



HAL
open science

Simple Conceptual Graphs and Simple Concept Graphs

Jean-Pierre Aubert, Jean-François Baget, Michel Chein

► **To cite this version:**

Jean-Pierre Aubert, Jean-François Baget, Michel Chein. Simple Conceptual Graphs and Simple Concept Graphs. ICCS: International Conference on Conceptual Structures, Jul 2006, Aalborg, Denmark. pp.87-101, 10.1007/11787181_7. lirmm-00113155

HAL Id: lirmm-00113155

<https://hal-lirmm.ccsd.cnrs.fr/lirmm-00113155>

Submitted on 10 Nov 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Simple Conceptual Graphs and Simple Concept Graphs

J.P. Aubert¹, J.-F. Baget², and M. Chein¹

¹ LIRMM {aubert, chein}@lirmm.fr;

² INRIA/LIRMM baget@inrialpes.fr

Abstract. Sowa’s Conceptual Graphs and Formal Concept Analysis have been combined into another knowledge representation formalism named Concept Graphs. In this paper, we compare Simple Conceptual Graphs with Simple Concept Graphs, by successively studying their different syntaxes, semantics, and entailment calculus. We show that these graphs are almost identical mathematical objects, have equivalent semantics, and similar inference mechanisms. We highlight the respective benefits of these two graph-based knowledge representation formalisms, and propose to unify them.

1 Introduction

Introduced in [19], Conceptual Graphs were extended in [20]. Since [5], the “Montpellier school of conceptual graphs” has been studying this knowledge representation formalism as a family of formal languages whose objects are graphs and where inferences are computed using graph-based operations (e.g. [3]). In the same way, [22] has proposed to combine conceptual graphs with Formal Concept Analysis (FCA). This work has been developed in [18, 7, 8].

In this paper, we compare these two approaches and focus on the mathematical and computational viewpoints. Since we are interested in conceptual graphs and concept graphs as logics, we will successively compare the syntax (Sect. 2), semantics (Sect. 3), and calculus (Sect. 4) of these two languages.

2 Syntax

We show here that simple conceptual graphs and simple concept graphs are avatars of the notion introduced by Sowa [19]. Only simple conceptual (or concept) graphs are considered, thus the adjective *simple* is implicit hereafter.

In the first subsection, we show that, up to a well-known transformation, the objects described by bipartite graphs and directed hypergraphs have the same structure. Then we show that the *vocabulary* (or support) upon which conceptual graphs are defined and the *alphabet* used for concept graphs are identical, with some minor variants. Finally, we compare various definitions used for conceptual and concept graphs.

2.1 Hypergraphs and bipartite graphs

Let us recall a very well known bijection between hypergraphs and bipartite graphs (see [4] for relationships between graphs and hypergraphs and [10] for a more recent introduction to graph theory). Let $H = (X, \mathcal{E})$ be a hypergraph over X , that is X is a set (of vertices) and \mathcal{E} a set of hyperedges i.e. non-empty subsets of X . Let $\alpha(H)$ be the bipartite graph (X, R, E) defined as follows:

- R is disjoint from X and there is a bijection f from \mathcal{E} to R ,
- let $c \in C$ and $r \in R$, (c, r) is in E iff $c \in f^{-1}(r)$.

It is simple to check that α is a bijection from the set of hypergraphs over X to the set of bipartite graphs with the first vertex set X .

$\alpha(H)$ is called the *incidence (bipartite) graph of H* . It is straightforward to extend the bijection α to a bijection from multi-hypergraph (i.e. \mathcal{E} is no longer a set of subsets but rather a family of subsets) to bipartite multi-graphs (E is a family of edges). Let us call *ordered multi-hypergraph* a multi-hypergraph in which any hyperedge is totally ordered. Let us call *ordered bipartite multi-graph* a bipartite multi-graph in which any set of edges incident to a vertex of R is totally ordered. α can be trivially extended to the ordered objects and one gets:

Property 1. The application α from the set of ordered multi-hypergraphs over X to the set of ordered bipartite multi-graphs with first vertex set X is a bijection.

A bipartite graph is a graph, this trivial remark leads to a first important consequence (other will be discussed after introducing conceptual graphs). When one wants to graphically represent a hypergraph, a representation of its incidence bipartite graph is generally drawn (see FIG. 1). Although conceptual graphs are usually defined via bipartite graphs, the alternative hypergraph definition is sometimes used (e.g. [2], for more efficient algorithms).

2.2 Vocabulary and alphabet

The structure, called *support* in [5], encoding *terms*, as well as type orderings is the core of the *canon* [20]. Here we use the name *vocabulary*, which is more standard in KR. A similar structure, named *alphabet*, is used in the concept graphs formalism [18, 7]. In this subsection we compare these two structures.

Definition 1 (Vocabulary). A vocabulary is a triple (T_C, T_R, \mathcal{I}) where:

- T_C, T_R, \mathcal{I} are pairwise disjoint sets.
- T_C , the set of concept types, is partially ordered by a relation \leq and has a greatest element denoted \top .
- T_R , the set of relation symbols, is partially ordered by a relation \leq , and is partitioned into subsets T_R^1, \dots, T_R^k of relation symbols of arity $1, \dots, k$ respectively. The arity of a relation r is denoted $\text{arity}(r)$. Furthermore, any two relations with different arities are not comparable.
- \mathcal{I} is the set of individual markers.

Definition 2 (Alphabet). Relationships between an alphabet $(\mathcal{G}, \mathcal{C}, \mathcal{R})$ and a vocabulary are as follows:

- $\mathcal{G} = \mathcal{I}$ the object names are the individual markers,
- $\mathcal{C} = T_C$ the concept names are the type of concepts,
- $\mathcal{R} = T_R \cup \{=\}$ the relation names include the equality symbol.

Some definitions incorporate the generic marker $*$ in the vocabulary. This is not necessary, since it is the same for all vocabularies, so we will only introduce it in the definition of conceptual graphs. In the same way, the equality symbol is not required in the definition of an alphabet. In logics, it is usually considered separately from the relation symbols.

Both vocabulary and alphabet encode the same information, which could be represented in logics by an ordered FOL language without function symbols in which some unary predicates are distinguished.

2.3 Comparing Conceptual Graphs and Concept Graphs

Conceptual Graphs The following definition is directly inspired from [5]. We will add in Sect. 4 conditions about coreference (they are only relevant for computational purposes).

Definition 3 (Conceptual graph). A conceptual graph over a vocabulary \mathcal{V} is a 5-tuple $(C, R, E, l, \text{coref})$ such that:

- (C, R, E) is a multi-bipartite graph,
- coref is an equivalence relation over C ,
- l is a labelling function of $C \cup R$ such that:
 - for any $x \in C$, $l(x) \in T_C \times (\mathcal{I} \cup \{*\})$,
 - for any $x \in R$, $l(x) \in T_R$
 - for any $x \in R$, the edges incident to x are labelled $\{1, \dots, \text{arity}(l(x))\}$

Concept Graphs The first definition of a concept graph was proposed in [22]. We also present here the definitions from [18] and [7].

Definition 4 (Concept graph, [22]). An abstract concept graph is a structure $\mathfrak{G} = (V, F, \nu, D, \kappa, \theta)$ for which:

- V and F are finite sets and ν is a mapping of E to $\bigcup_{k=1}^n V^k$ ($n \geq 2$ s. t. (V, F, ν) can be considered as a finite directed multi-hypergraph with vertices from V and edges from F (we define $|e| = k$ if $\nu(e) = (v_1, \dots, v_k)$),
- D is a finite set and κ a mapping of $V \cup F$ to D s. t. $\kappa(e_1) = \kappa(e_2) \Rightarrow |e_1| = |e_2|$ (the elements of D may be understood as abstract concepts),
- θ is an equivalence relation on V .

Prediger [18] slightly transforms the previous definition by removing the label set D , and replacing it by the (exterior) notion of an alphabet:

Definition 5 (Concept graph, [18]). A concept graph over the alphabet $(\mathcal{C}, \mathcal{G}, \mathcal{R})$ is a structure $\mathfrak{G} = (V, F, \nu, \kappa, \rho)$, where

- (V, F, ν) is a finite directed multi-hypergraph
- $\kappa: V \cup F \rightarrow \mathcal{C} \cup \mathcal{R}$ is a mapping such that $\kappa(V) \subseteq \mathcal{C}$ and $\kappa(F) \subseteq \mathcal{R}$, and all $e \in F$ with $\nu(e) = (v_1, \dots, v_k)$ satisfy $\kappa(e) \in \mathcal{R}_k$,
- $\rho: V \rightarrow \mathfrak{P}(\mathcal{G}) \setminus \{\emptyset\}$ is a mapping.

There are two other syntactical differences between Wille and Prediger, in Wille’s definition there is an equivalence relation over V , which is not the case in Prediger, and in Prediger two labels are associated to an element of V : an element of \mathcal{C} and a non-empty subset of \mathcal{G} . Thus, as Prediger said [18]:

”Apart from some little differences, the concept graphs correspond to the simple conceptual graphs as defined in [5] or [20].”

More precisely, it is straightforward to extend the canonical bijection α from a class of ordered multi-hypergraphs to the class of their incidence graphs to an injective mapping, also called α , from the class of concept graphs over \mathcal{V} to the class of conceptual graphs over \mathcal{V} .

Let $\mathfrak{G} = (V, F, \nu, \kappa, \theta)$ be a concept graph, $G = \alpha(\mathfrak{G}) = (C, R, E, l, \text{coref})$ is defined as follows. Any $x \in V$ with $\rho(x) = \{g_1, \dots, g_k\}$, $k \geq 2$, is duplicated into k nodes x_1, \dots, x_k . C is the union of the $\{x \in V \text{ s.t. } |\rho(x)| = 1$ and the set of duplicated nodes. If $\rho(x) = \{g\}$ then $l(x) = (\kappa(x), g)$. If $\rho(x) = \{g_1, \dots, g_k\}$ then $l(x_i) = (\kappa(x), g_i)$.

Any $e \in F$ with $\nu(e) = (v_1, \dots, v_k)$ is transformed into $|\rho(v_1)| \times \dots \times |\rho(v_k)$ relation nodes of R with label $\kappa(e)$ and whose neighborhood are the *arity*($\kappa(e)$)-tuples associated with $\rho(v_1) \times \dots \times \rho(v_k)$. The equivalence *coref* is the discrete equivalence. This coding preserves the graphs semantics (Sect. 3).

Let’s consider now the third definition of concept graphs [7].

Definition 6 (Concept graphs, [7]). A concept graph over \mathcal{V} is a structure $\mathfrak{G} = (V, F, \nu, \kappa, \rho)$, where:

- V and F are pairwise disjoint, finite sets whose elements are called vertices and edges,
- $\nu: F \rightarrow \bigcup_{k \in \mathbb{N}} V^k$ is a mapping (we write $|e| = k$ for $\nu(e) \in V^k$,
- $\kappa: V \cup F \rightarrow \mathcal{C} \cup \mathcal{R}$ is a mapping such that $\kappa(V) \subseteq \mathcal{C}$ and $\kappa(F) \subseteq \mathcal{R}$, and all $e \in F$ with $|e| = k$ satisfy $\kappa(e) \in \mathcal{R}_k$,
- $\rho: V \rightarrow \mathfrak{G} \cup \{*\}$ is a mapping.

This is almost the definition of conceptual graphs (modulo α). Instead of considering the equivalence relation induced by coreference links, it keeps, as Sowa, coreference links. Considering two steps as in DEF 6 (a symmetric relation over C or V , then its reflexo-transitive closure *coref* or θ), or directly the equivalence relation as in DEF 3 and 4 is a matter of taste.

Let $\mathfrak{G} = (V, E, \nu, \kappa, \rho)$ be a Dau’s concept graph over \mathcal{A} . The conceptual graph $\alpha(G) = (C, R, E, l, \text{coref})$ is defined as follows:

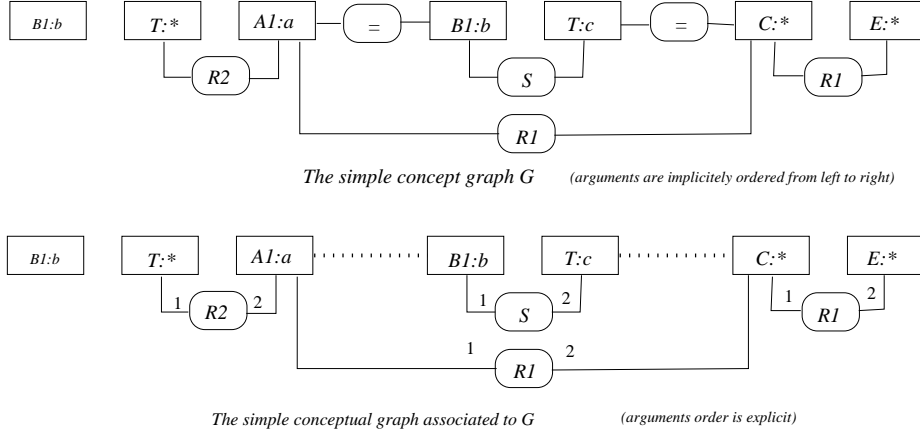


Fig. 1. Drawings of G and of $\alpha(G)$

- $C = V$, the concept nodes of $\alpha(G)$ are the vertices of G ,
- R is a set in bijection with E , to each edge e of G a relation node noted $\alpha(e)$ is associated (α is a bijection and C and R are disjoint),
- if $\nu(e) = (c_1, \dots, c_k)$, then for $i = 1, \dots, k$, $\{r = \alpha(e), c_i\}$ is an edge of $\alpha(G)$,
- the label of a concept node c is $l(c) = (\kappa(c), \rho(c))$,
- the label of a relation node r is $l(r) = \kappa(\alpha^{-1}(r))$,
- the label of an edge $\{r = \alpha(e), c_i\}$ of $\alpha(G)$ is i ,
- a class of *coref* is composed of a connected component of the graph $(V, =)$.

Let us consider the concept graph in FIG. 1 (copied from [7]). If the equality relation is replaced by coreference links, this is the drawing of $\alpha(\mathfrak{G})$. Note that, in the drawing of \mathfrak{G} there are no labels on the edges, but at least for the edges incident to the oval vertices labelled R_1, R_2, S they must be added since it is not stated that these relations are symmetrical. We think that it is interesting to consider that the equality is a specific relation, and this is achieved by *coref*, which is an equivalence relation and by drawing it with specific edges (coreference links). In Figure 1 a drawing of $\alpha(G)$ is reproduced besides the drawing of G .

At this moment of our discussion, preferring concept or conceptual graphs is a matter of taste not of real mathematical differences, since they are almost the same mathematical objects. In the rest of the paper, we will now call CGs conceptual as well as concept graphs, and will consider them as the same objects.

3 Semantics

In logics, semantics are provided to define the conditions under which an assertion is true. An *interpretation* is a structure encoding a possible world. An interpretation is a *model* of a formula F if the assertion encoded by F is true in that world. The notions of interpretations and models lead to *logical consequence* (or *entailment*), whose calculus will be detailed in Sect. 4.

Definition 7. Let \mathcal{L} be a logic, and G and H be two formulas of \mathcal{L} . We say that G entails H (or that H is a logical consequence of G) and note $G \models_{\mathcal{L}} H$ iff every interpretation that is a model of G is also a model of H .

We show here that models of CGs, defined in standard model-theoretic semantics [20, 16] or in Formal Concept Analysis [22, 18, 7], are equivalent.

3.1 Semantics for Conceptual Graphs

Historically, conceptual graphs semantics have been presented by a translation Φ to FOL. This “logical semantics” is equivalent to model-theoretic semantics.

FOL semantics of conceptual graphs FOL semantics of conceptual graphs described below were introduced by Sowa [20]. A FOL language is associated to a vocabulary \mathcal{V} , and is composed of a set of constants equal to \mathcal{I} and a set of predicates equal to $T_C \cup T_R$ with their arities. The order over the symbol types is represented by the following set of formulas.

Definition 8 ($\Phi(\mathcal{V})$). *Type ordering is translated by: $\forall t_1, t_2$ types of \mathcal{V} of arity p such that $t_2 \leq t_1$, we obtain the formula $\forall x_1 \dots x_p (t_2(x_1, \dots, x_p) \rightarrow t_1(x_1, \dots, x_p))$.*

Definition 9 ($\Phi(G)$). *Given any CG G , the formula $\Phi(G)$ is built as follows.*

1. A term $term(c)$ is assigned to each concept c in the following way. If c is generic (labelled by $*$), then $term(c)$ is a variable, and if c and c' are two different generic concepts, then $term(c) \neq term(c')$. If c is labelled by the individual i , then $term(c) = i$.
2. An atom is assigned to each relation or concept:
 - the atom $t(term(c))$ is assigned to each concept c of type t ;
 - the atom $r(term(c_1), \dots, term(c_k))$ is assigned to each relation node x , where r is its type, k the arity of r and c_i denotes the i -th neighbor of x .
3. To any coreference link between two nodes c and c' is associated the formula $term(c) = term(c')$. Let $\varphi(G)$ be the conjunction of all preceding atoms.
4. $\Phi(G)$ is the existential closure of $\varphi(G)$.

It is simple to check that definition 6 of a concept graph is simply a graph reformulation of such a formula.

Model Theoretic Semantics for conceptual graphs It is a direct translation [16] of the model-theoretic semantics of the formulas obtained by Φ .

Definition 10 (Interpretation of terms). *The terms of a vocabulary (T_C, T_R, \mathcal{I}) are the elements of $T_C \cup T_R \cup \mathcal{I}$. Their interpretation is a pair (D, δ) s.t.:*

- D is a non-empty set;
- δ maps each marker of \mathcal{I} to an element of D , each concept type of T_C to a subset of D , and each relation type of arity k in T_R to a subset of D^k .

Definition 11 (Model of a vocabulary). A model of a vocabulary \mathcal{V} is an interpretation $I = (D, \delta)$ of its terms s. t.:

- $\forall c, c' \in T_C, c \leq c' \Rightarrow \delta(c) \subseteq \delta(c')$
- $\forall r, r' \in T_R, r \leq r' \Rightarrow \delta(r) \subseteq \delta(r')$

Definition 12 (Model of a graph). Let $G = (C, R, E, l, \text{coref})$ be a conceptual graph over a vocabulary \mathcal{V} . An interpretation (D, δ) of the terms of \mathcal{V} is a model of G iff there is a mapping (an assignment) α from C to D s. t.:

- For any individual concept c with marker i : $\alpha(c) = \delta(i)$;
- $\forall c \in C, \alpha(c) \in \delta(\text{type}(c))$;
- $\forall r \in R$ with neighbors $(c_1, \dots, c_k), (\alpha(c_1), \dots, \alpha(c_k)) \in \delta(\text{type}(r))$;
- $\forall c, c' \in C, (c, c') \in \text{coref} \Rightarrow \alpha(c) = \alpha(c')$.

It is easy to check that the models (as usually defined in FOL) of $\Phi(\mathcal{V})$ (resp. $\Phi(G)$) are exactly the models of \mathcal{V} (resp. G).

Definition 13 (Deduction for Conceptual Graphs). Let \mathcal{V} be a vocabulary, and G and H be two conceptual graphs over \mathcal{V} . We say that H is deducible from G and \mathcal{V} and note $\mathcal{V}, G \models H$ iff every interpretation that is a model of \mathcal{V} and G is also a model of H .

3.2 Semantics for concept graphs

The first semantics, based upon power context families [22], for concept graphs was given by Prediger [18], we present here the slightly different version in [7].

Definition 14 (Power context family). A power context family is a family $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \dots)$ of formal contexts $\mathbb{K}_k := (G_k, M_k, I_k)$ such that $G_0 \neq \emptyset$ and for every $k : G_k \subseteq (G_0)^k$. The elements of G_0 are the objects of $\vec{\mathbb{K}}$. A pair (A, B) with $A \subseteq G_k$ and $B \subseteq M_k$ is called a concept of \mathbb{K}_k if and only if $A = \{g \in G_k \mid g I_k b \text{ for all } b \in B\}$ and $B = \{m \in M_k \mid a I_k m \text{ for all } a \in A\}$. A is called the extension $\text{ext}((A, B))$ and B is called the intension $\text{int}((A, B))$ of the concept (A, B) . The set of all concepts of a formal context \mathbb{K}_k is denoted by $\mathfrak{B}(\mathbb{K}_k)$. The elements of $\bigcup_{k \in \mathbb{N}_0} \mathfrak{B}(\mathbb{K}_k)$ are called concepts, and the elements of $\mathfrak{R}_{\vec{\mathbb{K}}} = \bigcup_{k \in \mathbb{N}} \mathfrak{B}(\mathbb{K}_k)$ are called relation-concepts.

The structure used to interpret concept graphs is a power context family. Below we split the definition of [7] (to differentiate interpretations and models), and use conceptual graphs notations to facilitate comparison.

Definition 15 (Contextual interpretation). Let \mathcal{V} be a vocabulary. A contextual interpretation of the terms of \mathcal{V} is a pair $(\vec{\mathbb{K}}, \lambda)$, where $\vec{\mathbb{K}}$ is a power context family and λ is a mapping that maps each marker of \mathcal{I} to an element of G_0 , each concept type of T_C to an element of $\mathfrak{B}(\mathbb{K}_0)$ (i.e. a formal concept of \mathbb{K}_0), and each relation type of arity k in T_R to an element of $\mathfrak{B}(\mathbb{K}_k)$.

Definition 16 (Contextual model of a vocabulary). Let \mathcal{V} be a vocabulary. A contextual interpretation of the terms of \mathcal{V} is a contextual model of \mathcal{V} iff the mapping λ is order-preserving.

Equivalence between interpretations and contextual interpretations

Here we present two transformations, the first $c2i$ is from contextual interpretations into interpretations, and the second $i2c$ from interpretations into contextual interpretations. These transformations preserve the models of a vocabulary.

- ($c2i$) Let $C = (\overline{\mathbb{K}}, \lambda)$ be a contextual interpretation of \mathcal{V} . We obtain $c2i(C) = (G_0, \delta)$ where δ is defined by:
- $\forall i \in \mathcal{I}, \delta(i) = \lambda(i)$;
 - $\forall t \in T_C, \delta(t) = \text{ext}(\lambda(t))$;
 - $\forall r \in T_R, \delta(r) = \text{ext}(\lambda(r))$.
- ($i2c$) Let (D, δ) be an interpretation of \mathcal{V} . We obtain $i2c(I) = (\overline{\mathbb{K}}, \lambda)$ as follows:
- $G_0 = D$;
 - $\forall c, c' \in T_C$, we note $c \leq_\delta c'$ iff $\delta(c) \subseteq \delta(c')$. \mathbb{K}_0 is then the power context over G_0 associated with the partial order \leq_δ (Dedekind-MacNeille Completion theorem, [11], pp. 48);
 - The power contexts \mathbb{K}_i are constructed in the same way from the sets of relation types of arity i .

Property 2. Let \mathcal{V} be a vocabulary. I is a model of $\mathcal{V} \Rightarrow i2c(I)$ is a contextual model of \mathcal{V} ; conversely, C is a contextual model of $\mathcal{V} \Rightarrow c2i(C)$ is a model of \mathcal{V} .

Proof. We successively prove the two assertions of this property:

- $t \leq t' \Rightarrow \delta(t) \subseteq \delta(t')$ (since I is a model of \mathcal{V} , with t and t' being concept or relation types) $\Leftrightarrow t \leq_\delta t'$ (by construction of $i2c$) $\Leftrightarrow \lambda(t) \leq \lambda(t')$ (Def. 16).
- $t \leq t' \Leftrightarrow \lambda(t) \leq \lambda(t') \Leftrightarrow \text{ext}(\lambda(t)) \subseteq \text{ext}(\lambda(t')) \Leftrightarrow \delta(t) \subseteq \delta(t')$.

□

Definition 17 (Contextual model of a graph). Let \mathcal{V} be a vocabulary and $G = (C, R, E, l, \text{coref})$ be a CG over \mathcal{V} . A contextual interpretation of the terms of \mathcal{V} is a contextual model of G iff there is a mapping α from C into G_0 s.t.:

- if c is an individual concept node having marker i , $\alpha(c) = \lambda(i)$;
- $\forall c \in C, \alpha(c) \in \text{ext}(\lambda(\text{type}(c)))$;
- $\forall r \in R$, with neighbors (c_1, \dots, c_k) , $(\alpha(c_1), \dots, \alpha(c_k)) \in \text{ext}(\lambda(\text{type}(r)))$;
- $(c, c') \in \text{coref} \Rightarrow \alpha(c) = \alpha(c')$.

It is simple to check that the following property holds:

Property 3. Let \mathcal{V} be a vocabulary, and G be a CG over \mathcal{V} . I is a model of $G \Rightarrow i2c(I)$ is a contextual model of G ; conversely, C is a contextual model of $G \Rightarrow c2i(C)$ is a model of G .

Definition 18 (Deduction for Concept Graphs). Let \mathcal{V} be a vocabulary, and G and H be two concept graphs defined over \mathcal{V} . We say that H is deducible from G and note $G \models_c H$ iff all contextual models of G are also contextual models of H .

The following theorem proves the equivalence between the two semantics. Thanks to Props. 2 and 3., its proof is straightforward.

Theorem 1 (Equivalence of deductions). *Let \mathcal{V} be a vocabulary, and G and H be two CGs over \mathcal{V} . Then $\mathcal{V}, G \models H$ iff $\mathcal{V}, G \models_c H$.*

In concept graphs, concept lattices are used to define the order relation on concepts and relations as well as their interpretations. In conceptual graphs, there is a separation between the syntax (the orders) and the semantics (set inclusions). By considering interpretations at a syntactic level, concept lattices theory provide useful tools to build a vocabulary from examples.

4 Calculus

In this section, we discuss the various calculi proposed to compute entailment in conceptual and concept graphs. In the first subsection (4.1), we compare the derivation rules used as a sound and complete calculus for conceptual graphs [15] and concept graphs [18, 7]. Then (4.2) we compare their reformulation as a kind of graph homomorphism named projection [5, 18], and discuss the interests of this global operation for efficiency purpose. Finally (4.3), we discuss the normality requirement for a sound and complete projection mechanism, and the various methods proposed to ensure that any graph could be put into its normal form.

4.1 Elementary Generalization/Specialization Rules

To compute conceptual graphs entailment, [20] proposed a sound set of derivation rules that transform one graph into another one. This set of derivation rules has been corrected in [15] to achieve completeness w.r.t. CGs semantics. Similar sets of rules have been proposed in [18, 7] for concept graphs.

These sets of derivation rules respect the same behavior: let G be a conceptual or concept graph, and \mathcal{R} be a set of derivation rules. A CG G' is *immediately derived* from G in \mathcal{R} if G' is obtained by applying a rule of \mathcal{R} to G . A CG H is *derived* from G in \mathcal{R} if there is a sequence $G = G_0, G_1, \dots, G_k = H$ where, for $i = 1$ to k , G_i is immediately derived from G_{i-1} in \mathcal{R} . We note $G \vdash_{\mathcal{R}} H$.

Rules for Conceptual Graphs Two sets of rules have been proposed for CGs in [20]. The first set \mathcal{S} of rules, *specialization rules*, transforms a CG into a more specific one, *i.e.* $G \vdash_{\mathcal{S}} H$ iff $G, \mathcal{V} \models H$. The second set \mathcal{G} of generalization rules transforms a CG into a more general one *i.e.* $H \vdash_{\mathcal{G}} G$ iff $G, \mathcal{V} \models H$. We present here the sound and complete version of these rules proposed in [15].

Specialization rules

1. *Relation simplify:* If two relation nodes have the same label and the same ordered neighbours, delete one of them.
2. *Restrict:* Replace the label of any node by a more specific one.
3. *Join:* Merge two concept nodes having the same label.
4. *Disjoint sum:* Draw another CG next to the original one.
5. *Co-reference addition:* Merge two co-reference classes.
6. *Co-identical join:* Merge two concept-nodes that belong to the same co-reference class.

Generalization rules The set of generalization rules is obtained by building the inverse rules of the specialization rules presented above.

1. *Relation duplicate*: Duplicate a relation node (with the same ordered neighbors and the same label).
2. *Unrestrict*: Replace the label of a relation or a concept node by a more general one.
3. *Detach*: Split a concept node into two nodes of the same label. The union of their neighbors is the original set.
4. *Substract*: Delete a connected component.
5. *Co-reference deletion*: Split a co-reference class.
6. *Co-identical split*: Split a node into two co-referent ones. The union of their neighbors is the original set.

Rules for Concept Graphs The following set of rules in [7] correspond to the generalization rules of [15]. They update the rules in [18] to take co-reference into account. The twelve rules in [7] are named *Erasure*, *Iteration*., *Deiteration*, *Generalization*, *Isomorphism*, *Exchanging references*, *Merging two vertices*, *Splitting a vertex*, \top -*erasure*, \top -*insertion*, *Identify erasure* and *Identify deletion*.

Since we have proven in Sect. 3 that conceptual graphs and concept graphs have equivalent semantics, and since both generalization rules in [15] and [7] are sound and complete w.r.t. these equivalent semantics, it follows that these two sets of rules create the same graphs (up to the bijection in Sect. 2).

4.2 From Specialization Rules to Graph Homomorphism

Although with generalization/specialization rules we have a sound and complete calculus for CGs, the need for efficient algorithms led us to consider another operation: a graph homomorphism named projection [5]. We first show that the equivalence between specialization rules and graph homomorphism is a well-known characterization in graph theory. We then present two versions of projection, although the first does not require any normality condition, the second is more efficient.

Graph Theoretical Background Let us first consider the two specialization rules in [15] that have a direct impact on the structure of the graph, *Join* and *Disjoint Sum*. Without any consideration on the labels, using these rules consists in checking whether or not we can obtain a graph G from a graph H by making a disjoint sum of H and a graph D , then by merging its vertices, *i.e.* by checking whether or not a sequence of merges on H leads to a subgraph of G . This is a well-known characterization of the *graph homomorphism* problem, where the merges are usually called *retracts* [13].

Basically, a graph homomorphism is a mapping from the nodes of a graph into the nodes of another one that preserves neighborhood. This standard definition is easily updated to bipartite graphs (it must also preserve the bipartition) and to labels (it must preserve some order relation on the labels). The main difficulty in extending this definition to CGs is to take coreference into account.

CGs Homomorphism We present here an extension of the usual graph homomorphism (usually called *projection*) that takes the particular features of CGs specialization rules into account. This version does not require any normality condition, since it projects coreference classes into other coreference classes (instead of nodes into nodes). The following definition [6] translates the algorithm in [12].

Definition 19 (Coreference projection). *Let G and H be two CGs over a vocabulary \mathcal{V} . A coreference projection (or coref-projection) from H into G is a mapping Π from the coreference classes of H into the coreference classes of G such that:*

- *For each co-reference class C in H , let us consider the set of individual markers $I = \{i_1, \dots, i_k\}$ labelling the nodes of C . Then I is a subset of the individual markers of $\Pi(C)$.*
- *For each relation node r in H , with neighbors x_1, \dots, x_k and label t , let us consider C_i the coreference class of x_i , for $1 \leq i \leq k$. Then there is a relation node r' in G whose label is more specific than t and whose neighbors y_1, \dots, y_k are such that $y_i \in \Pi(C_i), 1 \leq i \leq k$.*

Theorem 2 (Soundness and completeness [6]). *Let G and H be two CGs over a vocabulary \mathcal{V} . Then H coref-projects into G iff $G, \mathcal{V} \models H$*

Generalization/specialization rules and *coref*-projection are thus two calculi for entailment of CGs. An immediate advantage of generalization/specialization rules is that they allow us to generate all CGs that are more general/specific than the given one. However, *coref*-projection is more efficient w.r.t. computing entailment between two CGs. Let us consider the difference between “compute a graph homomorphism from H into G ” and “check if a sequence of retracts of H generates a graph isomorphic to a subgraph of G ”. This simplification of our problem corresponds to the core, which is the NP-complete part of our entailment problem. Deciding on the existence of an homomorphism is an NP-complete problem, and efficient algorithms can be used (*e.g.* see [15, 2] for relationships with constraint networks). However, for specialization rules, even checking if a graph is isomorphic to another one is an ISO-complete problem (an intermediary class between P and NP), and this test must be done after applying each application of a rule. And even with a carefully written algorithm, there can be an exponential number of rule applications.

The need for a normal form The normal form of a CG is an artefact used to optimize *coref*-projection. It is neither a syntactic (see Sect. 2) nor a semantic (see Sect. 3) requirement of CGs. A CG is said in *normal form* if every coreference class contains a single node. If the graph G we look for a projection into is in normal form, *coref*-projection becomes the standard projection [16] (and also [18] for concept graphs), as expressed by the following definition. Since we only have to examine the edges incident to the current node, and not those incident to all the nodes belonging to the same coreference class, the calculus is more efficient.

Definition 20 (Projection). Let G and H be two CGs over a vocabulary \mathcal{V} . A projection from H into G is a mapping π from the nodes of H into the nodes of G such that:

- For each concept node $c \in C(H)$, the type of c is more general than the type of $\pi(c)$, and if the marker of c is individual, $\pi(c)$ has the same marker.
- For each coreferent concept nodes c, c' in H , $\pi(c) = \pi(c')$.
- For each relation node r in H , with neighbors x_1, \dots, x_k and label t , there is a relation node r' in G having a more specific type and whose neighbors are $\pi(x_1), \dots, \pi(x_k)$.

Since projection is equivalent to *coref*-projection when G is in normal form, the following theorem is a direct consequence of the previous one:

Theorem 3 (Soundness and completeness [20, 16]). Let G and H be two CGs over a vocabulary \mathcal{V} , G being in normal form. Then H projects into G iff $G, \mathcal{V} \models H$.

Note that, historically, projection was proposed in [5] without any normality condition. A counterexample was exhibited simultaneously in [16, 21], leading to two corrections: the normal form presented here, and the antinormal form [21] for the query, which is less efficient for computational purposes.

4.3 CGs and Normalization

Although projection is an interesting, efficient algorithm to compute entailment of CGs, it requires putting a CG into its normal form. This is done by merging all vertices that belong to the same coreference class. However, this is not always possible (what is the resulting type of the merge of two nodes having different types?). Different solutions to this problem have been proposed:

1. *Syntactic restrictions:* The *conformity relation* (assigning a type to each individual marker), as well as explicit restrictions on co-reference are used to force all vertices belonging to the same coreference class to have the same type (e.g. [16]). A weakness of this solution is to impose syntactic restrictions to solve calculus problems. It is of interest from a KR standpoint: the conformity relation defined in the vocabulary is a modelling guide.
2. *Semantic modifications:* When merging concept nodes having different types, the resulting type is their greatest common subtype. This *lattice-theoretic interpretation* changes the semantics of the vocabulary, and also imposes a syntactic restriction: the order on concept types must be a lattice (see CG-list: *CG: Individual Markers refer to unique entities?* for a discussion on this topic).
3. *Syntactic extensions:* Using type conjunction (e.g. [2, 6]) in CGs naturally solves this problem, but does not extend the expressivity of the language (we have seen in Sect. 3 that concept types and unary relations have the same interpretation, so a conjunction of concept types could be represented by multiple unary relations).

5 Conclusion

In this paper, we have compared simple conceptual graphs and simple concept graphs w.r.t. their syntax, semantics, and calculus.

Syntax As mathematical objects, we have proven that, up to superficial differences, they are identical objects (there are simple injective mappings from one class of objects to the other one).

Semantics Concerning the interpretation of CGs, power context families are not specific since they can define any ordered FOL structure. This shows that the (model-theoretical) semantics for conceptual graphs and for concept graphs are identical. Furthermore, power context families are not wholly used in the definition of the entailment relation, only the order relation between concepts (and relation-concepts) is used. Thus, in the development of a unified CG theory, we propose to use power context families only for the construction of vocabularies (i.e. ordered FOL languages).

Calculus If the aim is to build software tools in order to solve actual problems, one has to go beyond the decision problem of deduction and consider algorithms for constructing solutions and thus computational efficiency. This explains why, besides the interesting visual aspect of graphs in knowledge representation, we emphasize the graph viewpoint in dealing with CGs. Graph theory is a mature mathematical theory with many mathematical and algorithmic results that can be imported into CGs, especially the homomorphism (projection) notion, which is central in many computational and combinatorial problems (from graph coloring to category representation along with constraint satisfaction problems or query inclusion problems in relational databases).

Further works During our work on the different CGs semantics, it appeared that conceptual graphs canonical models [14] (or isomorphic interpretations [1, 2]) and concept graphs canonical models [18] (or Standard models in [7]) are similar notions. Altogether, they correspond to Herbrand models in FOL. For space requirements, this part of our work was not included in this paper, but will be developed later.

Finally, we have shown in this paper that power context families were a too expressive a structure for the reasonings involved in CGs. We intend to study if this conclusion is still valid in different extensions of conceptual and concept graphs (*e.g.* negation).

References

1. J.-F. Baget. *Représenter des connaissances et raisonner avec des hypergraphes: de la projection à la dérivation sous contraintes*. PhD thesis, Université Montpellier II, Nov. 2001.

2. J.-F. Baget. Simple conceptual graphs revisited: hypergraphs and conjunctive types for efficient projection algorithms. In de Moor et al. [9], pages 229–242.
3. J.-F. Baget and M.-L. Mugnier. The Complexity of Rules and Constraints. *JAIR*, 16:425–465, 2002.
4. Claude Berge. *Graphes et hypergraphes*. Dunod, 1970.
5. M. Chein and M.-L. Mugnier. Conceptual Graphs: Fundamental Notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.
6. M. Chein and M.-L. Mugnier. Types and Coreference in Simple Conceptual Graphs. In K.E. Wolff et al, editor, *Proc. ICCS'04*, volume 3127 of *LNAI*. Springer, 2004. to appear.
7. F. Dau. Concept graphs without negations: Standardmodels and standardgraphs. In de Moor et al. [9], pages 243–256.
8. Frithjof Dau. *The Logic System of Concept Graphs with Negations (And its Relationship to Predicate Logic)*, volume 2892 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2003. PhD-Thesis.
9. Aldo de Moor, Wilfried Lex, and Bernhard Ganter, editors. *Conceptual Structures for Knowledge Creation and Communication, 11th International Conference on Conceptual Structures, ICCS 2003 Dresden, Germany, July 21-25, 2003 Proceedings*, volume 2746 of *Lecture Notes in Computer Science*. Springer, 2003.
10. Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, 3 edition, 2000.
11. B. Ganter and R. Wille. *Formal Concept Analysis*. Springer-Verlag, 1999.
12. O. Guinaldo and O. Haemmerlé. Knowledge Querying in the Conceptual Graph Model: the RAP Module. In Mugnier and Chein [17], pages 287–294.
13. P. Hell and J. Nešetřil. *Graphs and Homomorphisms*, volume 121. Oxford University Press, 2004.
14. G. Kerdiles. *Saying it with Pictures: a logical landscape of conceptual graphs*. PhD thesis, Univ. Montpellier II / Amsterdam, Nov. 2001.
15. M.-L. Mugnier. Knowledge Representation and Reasoning based on Graph Homomorphism. In Bernhard Ganter and Guy W. Mineau, editors, *ICCS*, volume 1867 of *Lecture Notes in Computer Science*, pages 172–192. Springer, 2000.
16. M.-L. Mugnier and M. Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'Intelligence Artificielle*, 10(1):7–56, 1996. Available at <http://www.lirmm.fr/~mugnier/>.
17. Marie-Laure Mugnier and Michel Chein, editors. *Conceptual Structures: Theory, Tools and Applications, 6th International Conference on Conceptual Structures, ICCS '98, Montpellier, France, August 10-12, 1998, Proceedings*, volume 1453 of *Lecture Notes in Computer Science*. Springer, 1998.
18. S. Prediger. Simple concept graphs: A logic approach. In Mugnier and Chein [17], pages 225–239.
19. J. F. Sowa. Conceptual Graphs. *IBM Journal of Research and Development*, 1976.
20. J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.
21. M. Wermelinger. Conceptual Graphs and First-Order Logic. pages 323–337.
22. R. Wille. Conceptual graphs and formal context analysis. In Dickson Lukose, Harry S. Delugach, Mary Keeler, Leroy Searle, and John F. Sowa, editors, *ICCS*, volume 1257 of *Lecture Notes in Computer Science*, pages 290–303. Springer, 1997.