V}$ is a 5-tuple $(C, R, E, l, coref)$, such that:

- (C, R, E, l) is a basic conceptual graph over $\mathcal{V}$.
- $coref$ is an equivalence relation over C, such that:

  any coref class is compatible,
  if two individual concepts have the same marker then they are in the same coref class.

In graph drawings, a coreference relation is usually only *partially* represented by coreference links (cf. Fig. 3.4). A *coreference link* connects two coreferent concepts. Coreference links are usually represented by dashed lines. Note that the coreference relation is the reflexive and transitive closure of the relation, which is the union of the set of coreference links and the relation linking two individual concept nodes having the same marker. When it is totally represented in a graph drawing, the coreference links form a "clique" on concepts belonging to the same coreference class, i.e., each pair of concepts of a coreference class is connected by a coreference link.
Example. An SG is presented in Fig. 3.4. Its coreference classes are: $\{c_1, c_2, c_3\}$, $\{d_1, d_2\}$, and the trivial classes $\{a\}$, $\{b\}$ and $\{c\}$. The coreference link between c_1 and c_3 is not represented, it is obtained by transitive closure.

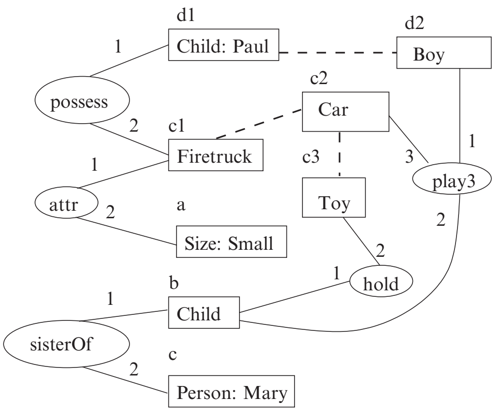

Fig. 3.4 A Simple conceptual graph

What are coreference links useful for? A first feature is that they can be exploited in *user interfaces*. Indeed, one advantage of BGs is their readability, a quality that is lost when the BG becomes too big (e.g., too big to be entirely drawn on a screen) or too complex. A BG that is too big can be split into pieces, with each piece being displayable on a screen. The split can also be performed to decompose a BG into semantic modules, or to enhance a particular substructure. Coreference links enable the representation of a BG by several pieces, while keeping its global structure. A user can build a BG piece by piece, then connect them with coreference links. Another important case is when a conceptual graph basis has been constructed by different people or at different times. Finally, coreference links can also be used to decompose a graph into a simpler structure that can be exploited in *algorithms*. See for instance the tree covering notion in Chap. 7.

We will see that it is easy to construct a BG (i.e., to get rid of the coreference relation) that is semantically equivalent to an SG. Thus, even if coreference is a useful representation tool, it does not add expressivity to BGs. The reasoning mechanisms defined for BGs can still be used, with the SGs being transformed into BGs before any computation. Let us point out, however, that coreference will play a more substantial role in more complex graphs such as nested conceptual graphs (Chap. 9) and rules (Chap. 10). It will also be a fundamental element in full conceptual graphs (Chap. 12).

Definitions concerning BGs can be directly adapted to SGs. Indeed, an SG is a BG plus a coreference relation on the set of concept nodes. Thus, let P be a notion concerning BGs, then an extension of this notion to SGs can be obtained as follows. Let $G = (C,R,E,l,coref)$ be an SG: The notion P is considered for the BG (C,R,E,l) and a condition taking the coreference relation into account is added. Let us give some examples.

Definition 3.11 (subSG). Let $G = (C,R,E,l,coref)$ be an SG. A *subSG* of G is an SG $G' = (C',R',E',l',coref')$, such that:

- (C',R',E',l') is a subBG of (C,R,E,l).
- $coref'$ is the restriction of $coref$ to C' (i.e., a class of $coref'$ is the intersection of a class of $coref$ with C').

The homomorphism notion for BGs can also be easily extended to SGs. Two coreferent concepts must have coreferent images: either the same image (since coref is a reflexive relation) or distinct coreferent images.

Definition 3.12 (SG homomorphism and $\succeq$). Let $G = (C_G,R_G,E_G,l_G,coref_G)$ and $H = (C_H,R_H,E_H,l_H,coref_H)$ be two SGs defined on the same vocabulary. A *homomorphism* π from G to H is a homomorphism from the BG (C_G,R_G,E_G,l_G) to the BG (C_H,R_H,E_H,l_H) that preserves the coreference, i.e. $\forall x,y \in C_G, coref_G(x,y) \rightarrow coref_H(\pi(x),\pi(y))$.
$G \succeq H$ means that there is a homomorphism from G to H.

An SG isomorphism is naturally defined as follows:

Definition 3.13 (SG isomorphism). Let $G = (C,R,E,l,coref)$ and $G' = (C',R',E',l',coref')$ be two SGs. G and G' are *isomorphic* if first, there is a BG isomorphism π from (C,R,E,l) to (C',R',E',l'); secondly, π preserves the coref classes, that is x and y are coreferent in G if and only if $\pi(x)$ and $\pi(y)$ are coreferent in G'.

Before studying the SG homomorphism in more detail it is useful to define generalization and specialization operations.

3.4 Generalization and Specialization Operations

In this section, the elementary generalization and specialization operations defined in Chap. 2 are extended to SGs, i.e., the coreference relation is taken into account.

We will group operations into three clusters: Equivalence operations, (plain) generalization operations and (plain) specialization operations. By "plain" we mean that these operations generally do not produce an equivalent SG although they may be equivalent in some cases. Note that when operations have the same name as for BGs, they are the same operation, i.e., the coreference relation is not changed by the operation.

Definition 3.14 (SG Equivalence operations). The elementary equivalence operations are:

- **Copy.** Create a disjoint copy of an SG G. More precisely, given an SG G, $copy(G)$ is an SG which is disjoint from G and isomorphic to G.
- **Relation duplicate.** Given an SG G and a relation r of G, $relDuplicate(G,r)$ is the SG obtained from G by adding a new relation node r' having the same type and the same list of arguments as r.
- **Relation simplify.** Given an SG G, and two twin relations r and r' (relation with the same type and the same list of neighbors), $relSimplify(G,r')$ is the SG obtained from G by deleting r'.
- **Concept split.** Split a concept into coreferent concepts. Let c be a concept node of G and $\{A_1,A_2\}$ be a partition of the edges incident to c. $split(G,c,A_1,A_2)$ is the SG obtained from G by: creating two new concept nodes c_1 and c_2 with the same label as c, attaching A_1 to c_1 and A_2 to c_2 (A_1 or A_2 may be empty), adding c_1 and c_2 to the coreference class of c, and finally deleting c.
- **Coreferent nodes merge.** Given an SG G, and two coreferent concepts c_1 and c_2 of G, merge c_1 and c_2, i.e., create a node c with the label being the greatest lower bound of c_1 and c_2 labels, attach to c all edges incident to c_1 or c_2 (replace every edge (r,i,c_1) or (r,i,c_2) with an edge (r,i,c)), add c in the coreference class of c_1 and c_2 and delete these nodes.
- **Concept label modify.** Let c be a concept node in a coreference class X of an SG $G=(C,R,E,l,coref)$. Change $l(c)=L$ into L' in such a way that the label of X is unchanged; otherwise said, the modification of the label of c does not change the glb of the label of concepts in X.

Definition 3.15 ((plain) SG Generalization operations). The elementary generalization operations are:

- **Increase.** Increase the label of a node (concept or relation). More precisely, given an SG G, a node x of G, and a label $L \geq l(x)$ $increase(G,x,L)$ is the SG obtained from G by increasing the label of x up to L, i.e., its type if x is a relation, its type and/or its marker if x is a concept.

- **Coreference delete.** Split a coreference class into two (non-empty) classes.
- **Substract.** Given an SG G, and a set of connected components $C_1,\dots,C_k$ of G, $substract(G,C_1,\dots,C_k)$ is the SG obtained from G by deleting $C_1,\dots,C_k$ (the result may be the empty graph).

Definition 3.16 ((plain) SG Specialization operations). The elementary specialization operations are:

Restrict. Given an SG G, a node x of G, and a label $l \leq l(x)$ $restrict(G,x,l)$ is the SG obtained from G by decreasing the label of x to l that is its type if x is a relation, its type and/or its marker if x is a concept. The compatibility condition has to be preserved.
Coreference add up. Make the union of two coreference classes of G provided that their union is a compatible set.
Disjoint sum. Given two disjoint SGs $G = (C,R,E,l,coref)$ and $H = (C',R',E',l',coref')$, $G+H = (C\cup C',R\cup R',E\cup E',l\cup l',coref\cup coref')$ (G or H may be empty).

Generalization and specialization of SGs are defined similar to BGs as a sequence of elementary operations.

Definition 3.17 (SG generalization and specialization operations). The set of generalization operations is composed of equivalence operations and plain generalization operations. The set of specialization operations is composed of equivalence operations and plain specialization operations. Given two SGs G and H, G is a generalization (resp. specialization) of H if there is a generalization (resp. specialization) sequence from G to H.

The disjoint sum operation is not the inverse of the substract operation, but the inverse of a substract operation can be performed by a disjoint sum plus a coreference addition, thus:

Property 3.4. Given two SGs G and H, G is a generalization of H if and only if H is a specialization of G.

Definition 3.18 (Gen-Equivalence). Given two SGs G and H they are called *gen-equivalent* if G is a generalization (or specialization) of H and reciprocally.

Operations on BGs are naturally included in operations on SGs, either directly under the same name, or they can be performed by a combination of operations of the same category on SGs. Indeed, a detach operation for SGs can be obtained by a split into coreferent nodes followed by a coreference deletion, which are both generalization operations, and a join for SGs can be obtained by a co-reference addition followed by a merging of co-referent nodes, which are both specialization operations.

3.5 Standard and Normal SGs

A *normal* SG is such that $coref$ is the identity relation, i.e., each node is uniquely coreferent with itself. In the notation of a normal SG, $coref$ is usually omitted; a normal SG is thus simply denoted by (C,R,E,l), as a (normal) BG. This is a consistent notation because a BG can be considered as an SG with the following implicit coreference relation: Each generic concept constitutes a class and all individual concepts with the same marker are coreferent.

Thus, normal SGs and normal BGs can be *identified*. Furthermore, a normal SG can be associated with any SG G by considering the SG $G/coref$, obtained by merging all concepts belonging to a coreference class. *Merging* a coreference class X consists of two successive sequences of elementary specialization operations: First, all concept labels in X are restricted to $l(X) = glb(\{l(c) \mid c \in X\})$, then the class is condensed to a single concept by a merge coreferent nodes operation. The first step is called the *standardization* of an SG G. The result is the *standard form* of G.

Definition 3.19 (Standard SG). A *standard* SG is an SG such that all coreferent nodes have the same label. The *standard form* of an SG G, denoted $stand(G)$, is obtained from G by restricting the label of each concept in a coreference class X to $l(X) = glb(\{l(x) \mid x \in X\})$.

Example. Figure 3.5 presents the standard form of the SG in Fig. 3.4.

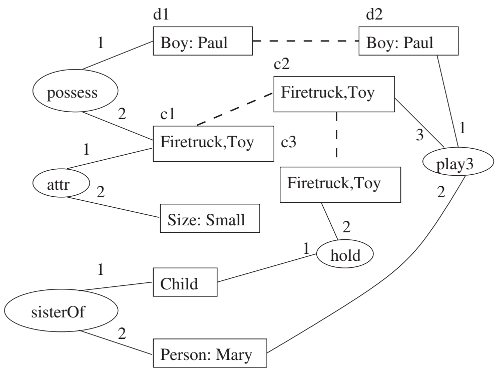

Fig. 3.5 Standard form of the SG in Fig. 3.4

The normal form of an SG G is indifferently denoted $norm(G)$ or $G/coref$. The latter notation shows that the normal form can be computed by first computing the (ordinary) quotient graph $graph(G)/coref'$, where $coref'$ is obtained from $coref$ by adding trivial classes corresponding to the relation nodes of G (each relation node constitutes a class), then by giving the correct labels to the nodes.

Definition 3.20 ($G/coref$ and $norm(G)$). Let $G = (C,R,E,l,coref)$ be an SG. $G/coref$ is the normal SG obtained as follows: for any coref class $X = \{c_1,\dots,c_k\}$, all c_i are merged into one concept X. $G/coref$ is also called the *normal form* of G, and denoted by $norm(G)$.

The normal form of the SG G in Fig. 3.4 is the SG in Fig. 3.6. Note that the standard and normal forms can be computed in linear time (relative to the size of the original graph) if the computation of the glb of k concept labels is in $O(k)$.

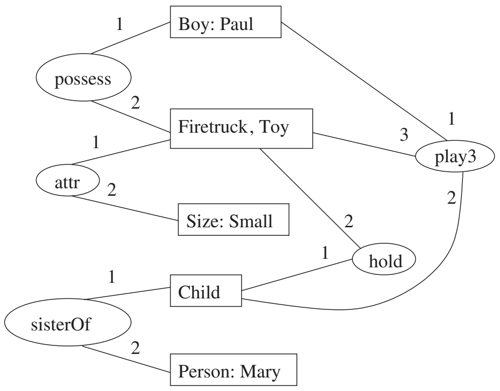

Fig. 3.6 $G/coref$

The three SGs G, $stand(G)$ and $norm(G)$ are gen-equivalent in the sense of Definition 3.18.

Property 3.5. For any SG G, the three SGs G, $stand(G)$, and $norm(G)$ are gen-equivalent.

Proof. One can see that there are sequences of equivalence operations for transforming any SG in $\{G, stand(G), norm(G)\}$ into any other. $stand(G)$ is obtained from G by a sequence of *concept label modify* operations. $norm(G)$ is obtained from $stand(G)$ by a sequence of *coreferent node merge* operations, finally G is obtained from $norm(G)$ by a sequence of *concept split* and *concept label modify* operations. $\square$

Intuitively, G, $stand(G)$ and $norm(G)$ have the same meaning. We will see later that their formal semantics are indeed equivalent.

3.6 Coref-Homomorphism

Let us now study the relationships between an SG G, its standard form $stand(G)$ and its normal form $norm(G)$ with respect to SG homomorphism. The following property is immediate:

Property 3.6. Let G be an SG. There is a homomorphism from G to $stand(G)$ and a homomorphism from $stand(G)$ to $norm(G)$. Thus: $G \succeq stand(G) \succeq norm(G)$.

The homomorphisms associated with the sequence of equivalence operations described above yielding $stand(G)$ and $norm(G)$ are called canonical homomorphisms from G to $stand(G)$ and $norm(G)$. The former is restricted to the identity (it is a bijective homomorphism) and the latter is the surjective mapping naturally associated with the quotient operation $G/coref$. Thus, as natural as it may seem, the SG homomorphism notion is not entirely satisfactory.

The converse homomorphisms do not generally occur (unless G is standard or normal).

We are faced with the problem we had foreseen on BGs: Due to the presence of coreferent nodes, graphs with the same intuitive semantic may not be equivalent for homomorphism. Let us go into further detail.

Property 3.7. Given SGs G and H, there is a bijective mapping between the set of homomorphisms from G to $norm(H)$ and the set of homomorphisms from $norm(G)$ to $norm(H)$.

Proof. Let π_G be the canonical homomorphism from G to $norm(G)$. Let P be the mapping that associates the homomorphism $\pi' = \pi \circ \pi_G$ from G to $norm(H)$ with any homomorphism π from $norm(G)$ to $norm(H)$.
P is injective because, given two distinct homomorphisms π_1 and π_2 from $norm(G)$ to $norm(H)$ and any node x of $norm(G)$, if $\pi_1(x) \neq \pi_2(x)$, then for any node y with $\pi_G(y) = x$, $\pi_1(\pi_G(y)) \neq \pi_2(\pi_G(y))$.
Let us prove that P is surjective. Let π' be any homomorphism from G to $norm(H)$, then we prove that it can be decomposed into $\pi \circ \pi_G$, where π is a homomorphism from $norm(G)$ to $norm(H)$. Indeed, π is built as follows: For every node x in $norm(G)$, if x is a node of G, in particular if x is a relation node, then $\pi(x) = \pi'(x)$. Otherwise let $c_1...c_n$ be the nodes of G merged into x in $norm(G)$; by definition of π_G, for all these nodes c_i, $\pi_G(c_i) = x$, and since π' is a homomorphism and $norm(H)$ is normal, all c_i have the same image by π', then, given any c_i, one takes $\pi(x) = \pi'(c_i)$. By definition of coreference and homomorphism, for any $i = 1,\dots,n$ $label(c_i) \geq label(\pi'(c_i)) = label(\pi(x))$, so $label(x) = glb(\{label(c_i) \mid i = 1,\dots,n\}) \geq label(\pi(x))$. Furthermore, let (r,i,x) be in $norm(G)$ then $\pi(r) = \pi'(r)$. If x belongs to G then $\pi(x) = \pi'(x)$. As π' is a homomorphism, $(\pi'(r),i,\pi'(x)) = (\pi(r),i,\pi(x))$ is in $norm(H)$. Otherwise, x is obtained from $c_1...c_n$ and there is a c_j, $1 \leq j \leq n$, with (r,i,c_j) in G. As π' is a homomorphism, $(\pi'(r),i,\pi'(x)) = (\pi(r),i,\pi(x))$ is in $norm(H)$. Thus, π is a homomorphism. Check that $\pi \circ \pi_G = \pi'$.
P is injective and surjective so it is bijective. □

Figure 3.7 shows Property 3.7 as a commutative diagram.

Thus, in order to obtain a "good" SG homomorphism behavior, one only has to ensure that the target graph H is in normal form, regardless of whether the source graph G is normal or not.

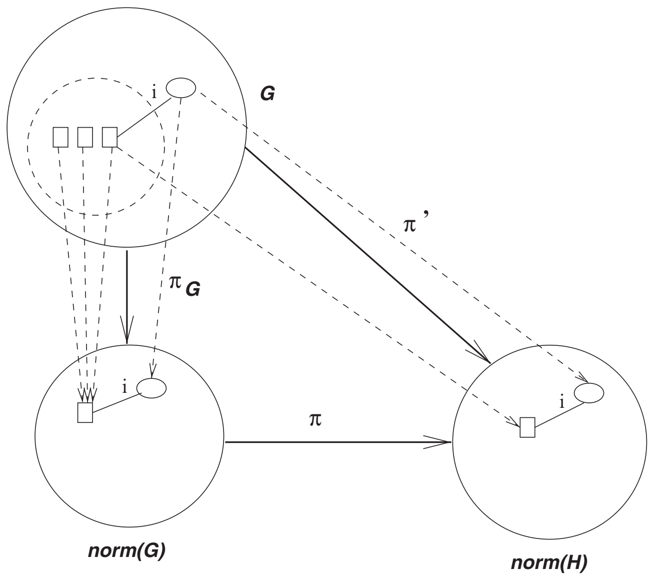

Fig. 3.7 Illustration of Property 3.7

The normal form of an SG always exists and is easily computable, however in some cases it may be important to keep SGs exactly as they are. For example, consider a set of SGs over the same vocabulary built by different users. If a query is made on this base the whole base has to be considered to compute the answers, but the SGs must not be changed. Another example is that of a base distributed on several sites.

The question now is the following: Why does homomorphism not deal correctly with coreference? The reason is that it is first of all *a mapping on concepts*; if we modify it so that it maps coreference classes onto coreference classes, we obtain the desired notion.

Definition 3.21 (coref-homomorphism). Let $G=(C_G,R_G,E_G,l_G,coref_G)$ and $H=(C_H,R_H,E_H,l_H,coref_H)$ be two SGs defined on the same vocabulary. A *coref-homomorphism* from G to H is a mapping π from $coref_G$ to $coref_H$ and from R_G to R_H, such that:

1. $\forall (r,i,c) \in G$, let C be the coreference class of c, then there is a concept $c' \in \pi(C)$ such that $(\pi(r),i,c') \in H$,
2. $\forall C \in coref_G$, let $l_H(\pi(C))$ be the greatest lower bound of the concept labels in $\pi(C)$ then, for all $c \in C$, $l_G(c) \geq l_H(\pi(C))$,
3. $\forall r \in R_G$, $l_G(r) \geq l_H(\pi(r))$.

Each homomorphism defines a coref-homomorphism, since all coreferent nodes have their images in the same coreference class. The following property specifies the relationships between both notions.

Property 3.8. Given SGs G and H, there is a bijective mapping between the set of coref-homomorphisms from G to H and the set of homomorphisms from $norm(G)$ to $norm(H)$.

Proof. There is a bijection, say b, between concept nodes of $norm(G)$ (resp. $norm(H)$) and coreference classes in G (resp. H). Let us show that b defines the desired mapping between the set of coref-homomorphisms from G to H and the set of homomorphisms from $norm(G)$ to $norm(H)$. Let π be a coref-homomorphism from G to H. Let π' be the induced mapping from nodes of $norm(G)$ to nodes of $norm(H)$: For each concept c in $norm(G)$, $\pi'(c) = b^{-1}(\pi(b(c)))$ and for each relation r of $norm(G)$, $\pi'(r) = \pi(r)$.

This correspondence is injective. Indeed, let π_1 and π_2 be two coref-homomorphisms from G to H. There is a concept c in $norm(G)$ with $\pi_1(c) \neq \pi_2(c)$, then $\pi_1'(c) \neq \pi_2'(c)$.

We check that π' is a homomorphism from $norm(G)$ to $norm(H)$: For every edge (r,i,c) in $norm(G)$, there is an edge (r,i,d) in G where d is in the coreference class $b(c)$, thus, due to condition 1 of the coref-homomorphism, there is an edge $(\pi'(r),i,d')$ in H where d' is in the coreference class $\pi(b(c))$. Thus an edge $(\pi'(r),i,\pi'(c))$ since $\pi'(c) = b^{-1}(\pi(b(c)))$. For every concept c in $norm(G)$, condition 2 of coref-homomorphism ensures that for all $c_i \in b(c)$, $l(c_i) \geq l(\pi'(c))$; thus by definition of a greatest lower bound, $l(c) \geq l(\pi'(c))$. In the same way, we check that any homomorphism π' from $norm(G)$ to $norm(H)$ yields a coref-homomorphism π from G to H using bijection b. For every edge (r,i,c_j) in G, let C be the corefence class of c_j, there is an edge (r,i,c) in $norm(G)$, where $C = b(c)$, hence an edge $(r,i,\pi'(c))$ in $norm(H)$, thus an edge (r,i,c_j') in H such that c_j' belongs to the coreference class $b(\pi'(c))$, which is exactly $\pi(C)$. For every coreference class C in G, for every c in C, $l(c) \geq l(\pi'(b^{-1}(C)))$, which is exactly $l(\pi(C))$, thus $l(c) \geq l(\pi(C))$.

Thus, the correspondence between coref-homomorphisms from G to H and homomorphisms from $norm(G)$ to $norm(H)$ is surjective and injective. □

The following theorem summarizes the “safe” ways of comparing two SGs.

Theorem 3.1. *Let G and H be two SGs. There is a bijective mapping between:*

- *coref-homomorphisms from G to H,*
- *homomorphisms from $norm(G)$ to $norm(H)$,*
- *homomorphisms from G to $norm(H)$.*

The intuitive meaning of generalization or specialization operations is better captured by coref-homomorphism than by homomorphism. This is shown in the following theorem for SGs, which is similar to the Theorem 2.3 for BGs where homomorphism is replaced by coref-homomorphism:

Theorem 3.2. *Let G and H be two SGs. The three following propositions are equivalent:*

1. *H is a specialization of G,*
2. *G is a generalization of H,*

3. there is a coref-homomorphism from G to H.

Proof. The equivalence between (1) and (2) has already been stated (cf. Sect. 3.4).
(1) $\Rightarrow$ (3): in a specialization sequence from G to H, each specialization step, say from H_i to H_{i+1}, defines a coref-homomorphism from H_i to H_{i+1}. By composition of these coref-homomorphisms, one obtains a coref-homomorphism from G to H.
(3) $\Rightarrow$ (1): Let π be a coref-homomorphism from G to H. Let us show that H is a specialization of G. From G, one builds $norm(G)$ using the equivalence operation *coreferent nodes merge*. By Property 3.8, there is a homomorphism from $norm(G)$ to $norm(H)$. Thus by Theorem 2.3, there is a BG specialization sequence from $norm(G)$ to $norm(H)$. This BG specialization sequence can be rewritten as an SG specialisation sequence, with the join operation being replaced either by a *coreferent nodes merge* if the joined nodes are individual nodes, or decomposed into a coreference addition followed by a *coreferent nodes merge* if the joined nodes are generic nodes. By Property 3.5, there is a specialization sequence from $norm(H)$ to H. □

3.7 Antinormal Form and Homomorphism

The normal form of an SG is, in a sense, the most compact form of an SG. The antinormal form studied in this section can be considered as the most scattered form of an SG: Each relation node is disconnected from the other relation nodes and there are no multiple edges. Thus, a relation node and its neighbors can be considered as a tuple. The antinormal form of an SG can be considered as a representation of an SG by tables in the database relational model. Indeed, gathering all tuples corresponding to relation nodes with the same label is equivalent to building a table. Then usual database systems and operations can be used. The correspondence between conceptual graphs and relational structures is studied in Chap. 5. In this section, it is assumed that an SG has no isolated concept node, and it is straightforward to extend all the results if isolated concepts exist.

Definition 3.22 (Antinormal SG). An SG is called *antinormal* if:

- any concept node is incident to exactly one edge,
- it is standard, i.e., all the concept labels in any coref class are identical.

For any SG G *antinorm(G)* is the SG obtained from G as follows: First it is standardized; then each concept node c with $k > 1$ incident edges, say $e_1, \dots, e_k$, is split into k coreferent nodes $c_1, \dots, c_k$, with each edge $e_i = (r, j, c)$ becoming an edge (r, j, c_i). $antinorm(G)$ is obtained from G by a sequence of specialization operations, moreover:

Property 3.9. Let G be an SG. There is a homomorphism from $antinorm(G)$ to $stand(G)$, i.e., $antinorm(G) \succeq stand(G)$.

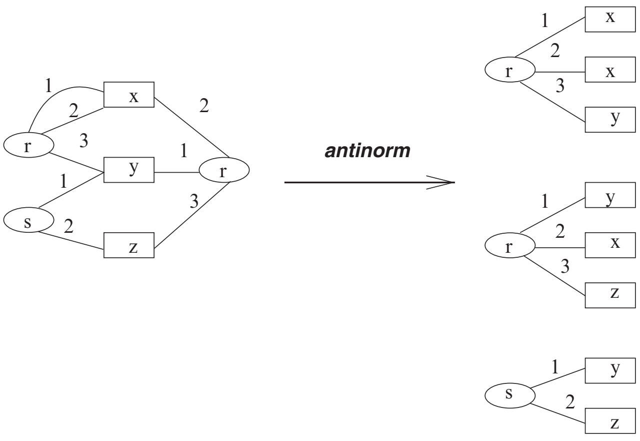

Fig. 3.8 An SG and its antinormal form

Example. Figure 3.8 presents an SG and its antinormal form. In this figure, the concepts labeled by x (or by y or by z) are coreferent.

Figure 3.9 shows that $antinorm(G)$ does not generally map to G if G is not standard.

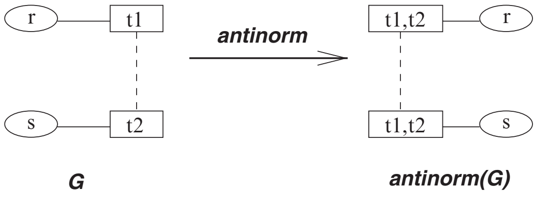

Fig. 3.9 antinorm(G) does not map to $\succeq G$

Note that the relation node sets of $antinorm(G)$ and G are equal, even if G is not standard.

If an SG is not antinormal then it does not generally map to its antinormal form by a homomorphism (cf. example given in Fig. 3.8).

The subsumption relation behaves in the same way for the normal forms and antinormal forms.

Property 3.10. Given two SGs G and H, there is a bijection from the set of homomorphisms from $norm(G)$ to $norm(H)$ to the set of homomorphisms from $antinorm(G)$ to $antinorm(H)$.

Proof. Let π be a homomorphism from $norm(G)$ to $norm(H)$ and ϕ_H be the canonical homomorphism from $antinorm(H)$ to $norm(H)$. A homomorphism π' from $antinorm(G)$ to $antinorm(H)$ can be constructed as follows. For any relation node,

$\pi'(r) = \pi(r)$ since the relation nodes of an SG and its normal form are the same. Let c be a concept node of $antinorm(G)$. c is in the coreference class of a concept d of $norm(G)$ and is incident to a unique edge (r,i,c). There is a unique concept c' of $antinorm(H)$, which corresponds to the edge $(\pi(r),i,\pi(d))$ in $norm(H)$. One takes $\pi'(c) = c'$ (cf. Fig. 3.10). If one considers two different homomorphisms from $norm(G)$ to $norm(H)$ the associated homomorphisms from $antinorm(G)$ to $antinorm(H)$ are also different. Reciprocally, let π' be a homomorphism from $antinorm(G)$ to $antinorm(H)$. The mapping π defined as follows is a homomorphism from $norm(G)$ to $norm(H)$. To any relation node r of $norm(G)$ $\pi(r) = \pi'(r)$. Let d be a concept of $norm(G)$. For any (r,i,d) in $norm(G)$ corresponds a single c in $antinorm(G)$ with (r,i,c) in $antinorm(G)$. $(\pi'(r),i,\pi'(c))$ is in $antinorm(H)$. One takes $\pi(d) = \phi_H(\pi'(c))$.

This does not depend on the choice of the edge (r,i,d) incident to d since for any other edge (r',j,d) in $norm(G)$ corresponding with the node c' in $antinorm(G)$, we have $\phi_H(\pi'(c)) = \phi_H(\pi'(c'))$ by the preservation of coreference by π'.

Figure 3.10 illustrates this part of the proof. Indeed, one can say that the diagram in Fig. 3.10 commutes. If one considers two different homomorphisms from $antinorm(G)$ to $antinorm(H)$, the associated homomorphisms from $norm(G)$ to $norm(H)$ are also different. The construction is injective in the two directions, so there is a bijection between the two homomorphism sets. □

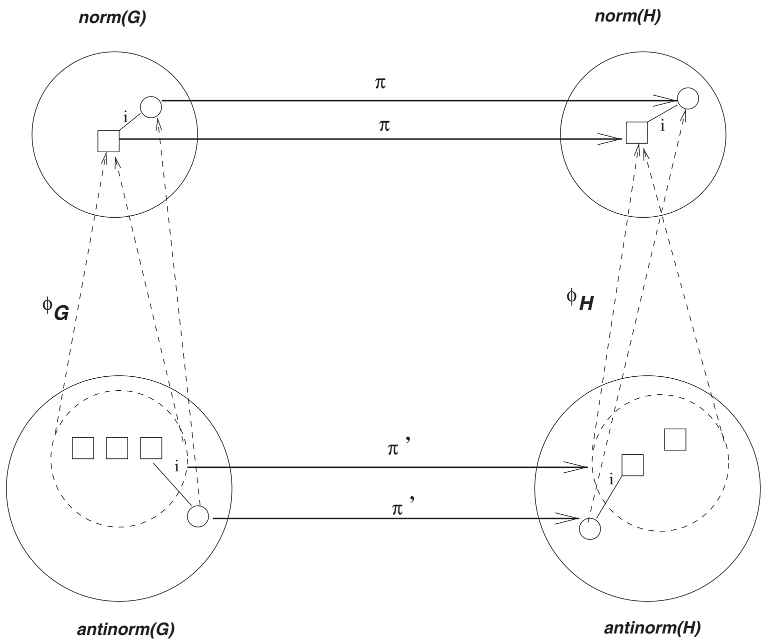

Fig. 3.10 Illustration of Property 3.10

Let us use an example to illustrate the importance for Property 3.10 of the last condition in Definition 3.22 of an antinormal SG. Figure 3.11 presents two non-standard SGs G_1 and G_2, where the relation types r and s are not comparable. These SGs satisfy the first antinormality condition since they are totally scattered. One has $norm(G_1) = norm(G_2) = G$, but there is neither a homomorphism from G_1 to G_2 nor from G_2 to G_1.

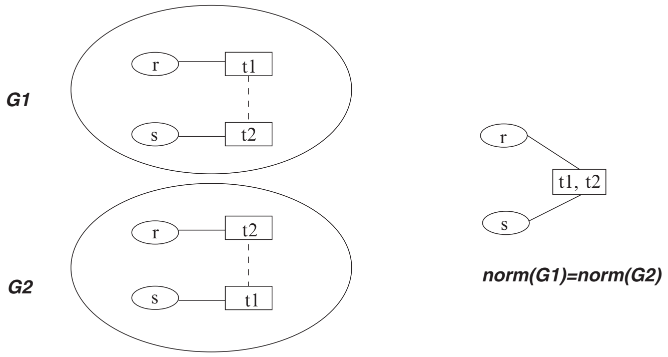

Fig. 3.11 G_1 and G_2 are not antinormal SGs

The following property shows the difference between the behavior of the homomorphism for the normal form and the antinormal form (cf. Property 3.7).

Property 3.11. Let G and H be two SGs. If $antinorm(G) \succeq H$, then $antinorm(G) \succeq antinorm(H)$. When H is standard, the converse is true: If $antinorm(G) \succeq antinorm(H)$ then $antinorm(G) \succeq H$.

Proof. Let π be a homomorphism from $antinorm(G)$ to H. A homomorphism π' from $antinorm(G)$ to $antinorm(H)$ can be constructed as follows. For any relation node, $\pi'(r) = \pi(r)$ since the relation nodes of an SG and its antinormal form are the same. Let (r,i,c) in $antinorm(G)$. $(\pi(r),i,\pi(c))$ is in H and the label of c is greater than or equal to the label of $\pi(c)$. There is a single concept d in $antinorm(H)$ which is adjacent to an edge numbered i to $\pi(r)$. The label of $\pi(c)$ is greater than or equal to the label of d and one takes $\pi'(c) = d$ (cf. Fig. 3.12). Let us assume that H is standard. Property 3.9 and the transitivity of $\succeq$ yield the second part of the property. $\square$

Example. Let us consider an SG H isomorphic to the SG G on the left side in Fig. 3.9). There is a (trivial) homomorphism from $antinorm(G)$ to $antinorm(H) = antinorm(G)$, but there is no homomorphism from $antinorm(G)$ to H.

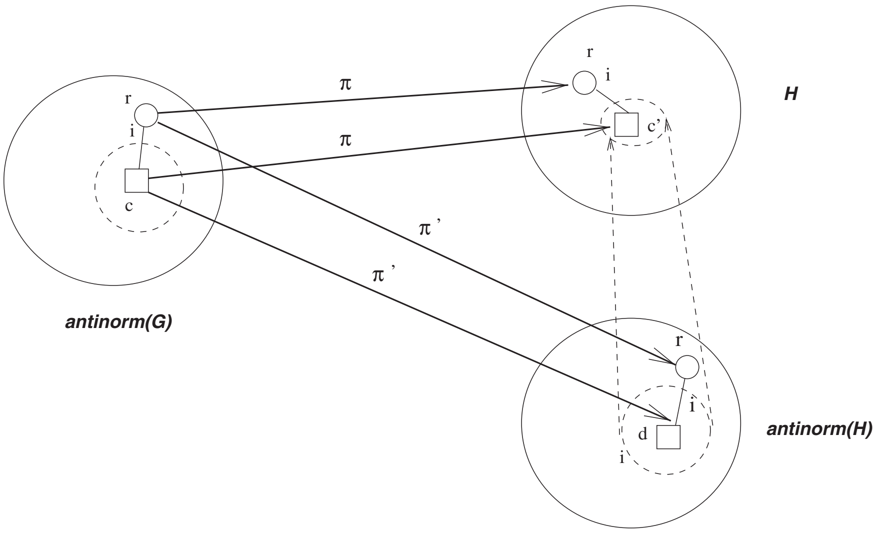

Fig. 3.12 Illustration of Property 3.11

3.8 Bibliographic Notes

The notion of a *coreference link* was introduced in [Sow84], mainly as a means of representing an anaphoric reference in the context of natural language processing.

In [Sow84] the concept type set is a *lattice*. In later works, it becomes a partially ordered set with a greatest element, thus a less constrained structure. The distinction between lattice-theoretical and order-theoretical interpretations is from [BHP^{+}92]), and its effect on reasoning with CGs was pointed out in [WL94] and [CCW97].

The framework presented in this chapter is based on [CM04], which itself gathers and generalizes several previous works. Let us mention [CMS98], in which coreference (called co-identity) was defined as an equivalence relation on concept nodes. In this work, coreferent nodes were either individual nodes with the same label, or generic nodes with the same type, with these restrictions being related to properties (a) and (b) discussed in the introduction to this chapter. The extension of generalization and specialization operations to this restricted coreference relation was presented in [Mug00]. Type conjunction in conceptual graphs was first considered in the context of fuzzy types [CCW97] (and later [TBH03a] [TBH03b]) or in relationship with description logics [BMT99]. In [Bag01], the set of conjunctive concept types was defined in intension by a hierarchy of primitive concept types; type conjunction was also defined for relations, on the condition that these relations have the same arity. [CM04] mainly added the notion of banned conjunctive type and coref-homomorphism. Note that if we allow to name conjunctive types, we obtain a simple type definition mechanism, with conjunction as the only constructor. More complex type definition mechanisms are mentioned in Chap. 8.

According to the unique name assumption, different individual markers represent different entities. An exception to this common assumption is the framework of [Dau03a] which allows coreference between concept nodes with different individual markers. In other words, there are aliases among the individual markers. This case could be included in our framework with slight modifications.

The notion of antinormal form of an SG was introduced in [GW95]. The definition did not include a standardization step. It was thus well-suited to the restricted case where all coreferent nodes have the same label but not to the general case.