Conceptual Graphs: fundamental notions

Michel Chein et Marie-Laure Mugnier

LIRMM (CNRS & Université de Montpellier II)
860 rue de Saint-Priest,
34090 Montpellier
e-mail: chein@crim.fr, mugnier@crim.fr

ABSTRACT. We state precise definitions of the basic notions of Sowa's framework [Sowa 84] and provide related results. These results mainly concern the structure of the specialization relation, correspondence between graph operations and logical operations, and algorithmic complexity of the model handling.

RESUME. Nous définissons précisément les notions de base du modèle des graphes conceptuels de Sowa [Sowa 84] et en étudions les propriétés. Nos résultats portent principalement sur la structure de la relation de spécialisation, la correspondance entre opérations de graphes et opérations logiques, et la complexité algorithmique de la mise en oeuvre du modèle.

KEY WORDS : Knowledge representation, conceptual graphs, Sowa's graphs, semantic networks, specialization / generalization, graphs, pattern matching, first order logic, algorithmic complexity.

MOTS-CLES : représentation de connaissances, graphes conceptuels, graphes de Sowa, réseaux sémantiques, spécialisation / généralisation, graphes, appariement, logique du premier ordre, complexité algorithmique.

Revised version of R.R. CRIM, April 91. *Appeared in* Revue d'Intelligence Artificielle, Vol. 6, n° 4, 1992, pp 365--406.

1. Introduction.

The motives leading to this paper can be summarized as follows.

First has been the deep interest arousen by the lecture of Sowa’s book on conceptual structures [Sowa 84] and particularly his unifying approach and elegant simplicity. Then the conflicting initial reactions which followed the publication of his book [Clancey 85] [Smoliar 87], and the response given by Sowa [Sowa 88], enhanced our curiosity. Finally the great number of research works and practical realizations presented in regular workshops have confirmed both the pertinence of Sowa's ideas and the need for formalization due to the lack of precision of some notions and the persistence of some errors (see [AWCG 88] ... [AWCG 92] for the workshop proceedings and [JETAI 92], [KBS 92], [Nagle Ed. 92] for selected papers; some of the larger practical projects are listed in [Sowa 92, § 4]).

“Conceptual graphs form a knowledge representation language based on linguistics, psychology, and philosophy” [Sowa 84]. Sowa proposes an abstract model which can be used at different levels. At a conceptual level, it can be the basis for a specialized communication language between specialists of different disciplines involved in a common cognitive work. At an implementation level, it can be the basis for a common representation tool used by several modules of a complex system, integrating: knowledge and databases, inference engines, sophisticated human-computer interfaces, learning modules …

In his book, Sowa settles a relatively simple model and, progressively, adds more complex notions. This is the basic model that we consider here. Our main objective is to propose precise definitions, theorems and algorithmic complexity results.

A large part of the notions and results presented here are explicitly or implicitly present in Sowa’s book. Nevertheless, for completeness reasons, and so that people not familiar with Sowa’s conceptual graphs can read this paper without referring to the book, we will be giving here some definitions and results of [Sowa 84].

Second part is devoted to the definition of a conceptual graph - that we call *S-graph* - related to a *support*. Indeed, *“a conceptual graph has no meaning in isolation. Only through the semantic network are its concepts and relations linked to context, language, emotion, and perception”* [Sowa 84]. Sowa described partially and separatly some components of this semantic network. Our notion of a support integrates these elements. The support represents general knowledge on a domain, while each S-graph asserts a single proposition related to this context.

In sections 3 to 6, we study the ground operations on S-graphs. Some are based on the *projection* notion, which is a particular graph morphism, other are based on *generalization / specialization* rules. We give a complete correspondence between these two views.

We precise in section 7 that the specialization relation - induced by the composition of specialization rules - is not an order as many people think, but a *preorder*, that is a reflexive and transitive but not antisymmetric relation.[1] This is an important point, because we prove in the last part that the problem of determining whether two S-graphs are equivalent, for the equivalence relation associated with the preorder, is NP-complete. We characterize the notion of an *irredundant* graph, which is a unique representative of its equivalence class.

In section 8, we study different acceptances of the *extended join* operation, which can be seen as a partial match between two graphs.

Sowa has proposed a *logical interpretation* of conceptual graphs, and consequently of specialization operations. Section 9 is devoted to this correspondence between graphs and logic.

Finally, we establish *algorithmic complexity* results of the main problems encountered in the preceding sections. Most of them are NP-complete in the general case and polynomial when the underlying graph of the S-graph is a tree.

Let us now place our work in reference to Sowa's book, more precisely to the basic model introduced in Chapter 3 of [Sowa 84][2].

We first explicit the notion of a *support* from several concepts introduced explicitly or implicitly by Sowa as components of the "semantic network" or of the "canon" ([Sowa 84], 3.4.5). We introduce the notion of an *S-graph* related to a given support - *S* for Sowa, for simple, and also for recalling the support - which corresponds to a *conceptual graph* having only generic or individual markers (as conceptual graphs introduced in 3.1.2 of [Sowa 84]).

In the third section, we give precise definitions for the following graph notions: *projection* ([Sowa 84], 3.5.4), *S-graph morphism* and *sub-S-graph*.

---

[1]This fact has been noted several times (see for instance [Chein 89] - after a remark of O. Cogis -, [Ellis 91], [Jackman 88]).
[2]Following paragraphes are not mandatory for the paper understanding. They only aim at precisely placing our contribution in Sowa's book context.

We recall in fourth part Sowa's elementary *specialization rules - restrict, join and simplify* - ([Sowa 84], 3.4.3) without the *copy* rule that can be considered as a particular case of the restrict rule. We explicitly introduce the *generalization rules*, duals of the specialization rules, used in particular to prove a result of logical completeness in the ninth section. Other specialization operations that we define are *concept nodes fusion* (in part 4), which is a composition of restrict and join operations, and *relation nodes fusion* (in part 6).

Sowa demonstrated in 3.5.4 of [Sowa 84] that, given two graphs G and H, if G is a specialization of H then there is a projection from H to G. We provide the reciprocal property, yielding a complete correspondence between the two notions (theorem 1, part 5).

Section 7 precises that the specialization relation is only a *preorder* and not an order as stated in [Sowa 84], 3.5.2. Indeed, it is not an antisymmetric relation. We introduce the notion of an *irredundant* S-graph, yielding an exact characterization of the equivalence relation associated with the preorder (theorems 2 and 3). However, this characterization is not polynomial, as shown in last section.

In section 8, we study the notion of an *extended join* between two S-graphs, and define a particular case called *isojoin* (which seems to be that actually considered in [Sowa 84] 3.5.6 up to 3.6.9). As finding a maximal isojoin - in the cardinal or inclusion sense - is NP-complete, we propose a notion of maximal isojoin, which is polynomially computable and seems to be the notion pratically used.

Section 9 is devoted to relationships with logic. We recall the logical interpretation of conceptual graphs given by Sowa ([Sowa 84] 3.3.2), and supplement this with a logical interpretation of the concept type lattice. Sowa proved in 3.5.3 that specialization rules are refutation rules, that is, given two conceptual graphs G and H, if G is a specialization of H, then the interpretation of G logically implies the interpretation of H. Using this result, the duals of the specialization rules, generalization rules, are inference rules. We establish that they constitute a complete set of inference rules on the set of the logical formulas associated with S-graphs (theorem 4).

2. S-graph related to a Support.

A *conceptual graph* is made up of two kinds of nodes: *relations* and *concepts.*

"In the graphs, concept nodes represent entities, attributes, states, and events, and relation nodes show how the concepts are interconnected" [Sowa 84]. Some concept vertices are said to be *individual* and design particular entities, the others are called *generic* and represent unspecified entities of a given type.

A conceptual graph is related to a *support*, which defines syntactic constraints, and provides background information on a specific application domain.

In this notion of a *support*, we regroup:
a set of *concept types*, structured in a lattice, representing an AKO - is-a-kind-of - hierarchy, and allowing multiple inheritance,
a set of *relation types*,
a set of star graphs, that we call *basis*, showing for every relation type what kind of concept types it can link (in other words, a star graph represents the signature of a relation type),
a set of *markers* for concept vertices: one generic marker and individual markers which allow to distinguish and name distinct entities,
a *conformity relation*, which defines association constraints between a concept type and a marker (intuitively, if we allow the association of a type *t* with a marker *m*, this means "there is an individual *m* that IS-A *t*").

One important point is the separation and structuration of knowledge into two levels: relationships between concept types or *classes* (*i.e.* the link AKO) which belong to the support, and relationships among individuals or *instances* which are encoded in a particular conceptual graph.

Figure *2* partially exemplifies the notion of a support, with a concept type lattice and a basis, and Figure *3* gives a sample of conceptual graphs definable on this support.

Definition.

A **support** is a 5-tuple $\boldsymbol{S=(T_c,T_r,M,conf,B)}$ where:

(1) $\boldsymbol{T_c}$, the set of concept types, is a finite lattice with:
$\leq$ as order, ***1*** as supremum (the universal type), ***0*** as infinum (the absurd type), $\wedge$ and $\vee$ denoting the lower and upper bounds,

(2) $\boldsymbol{T_r}$, the set of relation types, is a finite set; T_c and T_r are disjoint,

(3) $\boldsymbol{B}$ is a set of "star graphs", $\{B_{ri},\ r_i \in T_r\}$, in bijection with T_r; every B_{ri} is built as follows: exactly one vertex of B_{ri} is labelled by the element r_i of T_r, this vertex has a non empty and totally ordered set of neighbours, these neighbours being pairwise non adjacent, and each of them is labelled by an element of T_c. *(See Figure 1).*

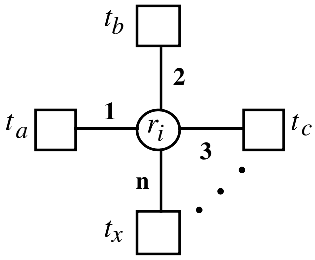


Figure 1. *The star graph B_{ri} associated with $r_i \in T_r$*

(4) $\boldsymbol{M}$ is a countable set of individual markers; in addition, there exist a marker called generic $*$ and an absurd marker $\boldsymbol{0}$; we provide the set $M \cup \{*, 0\}$ with a lattice structure by the order, denoted by $<$, such that any two elements of M are incomparable, and $\forall\, m \in M\colon\ 0 < m < *$.

(5) ***conf***, the conformity relation, is a predicate on $T_c X(M \cup \{*, 0\})$ satisfying:
$\forall\, m \in M$ and $\forall\, t,\ t' \in T_c$
(5.1) $conf(1,m)$ and *not* $conf(0,m)$ and *not* $conf(t,0)$
(5.2) $t' \leq t$ and $conf(t',m)$ imply $conf(t,m)$
(5.3) $conf(t',m)$ and $conf(t,m)$ imply $conf(t' \wedge t, m)$, thus $t' \wedge t > 0$
(5.4) $\forall t \in T_c \setminus \{0\}\ conf(t,*)$, and *not* $conf(0,*)$.

We now define an *S-graph* as a conceptual graph on a given support *S*. An S-graph represents a factual information, which is interpretable in the context of *S*.

<u>Definition.</u>

An **S-graph** related to a support *S*, is a multigraph $\boldsymbol{G=(R,C,U,lab)}$ where:

(1) *G* is a **bipartite, connected, finite** graph, $\boldsymbol{R}=\{r\text{-}vertices\}$ and $\boldsymbol{C}=\{c\text{-}vertices\}$ denote the two classes of relation and concept vertices, which we also call r-vertices and c-vertices, $\boldsymbol{U}$ is the set of edges, and $C \neq \emptyset$,

(2) this graph is partially ordered by *R:* for every $r \in R$, the set of edges adjacent to *r* is totally ordered, and we label these edges $i=1, \ldots, degree(r)$; we denote by $\boldsymbol{G(r)}$ the neighbour set of *r*; for every c-vertex *c*, we state $\boldsymbol{c = G_i(r)}$ if there exists an edge *rc* labelled *i*, and *c* is said to be the i-ith neighbour of *r*,

(3) every vertex has a **label** defined by a mapping *lab* which obeys to the following;
if $r \in R$ then $lab(r) = type(r) \in T_r$,
if $c \in C$ then $lab(c)=(type(c),ref(c))$, with $type(c) \in T_c$ and $ref(c) \in M \cup \{*\}$,
if $ref(c) \in M$, c is called an **individual** concept, and if $ref(c)=*$, c is called a **generic** concept,

(4) the degree in G of an r-vertex with type r is equal to the degree in B_r of the r-vertex (with type r), and the type of its i-ith neighbour is smaller or equal ($\leq$) to the type of the i-ith neighbour of the r-vertex in B_r,

(5) the conformity predicate, *conf*, satisfies:
$\forall c \in C, conf(type(c),ref(c))$.

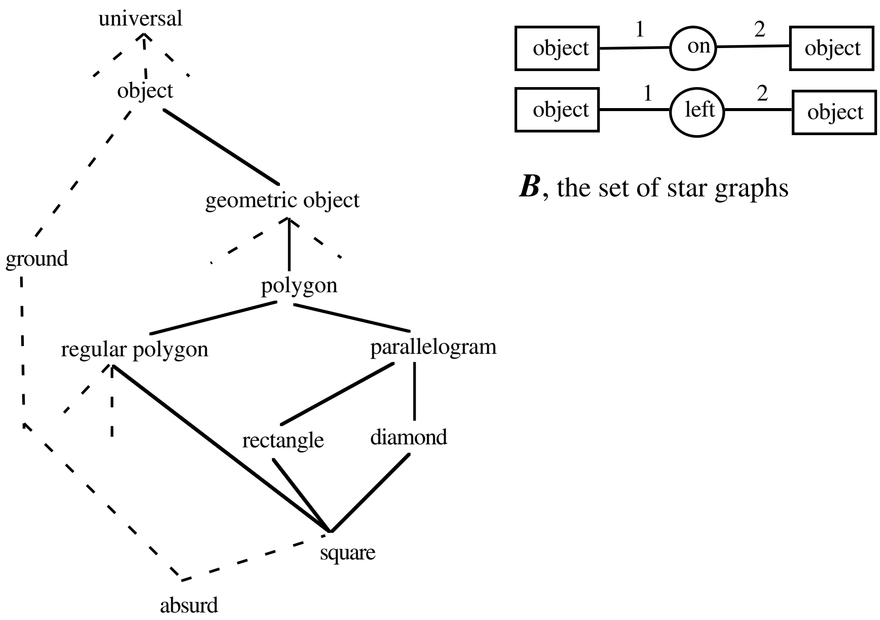
<u>*Remark.*</u> An isolated c-vertex is an S-graph.

Let $\boldsymbol{E_q}$ be the set of all possible labels for c-vertices, *i.e.* the set of elements (t,m) of $T_c x\{M \cup \{*,0\}\}$. E_q is a **lattice** whose order is defined as the product of the orders on T_c and on $\{M \cup \{*,0\}\}$.

The verification that a labelling fulfills the conformity relation *conf* - condition *5* concerning an S-graph - avoids some incoherences, it is an elementary form of reasoning. Furthermore, condition *5.3* applying to the conformity relation of a support *S* induces that, for any S-graphs set on *S*, the lower bound *t* of all types associated with the c-vertices having a same individual marker *m* is such that: $t > 0$ and $conf(t,m)$. In other words, if *m* is a t_1 and a t_2 then *m* is a $t_1 \wedge t_2$ that is not the absurd type.

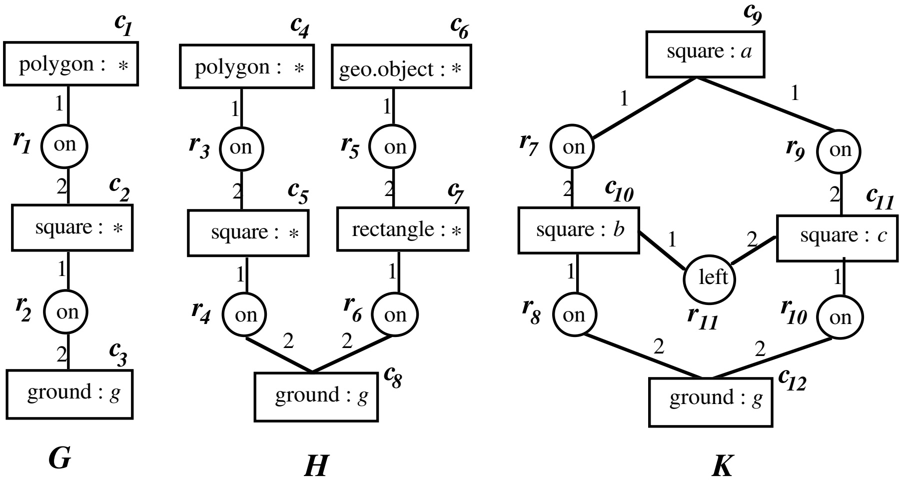
In the following, all S-graphs are implicitly related to a same support *S*. We will identify any star graph with the naturally associated S-graph: B_r is the S-graph having a unique r-vertex labelled *r*, whose neighbours are generic c-vertices. *B* is said to be the **basis** associated with *S*.

> We will study now the operations defined on S-graphs. We first introduce the notion of projection (section 3), which is an S-graph morphism. Then we will see less classical operations, Sowa's specialization rules (section 4), and give the correspondence between the two views (section 5).



Tc, the concept type lattice

Figure 2. *Partial definition of a support S*



A rectangle [t : *m*] represents a c-vertex with label *(t,m)*.

A circle (r) represents an r-vertex with label *(r)*.

Figure 3. *S-graphs G, H, K related to the support S*

3. Projections and S-graphs morphisms.

We define the "projection" and "S-graphs morphism" notions as morphisms of ordered bipartite graphs, which we provide with additional constraints on the vertices labellings.

<u>Definition.</u> *(See Figure 4)*

A **projection** from an S-graph $G = (R,C,U,lab)$ to an S-graph $G' = (R',C',U',lab')$ is an ordered pair (f,g) of mappings, f from R to R', and g from C to C', such that:

(i) $\forall r \in R$ and $\forall i \in \{1, \ldots , degree(r)\}$, $G_i(r) = c$ implies $g(c) = G'_i(f(r))$,[1]

(ii) $\forall r \in R \; lab'(f(r)) = lab(r)$,

(iii) $\forall c \in C \; lab'(g(c)) \leq lab(c)$.

Hence a projection is a morphism of the underlying graphs, that preserves the r-vertices labels and may restrict the c-vertices labels. To decrease the label of a c-vertex c is equivalently expressed as to restrict the type of c and / or, only if c is generic, to put an individual marker instead of its generic one, while satisfying the conformity relation *conf*.

E.g. for graphs H and K of Figure 3, $\Pi = \{(c_4,c_9), (c_6,c_9), (c_5,c_{10}), (c_7,c_{11}), (c_8,c_{12}), (r_3,r_7), (r_4,r_8), (r_5,r_9), (r_6,r_{10})\}$ is a projection from H to K.

<u>Definition.</u>

An S-graph **morphism** from an S-graph $G = (R,C,U,lab)$ to an S-graph $G' = (R',C',U',lab')$ is an ordered pair (f,g) of mappings fulfilling the same conditions as a projection, except for *(iii)* replaced by:

(iv) $\forall c \in C \; lab'(g(c)) = lab(c)$.

Hence a morphism is a particular projection.

E.g. for graphs G and H of Figure 3, $\Pi = \{(c_1,c_4), (c_2,c_5), (c_3,c_8), (r_1,r_3), (r_2,r_4)\}$ is an injective morphism from G to H.

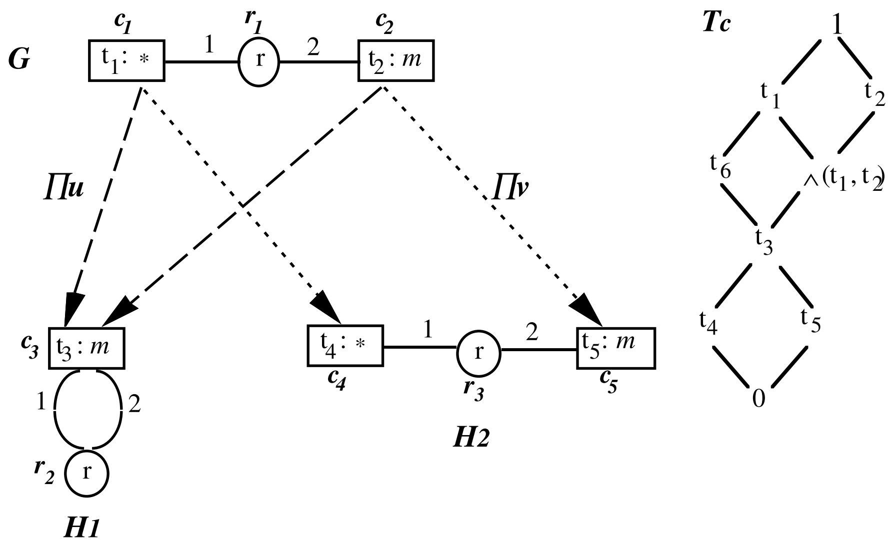
---

[1] In other words, for every edge rc labelled i in U, $f(r)g(c)$ is an edge labelled i in U'.

Definition.

An S-graph **isomorphism** (*resp.* **isoprojection**) is an S-graph morphism (*resp.* projection), say *(f,g)*, such that *f* and *g* are bijective.

Let us recall that condition *(4)* of definition of an S-graph implies that, given S-graphs *G* and *G'*, all r-vertices with same type have the same degree. So, if *f* and *g* are bijective, this suffices to ensure that the underlying graphs of *G* and *G'* are isomorphic (*i.e.* condition *(i)* of projection definition becomes: $\forall r \in R$ and $\forall i \in \{1, \dots, degree(r)\}$, $G_i(r) = c$ if and only if $g(c) = G'_i(f(r))$). Thus, an S-graph isomorphism is equivalently defined as an S-graph morphism whose reciprocal is also an S-graph morphism. This is false for an isoprojection, when it strictly restricts the label of at least one c-vertex.



$\Pi_u :$ $G \rightarrow H_1$, $r_1 \rightarrow r_2$, $c_1 \rightarrow c_3$, $c_2 \rightarrow c_3$

$\Pi_v :$ $G \rightarrow H_2$, $r_1 \rightarrow r_3$, $c_1 \rightarrow c_4$, $c_2 \rightarrow c_5$

Π_u is a non injective projection and Π_v is an isoprojection.

Figure 4. *Projections*

A composition of S-graph morphisms (*resp.* projections, S-graph isomorphisms, isoprojections) is an S-graph morphism (*resp.* projection, S-graph isomorphism, isoprojection).

Definition.

Let G be an S-graph; a **sub-S-graph** of G is a subgraph of G which is an S-graph when provided with the restriction of the G mapping *lab*.

Condition *(4)* applying to S-graphs implies that a strict partial subgraph of G is not a sub-S-graph, and more generally a subgraph of G that does not respect the arity of r-vertices is not a sub-S-graph; so we immediately have:

Property 1.

A sub-graph of an S-graph G is a sub-S-graph if and only if it is a connected component of an induced sub-graph of G obtained by deleting a set of r-vertices.

4. Specialization and generalization operations.

Let us now give Sowa's elementary *specialization* rules, and introduce the dual *generalization* rules.

Definition. ([Sowa 84], *3.4.3*)

The elementary specialization operations are internal operations on the set of S-graphs defined as follows:

(1) **deletion of twin r-vertices (simplify)**: if two r-vertices of an S-graph G with the same type are twin r-vertices, that is have exactly the same i-ith neighbours, then we build the S-graph G' obtained from G by deleting one of the duplicates,

(2) **elementary restriction (restrict)**: let c be a c-vertex of an S-graph G, then doing a restriction on c consists in building an S-graph G' obtained from G by replacing the label of c, say e, by e', such that: $e' \leq e$ and $conf(e')$,

(3) **elementary join**: let G, G' be two non necessarily distinct S-graphs, and c, c' be two c-vertices belonging respectively to G and G' and having the same label, then the result of the join of G and G' according to c and c' is the S-graph G'' obtained from G and G' by identifying these two vertices.

<u>*Definition.*</u>

G is said to be a **specialization** of H if H belongs to a specialization sequence ending with G.

Let us denote the specialization relation by $\leq$.

Given a set of S-graphs E, we note $\boldsymbol{\Sigma(E)}$ the closure of E under the specialization operations. An S-graph G of $\Sigma(E)$ is said to be **derived** from E.

Following operation *c-vertex fusion* is a combination of restrict and join operations.

<u>*Definition.*</u>

Given n c-vertices $c_1,\ldots,c_n$, $n\geq 1$ belonging to S-graphs $G_1, \ldots, G_k$, $k\leq n$, we say that they are **fusionnable** if $conf(e)$ holds, where e is the lower bound of their labels. A **fusion** of these n vertices consists in identifying them in a single vertex whose label, say e', satisfies: $e' \leq e$ and $conf(e')$. This operation is a composition of n restrictions followed by $n-1$ joins, thus is a specialization operation. If n=1, the fusion is an elementary restriction.

In Figure *4*, H_1 is obtained from G by doing a fusion of c_1 and c_2 (or by restricting the labels of c_1 and c_2, then by joining these vertices). To obtain H_2 from G, we need only to restrict the labels of c_1 and c_2. Note that H_1 and H_2 are incomparable for the specialization relation, in particular because c_4 and c_5 are not fusionnable vertices.

Condition *5.3* applying to the conformity predicate of a support ensures that in any graph we can fusion all c-vertices having the same individual marker. Hence, when the labelling mapping is not injective on the set of individual markers, we can use fusion to amount to an injective one.

Definition

The elementary **generalization** operations are the inverse of the elementary specialization operations:

(1) twin r-vertex **addition**,

(2) elementary **extension**, which replaces a c-vertex label by a greater label, such that it obeys to the type constraints issued from the basis B (*i.e.* for every edge rc with label i, $type(c) \leq$ type of the i-ith neighbour of the r-vertex in S-graph $B_{type(r)}$),

(3) elementary **split**, which duplicates a c-vertex, say c, into two vertices, say c_1 and c_2, with identical labels, and the set of edges adjacent to these new vertices is a bipartition - in a large meaning, one of the two parts may be empty - of the set of edges adjacent to c.

The latter operation produces several graphs if c is an articulation point or if one of the two parts is empty.

Note that dual restriction and extension operations are both limited by the support: increasing the type of a c-vertex is constrained by the basis B, whereas decreasing a label is constrained by the conformity relation *conf*.

Definition.

H is said to be a **generalization** of G if H can be obtained from G by a sequence of elementary generalization operations. Given a set E of S-graphs, we will be led to consider $\Gamma(E)$ the closure of E under the generalization operations.

We easily verify the following: if an elementary specialization (*resp.* generalization) operation yields G from H, then there is an elementary generalization (*resp.* specialization) operation which yields H from G. Then:

Property 2.

G is a specialization of H if and only if H is a generalization of G.

Remark.

The set of S-graphs definable on a support with basis B is exactly $\Sigma(B \cup \{[1,*]\})$, where $[1,*]$ is the S-graph reduced to a single generic c-vertex of universal type. In particular, for any S-graph set E, $\Gamma(E)$ and $\Sigma(E)$ are included in $\Sigma(B \cup \{[1,*]\})$.

> We have seen two basic operations, on one hand the projection and on the other hand specialization / generalization elementary rules producing the ≤ relation. We yield here a complete correspondence between these two notions.

5. Specialization and projection.

Lemma 1.

If there exists a projection from H to G, then $G \leq H$.

Proof.

Let us denote by $\Pi=(f,g)$ a projection from H to G.
For all c-vertex c of G, if $g^{-1}(c)=\{c_1,...,c_n\}$ then we can fusion the c-vertices c_i of H in a unique vertex whose label is that of c; if $n=1$, the fusion is simply a restriction. In doing that, we define a specialization sequence S_1 and a new S-graph H_1 obtained from H by S_1.
For all r-vertex r of G, if $f^{-1}(r)=\{r_1,...,r_m\}$ then the r_i are twin vertices of H_1. By deleting $m-1$ such vertices, we define a specialization sequence S_2 and a new S-graph H_2 obtained from H_1 by S_2.
Let $G'=\Pi(H)$ be the sub-S-graph of G image of H by the projection Π and isomorphic to H_2. Let us consider the graph $K=G \setminus G'$. Let us add to it the linking edges of G' in G, that is edges with one endpoint in G' and the other in G. The endpoint in G' of such edges is necessarily a c-vertex. Let K' be the obtained graph, then a connected component of K' is an S-graph. By successively joining the several connected components of K and K', we obtain G from G' *(see Figure 5)*. Let us denote by S_3 the specialization sequence it is associated with.
We go from H to G by the specialization sequence $S_1S_2S_3$. ❑

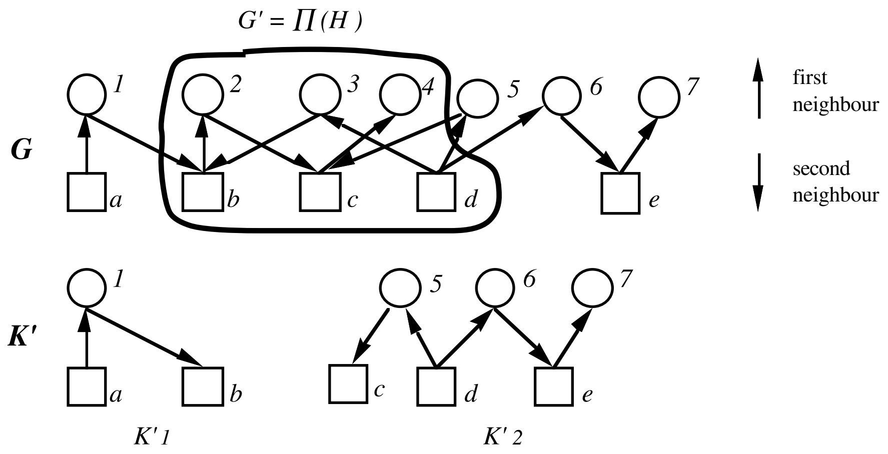


Figure 5. *Obtaining G from G' and K (in lemma 1 proof)*

<u>*Corollary:*</u> **If H is a sub-S-graph of G then $G \leq H$.**

<u>*Lemma 2.*</u>

If $G \leq H$ then with every specialization sequence from H to G we can associate a projection from H to G.

Proof. (See [Sowa 84], theorem *3.5.4*)

Let $H(=G_0)\ s_0\ G_1 \ldots s_{k-1}\ (G_k=)G$ be a sequence of elementary specializations. With every specialization s_i from G_i to G_{i+1} we can associate a projection from G_i to G_{i+1}. Indeed:
when we delete a twin r-vertex, we choose a projection unifying the two twin vertices,
when we do an elementary restriction, we take the projection induced by the decreasing of the c-vertex label,
and when we do an elementary join, we consider the projection induced by identifying the two joined c-vertices; let us note that, if the join applies to c-vertices of two disjoint S-graphs, then the projection is an injective morphism.
Finally, we obtain the wanted projection by composing all these elementary projections. ❑

The two previous lemmas yield:

<u>*Theorem 1.*</u>

$G \leq H$ if and only if there exists a projection from H to G.

Furthermore, we can associate with any specialization from H to G, and thus with any projection from H to G, a sequence in a particular form, that we call specialization in normal form; this specialization is made up of the three specializations defined in lemma 1: a first one contains only c-vertex fusions, and leads to an S-graph H_1, a second one consists exclusively of twin r-vertex deletions, and leads to an S-graph H_2, and a third one, the last, contains the joins from S-graphs pairwise disjoint and separated from H_2, yielding G.

If the projection from H to G is an injective morphism, then the specialization in normal form (associated with the projection) comprises solely the joins step. If it is surjective, then it does not contain the third joins step.

> We have seen ground operations on S-graphs. In next part, we complete these with definitions of r-vertex fusion, and compatible partitions, which are useful notions for studying the structure of the specialization relation (section 7) and matching operations (section 8).

6. Compatible partitions.

Let $G=(R,C,U,lab)$ denote an S-graph.

Definition.

A partition P_C of C is **compatible** if each of its classes is a set of fusionnable c-vertices - as defined in section 4 - (or **compatible set** of c-vertices).

Let $P_C = \{C_1, \ldots , C_n\}$ be a compatible partition of C, then we obtain a specialization G' of G by the surjective projection $\Pi=(f,g)$ defined as follows:

$f : R \rightarrow R'$ is the identity relation

$g : C \rightarrow C'=\{c_1, \ldots , c_n\}$ with if $c \in C_i$ then $g(c)= c_i$

$lab(c_i) = inf\{lab(c) / c \in C_i\}$

Figure *6* illustrates this kind of projection, with the partition $P_C=\{\{1,3,8\},\{6\}\}$, that gives the S-graph K from G.

Definition.

For any $r \in R$, let $\boldsymbol{P_r}$ be the partition of $\{1, \ldots , degree(r)\}$ induced by the equality of r neighbours, that is i and j are in the same class of P_r if and only if $G_i(r) = G_j(r)$.

Definition.

An r-vertex set, say A, is **compatible** if:

(i) $\forall r \in A,\ lab(r) = t$

(ii) Let P_A be the partition on $\{1, \dots, degree(t)\}$, upper bound of the partitions induced by the r-vertices of A: $P_A = \vee \{P_r / r \in A\}$.
Then for every class I of P_A, the c-vertex set $C_I = \{G_i(r) / i \in I \text{ and } r \in A\}$ is compatible.

The partition P_A induces a partition on C, called $\boldsymbol{PC(A)}$, with the following classes: the sets C_I and the singletons containing the c-vertices that do not belong to $\cup\{G(r) / r \in A\}$.

In Figure *6*, let us consider $A = \{4,7\}$, a set of G r-vertices with identical labels. The partitions induced by the neighbours equality are respectively $P_4=\{\{1,2\}\}$ and $P_7=\{\{1\},\{2\}\}$. Thus $P_A=\{\{1,2\}\}$ and it induces on the c-vertex set of G the partition $PC(A)=\{\{3,6,8\},\{1\}\}$, where each subset is compatible.

Definition.

The r-vertices of a compatible set, say A, are also said to be fusionnable: the **fusion** of these vertices consists in fusionning the c-vertices of each class of $PC(A)$ into one vertex whose label is the lower bound of the fusionned vertices labels, and in keeping only one representative of A (*i.e.* deleting $|A|-1$ twins of A).

Definition.

A partition P_R of R is **compatible** if:

(i) $\forall R_i \in P_R$, R_i is a compatible r-vertex set

(ii) Let $PC(P_R)$ be the partition on C induced by P_R, that is the upper bound of

the partitions $PC(R_i)$ induced by each class R_i of P_R; then each class of $PC(P_R)$ is a compatible c-vertex set.

We obtain a specialization G' of G by doing a fusion of the r-vertices according to $P_R = \{R_1, \dots, R_n\}$, that is by the following surjective projection $\Pi=(f,g)$:

$f : R \rightarrow R'=\{r_1, \dots, r_n\}$ with if $r \in R_i$ then $f(r)=r_i$ and $lab(r_i) = lab(r \in R_i)$

$g : C \rightarrow C'=\{c_1, \dots, c_p\}$ with $PC(P_R) = \{C_1, \dots, C_p\}$
if $c \in C_i$ then $g(c)=c_i$
and $lab(c_i) = inf\{lab(c) \mid c \in C_i\}$

Figure *6* illustrates this kind of projection, with $P_R = \{\{2,5\},\{4,7\}\}$, that gives the S-graph *H* from *G*.

Definition.

A partition $\boldsymbol{P = P_R + P_C}$ on the vertex set of an S-graph *G* is **compatible** if
(i) P_R is a compatible partition of *R*
(ii) P_C is a compatible partition of *C*, less thin than $PC(P_R)$.

Definition.

Let *G* be an S-graph and $P = P_R + P_C$ be a compatible partition of its vertex set. We provide the underlying quotient graph *G/P* with an S-graph structure by labelling every class of *R* with the label of one of its elements and by labelling every class of *C* with the lower bound of its element labels.

This graph is called the **quotient S-graph of *G* by *P***, and denoted by ***G/P***.

With a compatible partition *P* of *G* is naturally associated a surjective projection from *G* to *G/P*. Conversely, if there is a projection Π from *G* to *H*, then $P = P_R + P_C$, the partition induced by the equality of the images by Π, is a compatible partition. Indeed, P_R and P_C are compatible partitions by definition of a projection and P_C is less thin than $PC(P_R)$ by definition of the underlying graphs morphism. Furthermore, there exists an isoprojection from *G/P* to $\Pi(G)$: Π defines an isomorphism between the underlying graph of *G/P* and the one of $\Pi(G)$; and given any class $\{c_1,...,c_p\}$ of P_C, let *y* be the c-vertex of *H* such that for $i=1,...,p$, $\Pi(ci) = y$, then $\wedge(lab(ci)) \geq lab(y)$.

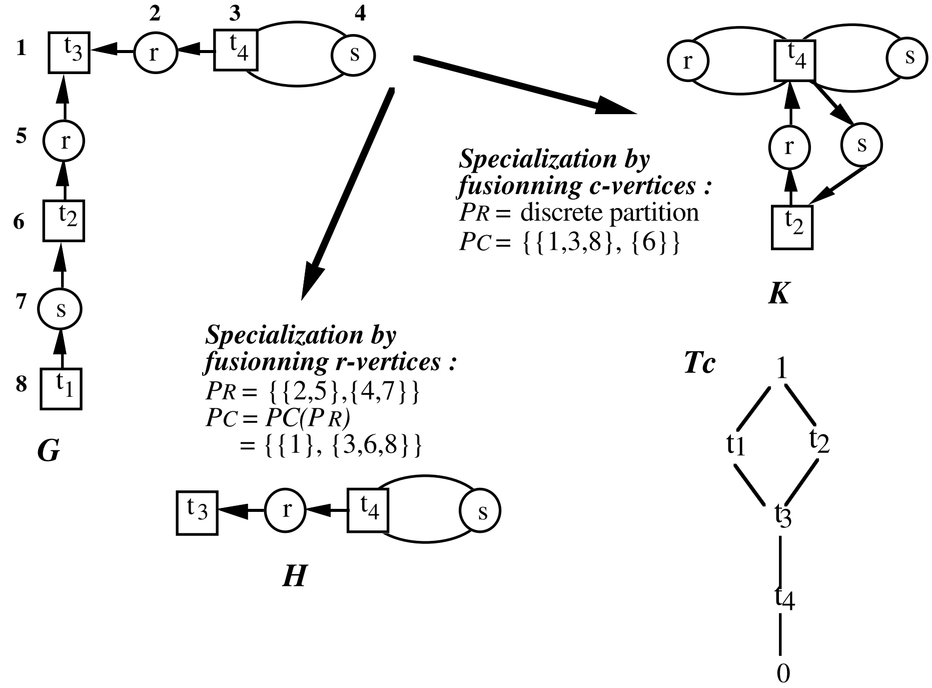


Figure 6. *Compatible partitions*

We now study the structure of the specialization relation ≤. We specify here that ≤ is only a preorder and not an order as often proposed (it is not an antisymmetric relation). This is a true distinction, because we will show in last part that the problem of determining whether two S-graphs are equivalent is NP-complete. In order to characterize exactly the equivalence relation associated with ≤, we introduce the notion of irredundant S-graphs.

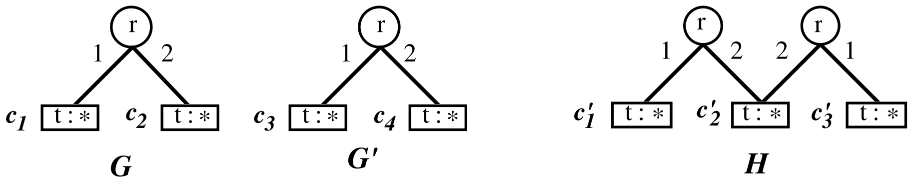
7. Equivalence.

The concatenation of two specialization sequences is a specialization sequence, so the specialization relation is a transitive relation. It is also a reflexive relation, if we consider that the empty sequence is a specialization sequence.

<u>*Property 3.*</u>

The specialization relation $\leq$ is a preorder.

It is not an order because, as shown in Figure *7*, where $G \leq H$ and $H \leq G$, the antisymmetry property is not fulfilled:



H is obtained from *G* by joining *G* and *G'* according to c_2 and c_4.
G is obtained from *H* by joining c'_1 and c'_3 then by deleting a twin r-vertex.

Figure 7. *$H \leq G$, $G \leq H$ and H is not isomorphic to G*

The purpose of this paragraph is to study the equivalence relation associated with the preorder.

If $G \leq H$ and $H \leq G$ then *H* and *G* are said to be equivalent, denoted by $G \equiv H$.
If *G* and *H* are equivalent, and $G=G_1,G_2,...,G_n=H$ is a specialization sequence, then all G_i are equivalent.
If $G \leq H$, and *H* is not equivalent to *G*, then *G* is said to be strictly more specific than *H*, denoted $G<H$. The $<$ relation is antisymmetric, transitive and absorbs the $\leq$ relation (*i.e.* $G<H$ and $G' \leq G \Rightarrow G'<H$, $G<H$ and $H \leq H' \Rightarrow G<H'$).

<u>*Definition.*</u>

A graph *G* is said to be **irredundant** if it has no strict sub-S-graph equivalent to it. Thus *G* is irredundant if and only if there does not exist a projection from it to one of its strict sub-S-graphs.

In Figure *6*, *G* and *H* are equivalent, *K* is strictly more specific than *G*, and *H* is irredundant but *K* is not. For another example, see Figure *3*, where *G* and *K* are irredundant and $K < H \equiv G$.

Any irredundant graph G has no twin r-vertices, since by deleting such a vertex we obtain a strict sub-S-graph of G which is equivalent to G.

Theorem 2.

For an S-graph *G*, the following properties are equivalent:

(1) *G* is irredundant,

(2) *G* has no twin vertices and every S-graph *G'* obtained from *G* by fusionning two c-vertices is strictly more specific than *G*,

(3) Every S-graph obtained from *G* by fusionning two r-vertices is strictly more specific than *G*.

Proof.

not (2) implies not (1):
Let G' be a graph obtained from G by fusionning two c-vertices and G' equivalent to G, then there exist a non injective projection from G to G' and a projection from G' to G by the equivalence definition, thus, by composing these two projections, we obtain a projection from G to one of its strict sub-S-graphs.

(2) implies (3):
Let G' be a graph obtained from G by fusionning two r-vertices. By hypothesis (2), G has no twin vertices, so this r-vertex fusion involves at least one c-vertex fusion. Let G'' be the graph obtained from G by one of these c-vertex fusions. Then $G' \leq G''$ and $G'' < G$ (by *(2)*), thus $G' < G$ (absorption property of $<$).

not (1) implies not (3):
Let us assume that there exists one sub-S-graph of G, say G', equivalent to G. Then, there is a projection Π from G to G', and Π is not injective on R, by definition of a strict sub-S-graph. Let a and b be some r-vertices of G with $\Pi(a) = \Pi(b)$. The graph G'' obtained from G by fusionning a and b is a specialization of G; now $G \leq G'$ and $G' \leq G''$, then, by transitivity, $G \leq G''$; thus $G'' \equiv G$. ❑

Property 4.

If some S-graph *G* is equivalent to some irredundant S-graph *H*, then *G* has a sub-S-graph isomorphic to *H*.

Proof.

H and G are equivalent, so there is a projection Π from H to G, and there is a projection Π' from $\Pi(H)$ to H.

We have $\Pi'(\Pi(H))=H$, otherwise there would exist a projection from H to one of its strict sub-S-graphs, and H would not be irredundant. Thus $\Pi(H)$ and H are isomorphic S-graphs. ❑

Theorem 3.

An equivalence class contains one and only one irredundant S-graph.

Proof.

Property *4* implies that two irredundant S-graphs are equivalent if and only if they are isomorphic. So an equivalence class contains at most one irredundant graph.
Given an equivalence class, let us consider some graph, say G, with a minimal number of vertices. Such a graph is irredundant, otherwise it would possess some strict sub-S-graph equivalent to it, and this would contradict the hypothesis on G. ❑

Thus, each equivalence class is provided with a unique representative. Furthermore, this representative is the unique S-graph having the smallest vertex number.

> Next part is devoted to definitions of the extended join operation, which is a form of pattern matching between S-graphs. It is the principal computation tool introduced by Sowa, because all more sophisticated operations are based on it. We introduce two forms: the extended join - the most general operation - and the isojoin - a restricted form which is the one usually used.

8. Extended join and isojoin.

Definition.

Two S-graphs G_1 and G_2 are **fusionnable** if there are partitions P_1 and P_2 respectively compatible on G_1 and G_2, such that there exists an isomorphism γ of the G_1/P_1 underlying graph onto the G_2/P_2 underlying graph, and γ fulfills: for every vertex x of G_1/P_1, $\{x, \gamma(x)\}$ is compatible.

The fusion of G_1 and G_2 consists in fusionning two by two the vertices of G_1/P_1 and G_2/P_2 matched by γ; the labels of the new c-vertices are the lower bound of the fusionned vertices labels.

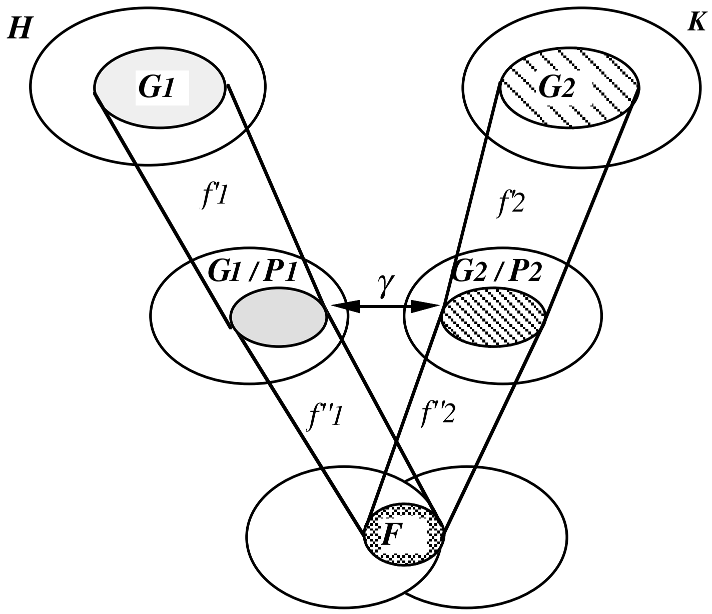
Let us denote by F the obtained S-graph. Equivalently, F is the S-graph constructed from G_1/P_1 in replacing the label of each c-vertex c by the lower bound of the c and $\gamma(c)$ labels; then G_1/P_1 and G_2/P_2 are more general or equal (≥) to F, and we have:

Property 6.

G_1 and G_2 are fusionnable if and only if there exist an S-graph F and surjective projections f_1: $G_1 \rightarrow F$ and f_2: $G_2 \rightarrow F$.

Definition.

An **extended join** operation between two distinct S-graphs *H* and *K* consists in fusionning two of their fusionnable sub-S-graphs *(see Figure 8).*



$f_1 = f''_1 \bullet f'_1$
$f_2 = f''_2 \bullet f'_2$

f''_1 and f''_2 are isoprojections. f'_1 and f'_2 are in general case surjective projections, and the identity (or isomorphisms) if *G1* and *G2* are isofusionnable (see below).

Figure 8. *Extended join between H and K*

Property 7.

G is derivable from an S-graph set E if and only if it is derivable from E by the extended join operation.

Proof.

⇐ Extended join is a composition of elementary specialization operations.

⇒ Reciprocally, we can express every elementary specialization operation with an extended join. An elementary join applying to distinct S-graphs is a particular case of extended join. To do a join between two c-vertices x and y of a same graph G consists in fusionning G and an other graph G' identical to G, by fusionning each vertex with its homologue, and also x with y (the fusion keeps the c-vertex labels). Deleting a twin r-vertex consists in fusionning G with a G' identical to G, in fusionning each vertex with its homologue, and in identifying the four r-vertices involved (the fusion keeps the c-vertex labels). To do a restriction on a c-vertex c of G, replacing its label by e', consists in fusionning G and G' identical to G, by fusionning each vertex with its homologue. This fusion keeps the c-vertex labels, except for the label of c replaced by e'. ❑

The composition of extended joins is generally not an extended join, due to the connectivity constraint on the fusionned subgraphs.

<u>*Definition.*</u>

Let $G = (R,C,U,lab)$ and $G' = (R',C',U',lab')$ be two S-graphs, α a bijection from A / R to A' / R', and β a bijection from B / C to B' / C'. The bijections pair $\gamma=(\alpha,\beta)$ is said to be **compatible** if:

(1) ¢*c*[*B*, *c* and $\beta(c)$ are fusionnable,

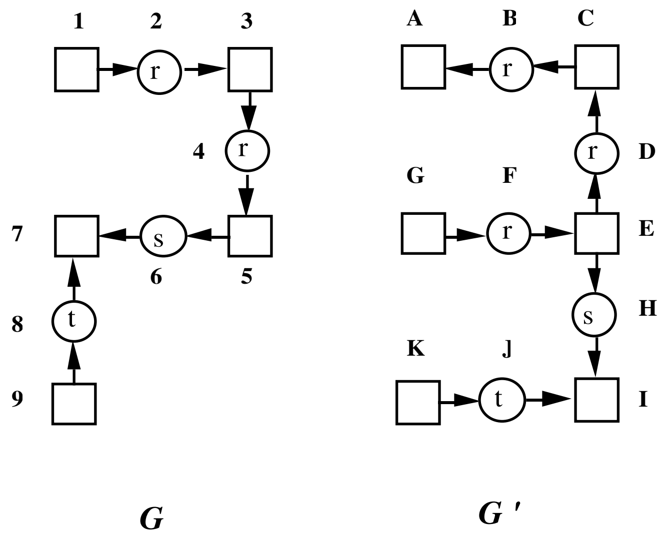
(2) ¢*r*[*A*, $G(r) / B$ and
let $r'=\alpha(r)$, then $lab(r)=lab'(r')$ and ¢*i:1..degree(r)*, $\beta(G_i(r))=G'_i(r')$

(3) *G*[*A"B*] and *G'*[*A'"B'*] are connected graphs (*i.e.* are S-graphs).

In this case, *G*[*A"B*] and *G'*[*A'"B'*] are fusionnable (sub)S-graphs with discrete partitions (*i.e.* γ is an isomorphism of their underlying graphs): they are said to be **isofusionnable**. The associated fusion is called **isojoin** between G and G' according to γ *(see Figures 8 and 9).*

The decisional problem associated with the search of an isojoin between G and G', such that the cardinal of *A"B* is maximal, is an NP-complete problem. It is still NP-complete when we restrict it to the search of an S-graph maximal for inclusion[1].

[1] These problems contain for instance the NP-complete problem "clique" [Garey Johnson 79, GT19]. See section 9 for transformations of unlabelled graphs to S-graphs.



We assume that all c-vertices are pairwise fusionnable.

γ maximal for inclusion:

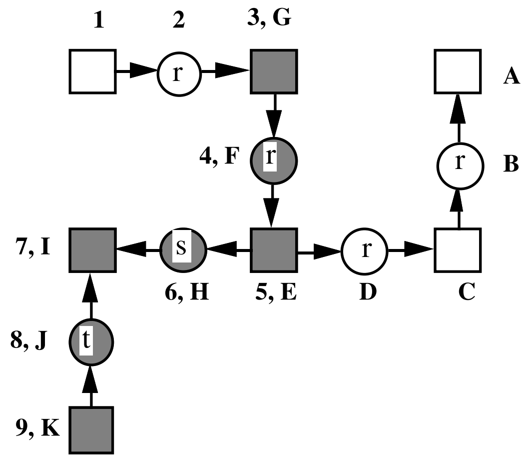
γ_1: $\alpha_1 = \{(2,B)\}$ $\beta_1 = \{(1,C),(3,A)\}$

$A \cup B$ maximal for inclusion:

γ_2: $\alpha_2 = \{(2,F),(4,D)\}$ $\beta_2 = \{(1,G),(3,E), (5,C)\}$

$A \cup B$ or γ with maximal cardinal:

γ_3: $\alpha_3 = \{(4,F),(6,H),(8,J)\}$ $\beta_3 = \{(3,G),(5,E), (7,I),(9,K)\}$



Graph obtained by an isojoin of G and G' according to γ_3

Figure 9. *Maximal isojoins between G and G'*

So we will often replace it by the search of an **isojoin with γ maximal for inclusion** (*i.e.* such that there is no extension of γ that conserves the connectivity of the fusionned subgraphs). Furthermore this last notion seems to be the one used by Sowa, starting moreover with two given c-vertices c and c' which respectively belong to C and C' and form a pair for β. It is immediate to build polynomial time algorithms computing a maximal isojoin - for the inclusion on the associated bijections pair - proceeding by depth-first, breadth-first or mixed search, starting with c and c'.

Figure *9* illustrates the different meanings of a "*maximal* isojoin".

If we restrict extended join to isojoin, then we can no more ensure property *7*: indeed, such an operation can not decrease the number of c-vertices of a graph, so it can not simulate an elementary join between two c-vertices of a same graph.

> Up to now, we have been focusing on graph operations: projection, elementary specialization / generalization rules, c-vertex and r-vertex fusions, extended join. The purpose of the next part is to study logical interpretation of S-graphs and graph operations. We will show that graph generalization rules constitute a complete set of inference rules for the set of logical formulas associated with S-graphs.

9. Logical interpretation.

9.1. Logical formula associated with an S-graph.

Sowa proposes, in *3.3.2* of [Sowa 84], to associate with every conceptual graph $\boldsymbol{G}$ a well formed formula of the first order predicate calculus, say $\boldsymbol{f(G)}$, by the following construction:

- with each generic c-vertex c_i, associate a distinct variable x_i, (there is a bijection between the generic c-vertices and the variables)
- with each individual marker m, associate a constant - that we also call m -, (there is a bijection between the individual markers and the constants, but not necessarily one between the individual c-vertices and the constants)
- the predicate set is in bijection with the set $\boldsymbol{T_c \cup T_r}$ of all types,
- given any c-vertex c of C, let *ident(c)* denote the variable or the constant associated with c, then:
  - with every c of C, associate the atom *type(c) (ident(c))*,
  - with every r of R, associate the atom *type(r) (ident(c$_1$), ..., ident(c$_n$))*, where n is the arity of r and c_i is the i-ith neighbour of r in G,

finally, let ***p(G)*** be the formula consisting of the conjunction of all atoms associated with the vertices, then ***f(G)*** is the existential closure of *p(G)*.

E.g. with the graph *H* of Figure *3*, is associated the formula $\exists x\ \exists y\ \exists z\ \exists t\ polygon(x) \wedge square(y) \wedge geo.object(z) \wedge rectangle(t) \wedge on(x,y) \wedge on(y,g) \wedge on(z,t) \wedge on(t,g)$.

Moreover, we assume that: for all *t, t'* concept types of T_C, **if** $t' \leq t$ **then** $\forall x\ t'(x) \rightarrow t(x)$.[1]

<u>Definition.</u>

Given a support *S*, we call **S-formula** any well formed formula in the form *f(G)* where *G* is an S-graph on *S*. The set of S-formulas is exactly $f(\Sigma(B \cup \{[1,*]\}))$, using notations of part 4.

9.2. Logical interpretation of specialization / generalization operations

Let us assume that we go from *G* to *H* with an elementary specialization rule (reciprocally from *H* to *G* with an elementary generalization rule). Let *f(G)* and *f(H)* be the logical formulas respectively associated with *G* and *H*, equivalently denoted by *EC(p(G))* and *EC(p(H))*, where ***EC*** denotes the existential closure.

(1) Twin r-vertices deleting / addition.

Let $r(t_1, \dots, t_k)$ be the atom associated with the deleted r-vertex of *G* yielding *H*, where *r* is a predicate and the t_i are terms (variables or constants). We go from *f(G)* to *f(H)* by deleting an occurrence of $r(t_1, \dots, t_k)$ in *f(G)*. From the logical equivalence $r(t_1, \dots, t_k) \wedge r(t_1, \dots, t_k) \equiv r(t_1, \dots, t_k)$, we directly deduce that $f(G) \equiv f(H)$.

(2) Restriction / extension of the label of a c-vertex c.

First case: marker restriction / extension.

We replace the label $(t,*)$ with (t,a), where *a* is an individual marker. Let *x* be the variable and *a* be the constant associated with the c-vertex *c*, respectively in *f(G)* and

[1] This classical hypothesis allows us not introducing, in this paper, Sowa's denotation operator *d* (3.2.2 of [Sowa 84]).

$f(H)$. $f(G) = EC(F[x] \wedge A)$ and $f(H) = EC(F[a] \wedge A)$, with $F[x] = (\exists x\ t(x) \wedge r_1(..., x, ...) \wedge .. \wedge r_k(..., x, ...))$, where $F[a]$ results from the substitution of a for x in $F[x]$, and A is a conjunction of atoms not containing x (a may appear as a constant associated with other individual c-vertices). We go from $f(G)$ to $f(H)$ by substituting a for x, so $(f(H) \rightarrow f(G))$ is a valid formula.

Second case: type restriction / extension.

$f(G)$ is in the form $EC(t(x) \wedge A)$ or $EC(t(a) \wedge A)$, where we replace the predicate t by the predicate t', with: $\forall x(t'(x) \rightarrow t(x))$. Thus $(f(H) \rightarrow f(G))$ is a valid formula. When we restrict (reciprocally extend) both type and marker, we compose the above two operations, so we preserve the validity of $(f(H) \rightarrow f(G))$.

(3) Join of two vertices / split of a c-vertex.

If H is obtained by joining two c-vertices of G, then the reciprocal split operation keeps H connected (following cases *1.1* and *2.1*). But, if H is obtained by joining a c-vertex of G and a c-vertex of another graph into a c-vertex c, then c is an articulation point of H, and the reciprocal split operation decomposes H into two connected components (following cases *1.2* and *2.2*).

First case: individual c-vertices.

(1.1) If we join two individual c-vertices of G (reciprocally split an individual c-vertex of H) then we do not change the associated logical formula. So, $f(G) = f(H)$.

(1.2) If we join an individual c-vertex of G with an individual c-vertex of G', $G \neq G'$, then $f(H) = EC(p(G) \wedge p(G'))$, where we assume that the variable sets of $p(G)$ and $p(G')$ are disjoint sets. So, $f(H) \equiv (f(G) \wedge f(G'))$.

Second case: generic c-vertices.

(2.1) Let c_1 and c_2 be generic c-vertices of G, joined into the generic c-vertex c of H, and x_1, x_2 and x be the respectively associated variables.
$f(G) = EC(F[x_1] \wedge F'[x_2] \wedge A)$ where: $F[x_1] = (t(x_1) \wedge r_1(..., x_1, ...) \wedge ... \wedge r_i(..., x_1, ...))$ and $F'[x_2] = (t(x_2) \wedge r_{i+1}(..., x_2, ...) \wedge ... \wedge r_j(..., x_2, ...))$ where x_2 does not appear in $F[x_1]$, x_1 may appear in an atom of $F'[x_2]$ containing x_2, and x_1 and x_2 do not appear in A.
$f(H) = EC(F[x_1] \wedge r_{i+1}(..., x_1, ...) \wedge ... \wedge r_j(..., x_1, ...) \wedge A)$, where A does not contain occurrences of x_1, thus $f(H)$ is equivalent to the formula obtained in substituting x_1 for x_2 in $f(G)$. So $(f(H) \rightarrow f(G))$ is a valid formula.

(2.2) When we join c_1 of G with c_2 of G', we have $f(G) = EC(F[x_1] \wedge A)$, $f(G') = EC(F'[x_2] \wedge A')$, where A and A' do not contain occurrences respectively of x_1 and x_2, and the variables sets of $f(G)$ and $f(G')$ are disjoint.
$f(H) = EC(F[x_1] \wedge r_{i+1}(..., x_1, ...) \wedge ... \wedge r_j(..., x_1, ...) \wedge A \wedge A')$. Thus $(f(H) \rightarrow f(G) \wedge f(G'))$ is valid.

The interpretation of the join operation is simple in a theory with equality; indeed, identifying two variables x and y appearing in a same formula, or in two formulas with disjoint variable sets, is equivalent to adding the atom *equal(x,y)*.

Theorem 4.

If $H \leq G$ then $f(H) \rightarrow f(G)$

Proof. (See [Sowa 84], theorem *3.5.3.*)

Let $G\ (=H_1)\ s_1\ H_2\ s_2\ ...\ s_{n-1}\ (H_n=)\ H$, be a sequence of elementary specializations from G to H. For every i, $i = 1, ..., n-1$, we have $f(H_{i+1}) \rightarrow f(Hi)$, due to the above given logical interpretation of specialization rules, thus, by composition, $f(H) \rightarrow f(G)$. ❑

Finally, generalization operations correspond to inference rules, and conversely, the rules associated with the specialization operations are refutation rules - *i.e.* if a formula, say f, can be derived from a set of false formulas by a specialization sequence, then f is false.

9.3. Completeness of generalization operations.

Definition.

Let f and g be two S-formulas. An **S-substitution** from g to f is a mapping σ which, to every term or atom of g, associates a term or atom of f, in the following way:

if a is a constant of g: $\sigma(a) = a$ (so a must be a constant of f),
if x is a variable of g: $\sigma(x)$ is a variable or a constant of f,
if t is the unary predicate associated with the concept type t: $\sigma(t) = t'$ where $t' \leq t$,
if r is the predicate associated with the relation type r: $\sigma(r) = r$,
In addition, the image by σ of every atom $v(x_1, ..., x_k)$ of g is an atom $\sigma(v)(\sigma(x_1),...,\sigma(x_k))$ of f.

In other words, $\sigma(g)$ is the S-formula obtained by substituting for every term or predicate in g its image in f, and then deleting the existential quantifiers that were applying to variables of g turned into constants of f. We immediately have the following property:

Property 8.

There is a projection from G to H if and only if there is an S-substitution from $f(G)$ to $f(H)$.

Definition.

Two S-formulas are **separated** if they do not share any variable or constant. As S-formulas are existentially quantified, the important point is that the two constant sets are disjoint.

We will note A the set of clauses associated with the concept type lattice, T_c, that is the set: $\{T_{ij}= \neg t_i(x) \vee t_j(x) / t_j$ covers t_i in $T_c\}$.
The closure of A under the resolution rule is the set of clauses $A' = \{T_{ij}= \neg t_i(x) \vee t_j(x) / t_i < t_j$ in $T_c\}$.

Lemma 3.

Let f and g be two S-formulas. If g is logically deducible from A and f then there exists an S-substitution from g to f.

Proof.

Let us write: $f = \exists x_1...\exists x_p(t_{v1} \wedge ... \wedge t_{vm} \wedge r_{w1} \wedge ... \wedge r_{wn})$, $p \leq m$, and
$g = \exists y_1...\exists y_q(t_{i1} \wedge ... \wedge t_{ih} \wedge r_{j1} \wedge ... \wedge r_{jk})$, $q \leq h$,
where the t_v and t_i (*resp.* r_w and r_j) are atoms associated with c-vertex types (*resp.* r-vertex types).
"g is logically deducible from f and A" is equivalent to "$f \wedge A \wedge \neg g$ is an inconsistent formula". The clausal form C of the formula $f \wedge A \wedge \neg g$ contains the following clauses :

the clause associated with $\neg g$: $\neg t_{i1} \vee ... \vee \neg t_{ih} \vee \neg r_{j1} \vee ... \vee \neg r_{jk}$ that contains the variables $y_1...y_q$,

the set C_f of clauses associated with f, where each clause consists of a single atom and contains no more variables, since the variables $x_1...x_p$ have been changed into Skolem constants $a_1...a_p$,

and the clauses of A.

Since the formula $f \wedge A \wedge \neg g$ is inconsistent, it is possible to obtain the empty clause by using the resolution rule. The last time this rule is used in a sequence ending with the empty clause, it applies to: either a t_{ij} of $\neg g$ and, either a t_{va} of C_f or a t_s obtained by resolution between A' and a t_{va} of C_f, or a r_{js} of $\neg g$ and a r_{wb} of C_f. So we have to empty $\neg g$ in order to obtain the empty clause, and we can assume that it is done following the order $t_{i1},...,t_{ih}$, $r_{j1},...,r_{jk}$. If we note σ_l the substitution that deletes $\neg t_{il}$, σ_{h+l} the one that deletes $\neg r_{jl}$, the composition of these substitutions defines a substitution from g to f (this composition is actually only a union). ❑

Lemma 4.

Let $\{f_1,\ldots,f_k\}$ be a set of pairwise separated S-formulas.
For every S-formula f we have:
if f is logically deducible from $A \cup \{f_1,\ldots,f_k\}$ then there is a formula f_i, $1 \le i \le k$, such that f is a logical consequence of A and f_i.

Proof.

Let us go back to the proof of lemma *3*, and, consider the set of clauses associated with the formulas $f_1,\ldots,f_k$, instead of f solely. And let us consider a sequence of applications of the resolution rule emptying $\neg g$.
The first clause occurring in a resolution with $\neg g$ comes from one of the formulas $f_1,\ldots,f_k$, for example f_1, or is obtained by resolution between a clause of f_1 and a clause of A'. Now g is "connected", that is, if we consider any two terms u and v of g, there is a sequence $u = w_1\ A_1\ w_2 \ldots A_n\ w_{n+1} = v$, where, for $1 \le i \le n$, A_i is an atom of g, and w_i, w_{i+1} are some terms occurring in A_i. In addition, the clauses associated with f_i, $1 \le i \le k$, are without variables and have constant sets pairwise disjoint (condition of separability and skolemization). So, to empty $\neg g$, we can only use the clauses associated with f_1 and those associated with A. ❑

We can now establish the following theorem:

Theorem 5 (of completeness).

Let E be a set of S-graphs, and $\{f_1,\ldots,f_k\}$ be a subset of $f(E)$ consisting of pairwise separated S-formulas.
For every S-formula f we have: if f is logically deducible from $A \cup \{f_1,\ldots,f_k\}$ then $f \in f(\Gamma(E))$.

Proof.

If f is logically deducible from A and $\{f_1,\ldots,f_k\}$, pairwise separated, then, with lemma *4*, there is a formula f_i, $1 \le i \le k$, such that f is logically deducible from A and f_i. Thus, with lemma *3*, there is an S-substitution from f to f_i, and we conclude with property *8*. ❑

Theorems *4* and *5* yield that the set of generalization operations on S-graphs corresponds to a sound and complete set of logical rules on separated S-formulas.

We can not relax the separation constraint on formulas, as shown in the following example:

$$G_1 = [\ t : a\ ] \rightarrow (\ P\ ) \rightarrow [\ t' : *\ ]$$
$$G_2 = [\ t'' : *\ ] \rightarrow (\ P\ ) \rightarrow [\ t : a\ ]$$
$$g_1 = \exists x\ t(a) \wedge P(a,x) \wedge t'(x)$$
$$g_2 = \exists y\ t''(y) \wedge P(y,a) \wedge t(a)$$
$$G = [\ t'' : *\ ] \rightarrow (\ P\ ) \rightarrow [\ t : a\ ] \rightarrow (\ P\ ) \rightarrow [\ t' : *\ ]$$
$$g = \exists x\ \exists y\ t''(x) \wedge P(x,a) \wedge t(a) \wedge P(a,y) \wedge t'(y)$$

g can be deduced from g_1 and g_2, which are non separated formulas, but G can not be obtained from G_1 and G_2 by the generalization rules (on the other hand, it is possible with the specialization rules).

> We have given a framework consisting of S-graphs related to a support, and associated graphs operations, provided with a logical interpretation. Last but not least is now to study algorithmic complexity of its handling. Next part is devoted to this study. In a simplified manner, encountered problems are generally NP-complete and polynomial on instances involving S-graphs whose underlying graphs are trees.

10. Algorithmic complexity of basic operations.

Underlying graphs of S-graphs are bipartite graphs with any structure: therefore, when a graph problem, in its decisional form, is NP-complete, then its transposition in an S-graph problem is NP-complete too.[1]

Any directed graph G can be seen as a particular case of S-graph: the vertices of G are c-vertices having all the same label, and every edge (x,y) represents a binary r-vertex whose x is the first neighbour and y the second one, the r-vertices having all the same label.[2]

This transformation yields a simple proof for NP-completeness of the two following problems:

Problem 1: *Projection (comparison by the specialization / generalization relation)*

Instance	:	Two S-graphs A and B.
Question	:	Is there a projection from A to B ?

Property 9.

The "projection" problem is NP-complete.

Proof.

Let us consider the following problem "directed graphs morphism": given two directed graphs $G_1 = (X_1,E_1)$, $G_2 = (X_2,E_2)$, is there a mapping from X_1 to X_2 satisfying: for all x,y of X_1, if (x,y) belongs to E_1 then $(\Pi(x),\Pi(y))$ belongs to E_2? This problem is NP-complete since it contains as a particular case the NP-complete problem "clique" [Garey Johnson 79, GT19].

Transforming G_1 and G_2 into the S-graphs G'_1 and G'_2 by the above mentioned transformation, we have: there is a morphism from G_1 to G_2 if and only if there is a projection (here it is an S-graphs morphism) from G'_1 to G'_2. ❑

Problem 2: *"Sub-S-graph"*, [*injective projection*]

Instance	:	Two S-graphs A and B.
Question	:	Is A isomorphic to a sub-S-graph of B?

Property 10.

The "sub-S-graph" ["injective projection"] problem is NP-complete.

[1] All problems studied here are clearly in NP. In next NP-completeness proofs, we will omit to precise it again.

[2] This holds for non-directed graphs since they can be seen as a particular case of directed graphs: each non directed edge xy corresponds to symetric directed edges (x,y) and (y,x).

Proof.

It admits as a particular case the NP-complete problem "isomorphic subgraph" [Garey Johnson 79, GT48]. ❑

In the two previous problems, the labels have no part in NP-completeness, and, on the other hand, may simplify the problem in practical applications. But it is not the same for the following problems "isoprojection" and "isofusionnability".

Problem 3: *Isoprojection*

Instance	:	Two S-graphs G_1 and G_2.
Question	:	Is there an isoprojection from G_1 to G_2 ?

If we add the constraint that a vertex and its image have the same label, then the problem becomes equivalent to the one of the existence of an isomorphism between any two graphs. Let us recall that this problem is still unclassified and that its class is called "isomorphism complete". Otherwise, problem *3* is NP-complete:

Property 11.

The "isoprojection" problem is NP-complete - even if we are given a total order on the c-vertex labels.

Proof.

We prove the NP-completeness of "isoprojection" by proving the NP-completeness of a particular case, the following problem ***A***, which restricts the S-graphs to directed graphs labelled on the vertices, the labels being totally ordered.

Problem A: *Labelled graphs isomorphism*

Instance	:	Two directed graphs labelled on the vertices, G_1 and G_2, $\geq$ a total order on the label set.
Question	:	Is there an isomorphism b of G_1 into G_2 satisfying: For every vertex x of $G1$, $label(x) \geq label(b(x))$?

Let us consider the NP-complete problem *"isomorphic partial graph".*

Instance	:	Two non directed graphs $H=(X_h,U_h)$ and $K=(X_k,U_k)$ with the same vertex number n,

Question : Is H isomorphic to a partial graph of K (*i.e.* a graph obtained by deleting edges) ?[1]

We will reduce "isomorphic partial graph" to the previous problem A. We transform H and K in bipartite graphs H' and K' as follows:
let Y_h, Y_k be the vertex sets corresponding to the edges of complete graphs whose vertex sets are respectively X_h and X_k; H' is bipartite on Y_h and X_h, every element xy of Y_h having the neighbours x and y in X_h *(see Figure 10)*. All edges have the same direction (for instance from X_h to Y_h). Similarly, we construct K' on Y_k and X_k.
Let us consider the total order $\leq$ on $\{0,1\}$: every vertex xy of Y_h (*resp.* Y_k) is labelled 0 if xy is an edge of H (*resp.* K) and 1 otherwise. The vertices of X_h and X_k are labelled 0.

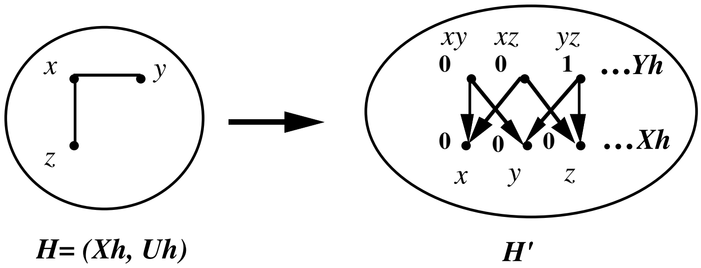


Figure 10. *Transforming H into H' (in proof of property 11)*

There is a labelled graph isomorphism of H' onto K' if and only if H is isomorphic to a partial graph of K. ❑

Problem 4: *Isofusionnability*

Instance	:	Two S-graphs G_1 and G_2.
Question	:	Are G_1 and G_2 isofusionnable?

<u>*Property 12.*</u>

The "isofusionnability" problem is NP-complete.

[1] This problem is NP-complete since it contains as a particular case the NP-complete problem "isomorphic spanning tree" - Given a graph G and a tree T, does G contain a spanning tree isomorphic to T? - [Garey Johnson 79, ND8].

Proof.

Let us consider the following particular case of isofusionnability.

Problem B: *Labelled graphs isofusionnability*

Instance	:	Two (directed) graphs G_1 and G_2 labelled on the vertices, and the label set is structured in a lattice.
Question	:	Is there an isomorphism of G_1 onto G_2 satisfying: for every vertex x of $G1$, $label(x) \wedge label(b(x)) \neq 0$?

We will reduce problem A to problem B. Let us consider an instance of problem A, with the total order T on the label set: $x_1 < x_2 < \ldots < x_n$. Let T' be the lattice constructed as follows *(see Figure 11)*:

it has a chain $y_1 < y_2 < \ldots < y_n$,

n vertices $z_1, z_2, \ldots z_n$, with, for every i, z_i is covered by y_i,

there is an element 0 covered by each z_i.

Let G'_1 (*resp.* G'_2) be the graph obtained from G_1 (*resp.* G_2) by replacing labels x_i by y_i (*resp.* z_i.). It is easily verified that the label of a vertex x of G_1 is greater or equal to the label of a vertex y of G_2 if and only if the lower bound, in T', of the x label in G_1 and the y label in G'_2 is $\neq 0$. Thus there is an isomorphism between G_1 and G_2 satisfying the condition of problem A if and only if there is an isomorphism between G_1 and G'_2 satisfying the condition of problem B. ❑

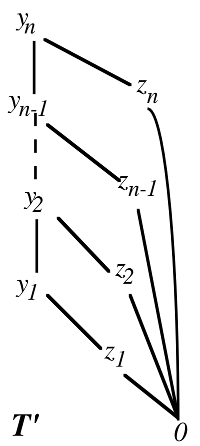


Figure 11. *The lattice T' (in proof of property 12)*

Problem 5: *Equivalence*

Instance	:	Two S-graphs A and B.
Question	:	Are A and B equivalent ?

Property 13.

The "equivalence" problem is NP-complete.

Proof.

Let G_1 and G_2 be instances of "projection". We construct G'_1 (*resp.* G'_2) by adding to G_1 (*resp.* G_2) an individual c-vertex x whose referent does not appear in any G_1 or G_2 vertex label, and by introducing for each c-vertex y of G_1 (*resp.* G_2) a binary r-vertex with type r with first neighbour x and second neighbour y. We construct a third S-graph G''_2 by joining G'_1 and G'_2 on x *(see Figure 12).*
We will show that there is a projection from G_1 to G_2 if and only if G'_2 and G''_2 are equivalent. If there is a projection from G_1 to G_2, then it is immediately verified that there is a projection from G''_2 to G'_2; since there is a trivial projection from G'_2 to G''_2, we only have to prove the reciprocal.
So let us assume that G'_2 and G''_2 are equivalent. Hence there is a projection from G''_2 to G'_2. x can be projected only on x since it is an individual c-vertex whose marker does not appear in G_2. Furthermore, no vertex of G''_2, and in particular of G_1, can be projected on x otherwise $G'2$ would have a loop on x containing r. Thus, the restriction to G_1 of the projection from G''_2 to G'_2 is a projection from G_1 to G_2. ❑

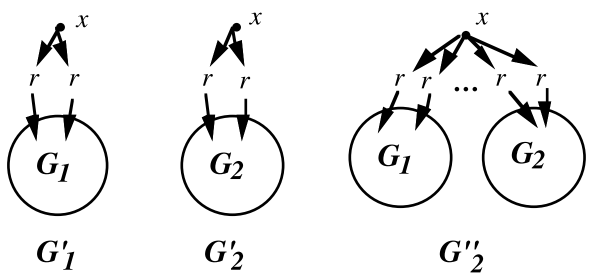


Figure 12. *Building G'1, G'2 and G"2 from G1 and G2 (in proof of property 13)*

Remark.

One sidelight of the above proof is that it actually shows that the problem is still NP-complete when B is isomorphic to a strict sub-S-graph of A: it suffices to note that, in the proof, G'_2 is isomorphic to a sub-S-graph of G''_2.

Problem 6: *Irredundance*

Instance	:	One S-graph A.
Question	:	Is A irredundant ?

<u>*Property 14.*</u>

The "irredundance" problem is co-NP-complete.

Proof.

Let us recall than a problem is co-NP-complete if it is the complementary of an NP-complete problem. "Irredundance" is the complementary of the following problem: "is there a projection from A to one of its strict sub-S-graphs?", on which we will prove the NP-completeness. Let us call this problem $\boldsymbol{6^c}$.
Assuming that A has no twin r-vertices, and remarking that a wanted projection is not necessarily surjective, we can reformulate 6^c as follows: "Is there a projection from A to one of its sub-S-graphs corresponding to a connected component of the graph obtained by deleting in A a c-vertex and its neighbours?".
Let us consider the following particular case of "injective projection" (problem *2*) that remains NP-complete: "Given the S-graphs A and B, and x a c-vertex of A, y a c-vertex of B, is there an injective projection Π from A to B, with $\Pi(x)=y$?".
We will transform an instance of the above problem, say $\{A, B, x, y\}$, to an instance of problem 6^c, say G', as follows:
Let r be a binary r-vertex type that does not appear as a label of any A or B r-vertex. We construct the S-graph G from A and B and new c-vertices c and c', with $lab(c) = lab(c')$ linked to x and y as illustrated in following Figure *13*.

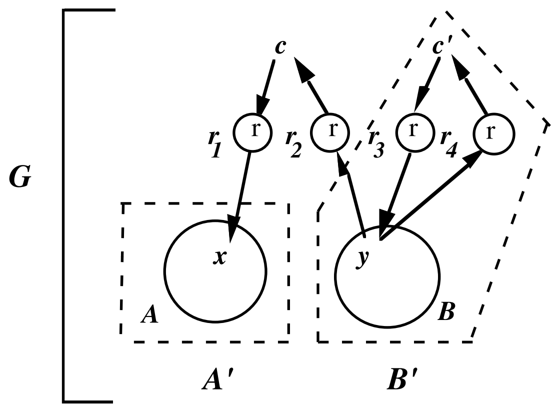


Figure 13. *Constructing G (in proof of property 14)*

The graph $(G \setminus c)$ obtained from G by deleting the c-vertex c and its neighbours, has two connected components A' and B': $A'=A$ and B' includes B. There is a projection Π from A to B with $\Pi(x) = y$ if and only if there is a projection Π' from G to $(G \setminus c)$, *i.e.* from G to B' since necessarily $\Pi'(c')= c'$; for example, $\Pi'=\Pi\cup\{(c',c'), (c,c'), (r_1,r_3), (r_2,r_4)\}$.
Let s be a binary r-vertex type that does not appear as a label of any G r-vertex. With every c-vertex pair (c_1,c_2) of A (*resp.* of B), we associate two r-vertices of type s, one with first neighbour c_1 and with second neighbour c_2, and vice-versa for the other r-vertex. In doing so, we transform A and B to irredundant graphs A_s and B_s and every projection from A_s to B_s becomes injective. Let G' be the graph finally obtained from G. There is an injective projection Π from A to B with $\Pi(x) = y$ if and only if there is a projection from G' to one of its strict sub-S-graphs (necessarily B'_s). ❑

In order to study polynomial cases for the above problems, let us introduce S-graphs whose underlying graphs are trees.

Definition.

An **S-tree** is an S-graph without cycles, except cycles created by multi-edges between an r-vertex and a c-vertex. A **rooted S-tree** is an S-tree satisfying: if we orientate the S-tree - or choose a root - such that, for every r-vertex, its predecessor is its first neighbour, and its successors are its other neighbours, then we obtain a rooted tree.

S-trees are frequently used in type definition of relations or concepts (see for example [Sowa 84]). Let us also mention the work of M. Liquière in machine learning, where rooted S-trees are used to describe the knowledge learnt on objects represented by conceptual graphs [Liquière Sallantin 89].

Polynomial cases for the previous mentioned problems are based upon these particular S-graphs. Since the aim of this paper concerns fundamental notions and not concrete algorithmic problems, we only mention, without proof, the principal results (for algorithms and proofs, see [Mugnier Chein 92] and [Mugnier 92]).

Projection*:* when A is an S-tree and B is any S-graph, then it is immediate to find a polynomial algorithm computing a projection from A to B, and counting the projections from A to B. If B is a rooted S-tree and A is any S-graph, we also have a polynomial algorithm, because we can construct from A a rooted S-tree, say A', such that each projection from A to B corresponds to exactly one projection from A' to B.

Sub-S-graph*:* the polynomial cases of this problem come directly from those known for *subgraph isomorphism* [Garey Johnson 79, GT 48]. In particular, when A and B are S-trees, we have a polynomial algorithm that computes an injective morphism (or projection) from A to B. This algorithm is an extension of the one analyzed in [Reyner 77]. On the other hand, when A is an S-tree and B is any S-graph, then the problem is still NP-complete. Furthermore, counting the injective morphisms between two rooted S-trees is an NP-hard problem.

Polynomial cases for ***equivalence*** come directly from *projection*. We do not know if there are S-graphs classes for which *equivalence* is polynomial whereas *projection* is NP-complete.

Isoprojection and ***isofusionnability*** are polynomial on S-trees. Is it possible to find other S-graphs classes for which these problems are polynomial? These classes should be derived from polynomial cases known for classical graph isomorphism problem.

11. Conclusion

In this paper, we tried to state precise definitions for what we think to be Sowa's basic notions, and we studied the first consequences of these definitions. We yielded a complete correspondence between the ground S-graph operations, namely projection and generalization / specialization rules. On a logical viewpoint, we established that the generalization operations define a complete set of inference rules on the set of formulas associated with S-graphs. On an algorithmic complexity viewpoint, the graph formulation allowed us to characterize some polynomial cases of the main problems we encountered. Finally, the fact that the $\leq$ relation is only a preorder is a true distinction, because the problem of determining whether two S-graphs are equivalent is NP-complete.

Nevertheless there are yet a lot of problems to study and we will indicate those we are working on now.

A first point is the final clarification of some basic notions.

As an example let us consider the ambiguities related to the fundamental specialization / generalization relation. Some definitions and results of [Sowa 84] implicitly refer to *injective* projection[1]. Consequently, coherence with previous results implies to limit the use of specialization rules: in particular, it should be forbidden to join c-vertices of a *same* graph. This change is not a slight one since it leads, in particular, to restrict the fusionnability to isofusionnability, and the extended join to the isojoin. Furthermore the specialization relation becomes $H <_{\mathrm{ssg}} G$ if and only if there exists an isoprojection from G to a sub-S-graph of H. This notion seems to be widely used, for instance in the french IBM project "KALIPSOS" [Fargues and al. 86]. Let us indicate some immediate consequences of the latter approach. On one hand, it provides a partial order relation. On the other hand, it drastically restricts the comparability between two S-graphs. The complexity of the comparison between two S-graphs can even increase. For instance if we are given an S-tree A and an S-graph B then the problem $B <_{ssg} A$ is NP-complete whereas $B \leq A$ is polynomial (see *projection* and *sub-S-graph* problems of section 10). These are only remarks and a careful study would be necessary to point out all consequences of the restricted definition. In addition, a true comparative study of the two definitions needs a discussion on the use of each approach, in relation with works in domains like concept learning from examples, *e.g.* [Haussler 89], or pattern recognition, *e.g.* [Shapiro Haralick 81].

A second point is to extend the kernel of notions.

In the basic model we presented, the set of S-graphs definable on a support S is exactly the set of S-graphs derivable by a sequence of elementary specialization rules from the basis B (the star graph set), or the generic c-vertex with universal type if the S-graph is reduced to a single c-vertex. This notion of a basis can be generalized in considering, instead of a star graph set, a set of any conceptual graphs, called "canonical basis" in [Sowa 84]. Any S-graph derivable from the canonical basis by a sequence of elementary specialization rules is then said to be canonical. The introduction of a canonical basis leads to new problems (in particular, the recognition of a canonical graph) and requires to adapt our results: to reconsider the correspondence between projection and specialization rules, to redefine the generalization rules (in particular, a sub-S-graph of a canonical S-graph is not necessarily a canonical S-graph) and to reformulate the links with the logical operations.

[1]For instance, Sowa's results on "compatible projections" which define the "extended join" - theorem 3.5.7. of [Sowa 84] - do not hold if projections are not *injective* on the c-vertex set.

Sowa uses a lot of different types of markers and it is clearly insufficient to consider only individual markers. In the same way, an actual application needs dynamic definitions of types of concepts and relations, thus a careful algorithmic study of type contraction and expansion should be useful.

12. References

[AWCG 88] *Proceedings of the third Annual Workshop on Conceptual Graphs,* at AAAI-88, St Paul, Minesota, (1988).

[AWCG 89] *Proceedings of the fourth Annual Workshop on Conceptual Graphs,* at IJCAI-89, Detroit, Michigan (1989).

[AWCG 90] *Proceedings of the fifth Annual Workshop on Conceptual Graphs,* American meeting at AAAI-90, Boston, MA, European meeting at ECAI-90, Stockholm, Sweden (1990).

[AWCG 91] *Proceedings of the sixth Annual Workshop on Conceptual Graphs,* Bighamton, NY, Way E. (Ed), Suny Bighamton, (1991).

[AWCG 92] *Proceedings of the seventh Annual Workshop on Conceptual Graphs,* Las Cruces, New Mexico, Pfeiffer H.D. (Ed), New Mexico State University, (1992).

[Chein 89] Chein M., “S-graphes (1): notions de base”, *R.R. CRIM, 87,* (Octobre 89).

[Clancey 85] Clancey W.J., “Sowa's book review”, *Artificial Intelligence, 27,* (1985), 113-128.

[Ellis 91] Ellis G., “Compiled Hierarchical Retrieval”, *Proceedings of the sixth Annual Workshop on Conceptual Graphs,* Bighamton, NY, Way E. (Ed), Suny Bighamton, (1991), 187-207.

[Fargues and al. 86] Fargues J., Landau M.C., Dugourd A., Catach L., Conceptual Graphs for Semantics and Knowledge Processing, *IBM J. Res. Develop., 30, 1,* (1986), 70-79.

[Garey Johnson 79] Garey M.R., Johnson D.S., *Computers and intractability - A guide to the Theory of NP-Completeness*, Freeman W.H. and Co, (1979).

[Haussler 89] Haussler D., “Learning conjunctive concepts in structural domains”, *Machine Learning, 4,* (1989), 7-40.

[Jackman 88] Jackman M.K., Inference and the Conceptual Graph Knowledge Representation Language, in Moralee S. (Edt), *Research and Development in Expert Systems IV,* Cambridge University Press, (1988).

[JETAI 92] *Journal of Experimental & Theoretical Artificial Intelligence (JETAI),* “Special Issue: Conceptual Graphs”, 4, (2), (September 92).

[KBS 92] *Knowledge-Based Systems,* “Special Issue: Conceptual Graphs”, 5, (3), (September 92).

[Liquière Sallantin 89] Liquière M., Sallantin J., “INNE: Induction in Networks”, *IEEE International Workshop on Tools for IA,* Fairfax USA, (1989).

[Mugnier Chein 92] Mugnier M.L., Chein M., “Polynomial algorithms for projection and matching”, *Proceedings of the seventh Annual Workshop on Conceptual Graphs,* Las Cruces, New Mexico, Pfeiffer H.D. (Ed.), New Mexico State University, (1992), pp 49-58.

[Mugnier 92] Mugnier M.L., *“Contributions algorithmiques pour les graphes d’héritage et les graphes conceptuels”,* PhD thesis, LIRMM & Université Montpellier II, (October 92).

[Nagle ed. 92] *Conceptual Structures - Current Research and Practice,* Nagle T., Nagle J., Gerholtz L., Eklund P. (Eds), Horwood E. Publishers, (1992).

[Reyner 77] Reyner S.W., “An analysis of a good algorithm for the subtree problem”, *SIAM J. Computer, Vol. 6, N°4,* (December 1977), 730-732.

[Shapiro Haralick 81] Shapiro L.G., Haralick R., “Structural Description and Inexact Matching”, *IEEE Transactions, Vol. PAMI-3, 5,* (September 81).

[Smoliar 87] Smoliar S.W., “Sowa's book review”, *Artificial Intelligence, 33,* (1987), 259-266.

[Sowa 84] Sowa J.F., *Conceptual Structures - Information Processing in Mind and Machine,* Addison-Wesley (1984).

[Sowa 88] Sowa J.F., “On Conceptual Structures, a response to the review by S.W. Smoliar”, *Artificial Intelligence, 34,* (1988), 388-394.

[Sowa 92] Sowa J.F., “Conceptual Graphs as a universal knowledge representation”, in *Semantics Networks in Artificial Intelligence,* Lehmann F. (Ed.), Pergamon Press, (1992), 75-93.