Basic Conceptual Structures Theory*

Michel Wermelinger José Gabriel Lopes

Departamento de Informática, Universidade Nova de Lisboa
2825 Monte da Caparica, PORTUGAL
E-mail: {mw,gpl}@fct.unl.pt

Abstract

Although the theory of Conceptual Structures is over 10 years old, basic notions (like canonical graphs) are far from settled and are subject to constant extensions and reformulations. However, most of these are done in an informal way, which doesn't help in clarifying the issues involved. It is our hope that this paper will provide a first step towards the complete and *rigorous* account of Conceptual Structures (CS) Theory, which is needed for ongoing standardization and implementation efforts.

Towards that goal, we present formal definitions of some of the central notions of CS theory (type, referent, concept, relation, conceptual graph, canonical formation rules, canon, and canonical graph) in its simplest form, i.e. no contexts nor coreference links are allowed and referents must be individuals. We thereby introduce higher-order types in order to enable the use of conceptual graphs at the metalevel, the restriction operation of the canonical formation rules is extended to make use of the relation hierarchy, we show the relationship between denotation and conformity relation, and we give a rigorous meaning to the canonical basis, among other things.

Key phrases: formalization of CS theory, higher-order concept and relation types, type and marker hierarchies, metalevel and instance level

1 Introduction

The "bible" of Conceptual Structures Theory is [16], which appeared over 10 years ago. As Sowa himself recognizes[1], [16] is written in a tutorial style, which means that several concepts introduced early in the text weren't updated in later sections. Furthermore, the informal and incomplete formulation of several definitions has led to many questions about the theory, even about some of its fundamental aspects, as the CG mailing list testifies.

Also, since its first appearance, the theory has undergone multiple changes and extensions due to the work of a growing scientific community. Although many of the concepts and notations introduced are motivated by specific application domains (like the analysis of tense in a discourse), several recent papers (like [2, 17, 19, 6, 18, 11, 15, 5, 14, 8]) deal with the abstract theory itself. Finally, the emergence of the ANSI IRDS standard [13], the KIF

*Slightly revised and corrected version of a paper that appeared in the Proceedings of the Second International Conference on Conceptual Structures, College Park MD, USA, 16–20 August 1994, Lecture Notes in Artificial Intelligence 835, Springer-Verlag.

[1]Unless otherwise stated, all personal opinions and statements were expressed in messages sent to the CG mailing list — send a message to `cg-request@cs.umn.edu` to subscribe it.

language [7], and the PEIRCE workbench [4] has made it clearer that it is about time to have a precise and complete definition of the core theory.

For all these reasons, we have proposed ourselves to give a formal account of the basic notions of Conceptual Structures Theory. It is our hope that this paper will clarify some issues, and provide a basis for the future standards' documentation, as well as serving as a guideline for implementors.

We use examples from published papers in order to show how the framework we define incorporates the informal notions presented in those papers. We assume the reader has been previously exposed to CS theory, e.g. as presented in [16] or [17]. Whenever we write Assumption (or Theorem or Definition) $x.y.z$ we are referring to [16].

1.1 Overview

The proposed framework, to be detailed in the remaining sections, can be briefly summarized as follows. There is a set of concept type lattices, one for each order. First-order types denote sets of individuals, while nth-order types represent sets of $(n-1)$th-order types. Furthermore there are relational concept lattices, one for each possible order. Intuitively, an nth-order relational concept type represents a set of nth-order relation types, and a nth-order relation type denotes a set of nth-order relations, which are relations having at least one argument which is a $(n-1)$th-order relation. In particular, first-order relations have as arguments only concepts.

As we want to use conceptual graphs as the meta-language, we must be able to talk about types as individuals. Therefore, for each concept type and for each relational concept type there will be a corresponding individual marker. Of course, there will be also individual markers that denote the individual objects of the domain of discourse. Adding a generic marker and an absurd marker, it is possible to obtain marker lattices. Having types and markers (also called referents), it is possible to define concepts, which are tuples consisting of a concept type and a referent, and relations, which are tuples consisting of a relation type and concepts (called the arguments of the relation). As types and referents are organized in lattices, concepts (and therefore relations) also form lattices.

This makes the formalism more regular and facilitates the definition of the canonical formation rules, which enable the derivation of new graphs from given ones. The canonical basis is a set of graphs which state for each relation what are its possible arguments. Therefore, canonical graphs (those derived from the canonical basis) are guaranteed to obey the selectional constraints.

1.2 Notation

We assume the reader is familiar with some of the usual mathematical terminology and symbology, especially regarding ordered sets. For good introductions see [16, Appendix A] and [3]. Some of the notational conventions used in this paper are:

- $\mathbb{N}$ is the set of natural numbers, which do not include zero;
- $\wp(S)$ denotes the powerset (i.e., the set of all subsets) of a set S;
- for any (bounded) lattice L, $top(L)$ designates the top element, $bottom(L)$ its least element, $x \wedge y$ the greatest lower bound of elements x and y, and $x \vee y$ their least upper

bound[2];

- for any partially ordered set S, the symbol $\leq$ designates the partial order, and the following equivalences apply: $x \leq y \Leftrightarrow y \geq x$, $x < y \Leftrightarrow x \leq y \wedge x \neq y$, $x > y \Leftrightarrow x \geq y \wedge x \neq y$;
- $t_1, \ldots, t_n < t'_1, \ldots, t'_m$ is a compact notation to state that for each possible combination of $i = 1, \ldots, n$ and $j = 1, \ldots, m$ one has $t_i < t'_j$.

In order to use few symbols and subscripts, we will overload functions when there is no possibility of confusion. For example $f : A \to B$ and $f : C \to D$ means that $f(x)$ returns an element of B if $x \in A$ and that $f(x) \in D$ if $x \in C$. We always guarantee that A and C are disjoint domains to avoid any misunderstanding. Unless otherwise stated, all functions that we will define are total.

2 Types and Individuals

In [17], Sowa shows the need for higher-order types, and notes that they can't be all put into the same hierarchy in order to avoid paradoxes. We therefore begin by generalizing the single (first-order) type hierarchy of Assumptions 3.2.3 and 3.2.5 to several hierarchies, each of a different order. As usual, we also assume the hierarchies to be lattices, which has computational advantages. Furthermore, as the type hierarchies will have to be specified by the user, we will assume finiteness.

Assumption 1. There is a finite set $\mathcal{T}_\mathcal{C} = \{T_1, \ldots, T_n\}$ of finite *concept type lattices.*

In the following, we will always use the variable n to denote the highest occurring order, i.e., $n = |\mathcal{T}_\mathcal{C}|$. Next we define the usual nomenclature for type hierarchies.

Definition 2. For each $i \in \{1, \ldots, n\}$

- T_i is the hierarchy of ith-order concept types;
- $top(T_i)$ is the ith-order universal (concept) type;
- $bottom(T_i)$ is the ith-order absurd (concept) type;

and for any $s, t, u \in T_i$

- if $s \leq t$ then s is called a subtype of t, and t is called a supertype of s;
- if $s < t$ then s is called a proper subtype of t, and t is called a proper supertype of s;
- if $s \leq t$ and $s \leq u$ then s is called a common subtype of t and u;
- if $s \geq t$ and $s \geq u$ then s is called a common supertype of t and u;
- if $s = (t \wedge u)$ then s is called the maximal common subtype of t and u;

[2]We also use $\wedge$ for the logical conjunction and $\vee$ for the disjunction, but the intended use is always clear from the context.

- if $s = (t \vee u)$ then s is called the minimal common supertype of t and u.

Example 1. According to the above, the examples of [17] are:

- $\top$ is the first-order universal type, i.e., $\top = top(T_1)$;
- $\bot$ is the first-order absurd type, i.e., $\bot = bottom(T_1)$;
- `TYPE` is the second-order universal type, i.e., `TYPE` $= top(T_2)$;
- `TYPE'` is the third-order universal type, i.e., `TYPE'` $= top(T_3)$;
- `GENUS, KINGDOM, SHAPE < TYPE` but `GENUS` is not a subtype of `KINGDOM`;
- `RANK, CHARACTERISTIC < TYPE'`.

Intuitively, an $(i+1)$th-order concept type enables us to talk about ith-order concept types. However, if we want to make statements about relations, we will have to "conceptualize" them, getting graphs like

Example 2 (adapted from [18]). `[RELATION]->(ATTR)->[TRANSITIVITY].`

This calls for further type hierarchies.

Assumption 3. There is a finite set $\mathcal{T}_{\mathcal{RC}} = \{R_1, \ldots, R_m\}$ of finite *relational concept type lattices*.

Again, in the following we will always use m to denote the highest occurring order of relational concept types, i.e. $m = |\mathcal{T}_{\mathcal{RC}}|$. We will furthermore adapt to relational concept types the nomenclature used for concept types.

Definition 4. For each $i \in \{1, \ldots, m\}$

- R_i is the hierarchy of ith-order relational (concept) types;
- $top(R_i)$ is the ith-order universal relational (concept) type;
- $bottom(R_i)$ is the ith-order absurd relational (concept) type.

Example 3. The first-order universal relational type might be called `RELATION` and R_1 might include the following types

- `MONADIC, DYADIC < RELATION`;
- `TRANSITIVE, REFLEXIVE, ANTI-SYM < DYADIC`;
- `PARTIAL-ORDER < TRANSITIVE, REFLEXIVE, ANTI-SYM`;

where `TRANSITIVE` might be defined by the graph in Example 2. Furthermore, notice that if we want to talk about second-order dyadic relations, we will have to use another type, say `DYADIC-2`, which will be a member of R_2.

Finally we need relation type hierarchies. The basic idea is to classify relation types according to their arity and order. A relation of arity i (also called an i-adic relation) is a relation with exactly i arguments. Similar to higher-order functions, an $(i+1)$th-order relation is a relation that has at least one argument which is an ith-order relational concept. This means that there must be $m+1$ relation type orders, since there are m relational concept type orders.

According to Assumptions 3.2.7 and 3.6.13 *all* (first-order) relations are in a single hierarchy whose top element is the dyadic relation `LINK`. However, in the rest of [16] the relation hierarchy is never put to use. Moreover, the exact meaning of something like `BETW` $<$ `LINK`, where `BETW` is triadic while `LINK` is dyadic, remains unclear. This explains that (to our knowledge) no one has ever precisely defined the concept of "relation restriction" — even [14] doesn't deal with the arity issue — although it is often said that `LINK` may be restricted to any other relation type.

For these reasons we don't put all relations of the same order into the same hierarchy. Instead, each hierarchy contains all relations with the same *signature* (i.e. order, arity, and arguments' orders). Therefore, relation r can be a subrelation of relation r' (written $r \leq r'$) only if r and r' have the same signature.

Assumption 5. There is a finite set $\mathcal{T}_{\mathcal{R}}$ of finite *relation type lattices* $R_{\langle x_1,\ldots,x_d\rangle}$ such that for each lattice $d \geq 1$ and the *signature* $\langle x_1,\ldots,x_d\rangle$ is unique, where $x_i \in \mathcal{T}_{\mathcal{C}} \cup \mathcal{T}_{\mathcal{RC}}$ for each $i \in \{1,\ldots,d\}$. Inversely, each element of $\mathcal{T}_{\mathcal{C}} \cup \mathcal{T}_{\mathcal{RC}}$ must occur in the signature of at least one relation type lattice.

The last restriction ensures that every imaginable (relational) concept can be linked to at least one relation in a conceptual graph (see Section 3). Next we define the order of a relation as the successor of the maximal order of its relational arguments.

Definition 6. The function $order : \bigcup R_{\langle x_1,\ldots,x_d\rangle} \rightarrow \mathbb{N}$ that returns for each relation type its *order* is defined as

$$order(r) = 1 + max(\{j | \exists i \in \{1,\ldots,d\} \text{ such that } x_i = R_j\})$$

with the understanding that $max(\{\}) = 0$. The set of all relation types of order k will be written as R^k.

Due to the last requirement of Assumption 5, it is possible to write the set of all relation types as $\bigcup_{k=1}^{m+1} R^k$, since the existence of mth-order relational concept types implies the existence of $(m+1)$th-order relations.

Notice that R^i and R_i mean different things. The former is the set of all ith-order relation types, while the latter is the lattice of all ith-order relational concept types. As expected, there is a relationship between them (see Assumption 10).

Definition 7. The function $degree : \bigcup R_{\langle x_1,\ldots,x_d\rangle} \rightarrow \mathbb{N}$ that returns for each relation type its *degree* (also called *arity*) is defined as

$$degree(r) = d \iff r \in R_{\langle x_1,\ldots,x_d\rangle}$$

If a relation has arity d, it is said to be d-adic. 1-adic, 2-adic, and 3-adic relations are respectively called monadic, dyadic, and triadic.

The formal notation may seem rather confusing, but is in fact quite intuitive. Let us see some examples.

Example 4. Of all elements of $\mathcal{T}_{\mathcal{R}}$, the lattice $R_{\langle T_1,T_1\rangle}$ is probably the largest one, as it includes the relation types that most frequently occur in Conceptual Graphs papers, namely those of first-order dyadic relations between first-order concept types. For example, most of the relations of the Conceptual Catalog in [16, Appendix B.3], like `AGNT`, `LOC`, `OBJ`, and `PART`, belong to that hierarchy and `LINK` is its top element.

Example 5. The lattice $R_{\langle T_1,T_2\rangle}$ of all relations whose first argument is a first-order concept and whose second argument is a second-order concept includes both `KIND` and `CHRC` as used in [17]. Notice that although `CHRC` is defined in terms of `KIND`, one has `KIND < CHRC` because `KIND` denotes a special characteristic of an individual: its type.

Example 6. A typical member of $R_{\langle R_1,R_1\rangle}$ is `INVERSE-OF`, a second-order relation between two first-order relations.

Example 7. The relation `ATTR` that appeared in Example 2 should not be confused with the one that appears in `[BALL]->(ATTR)->[RED]`! The latter relates two first-order concepts while the former involves a relation (more precisely, a relational concept). Therefore, the latter is an element of $R_{\langle T_1,T_1\rangle}$ while the relation of Example 2 belongs to $R_{\langle R_1,T_1\rangle}$. To distinguish such cases, we will use labels like `ATTR-T1-T1` and `ATTR-R1-T1`.

It is obvious that no type can be of two different orders, nor can it be simultaneously of two different kinds (e.g. a concept type and a relation type).

Assumption 8. $\forall L, L' \in \mathcal{T}_{\mathcal{C}} \cup \mathcal{T}_{\mathcal{R}} \cup \mathcal{T}_{\mathcal{RC}}\ L \neq L' \Rightarrow L \cap L' = \emptyset$.

Now that we have all types we need, we can give them a semantics. For that purpose, a domain is needed.

Assumption 9. The *universe of discourse* (also called *domain*) is $\mathcal{U} = I \cup T_1 \cup \ldots \cup T_{n-1} \cup \bigcup_{k=1}^{m} R^k$, where I is the set of individuals.

The interpretation of types will be a generalization of the first-order case presented in Definition 3.2.2 and Assumption 3.2.4. Following the informal guideline presented in [17], the denotation of an $(i+1)$th-order concept type is a subset of the ith-order concept types, whereas the denotation of a relation type is a subset of the tuples formed by the denotation of their arguments. In both cases the hierarchical relations must be preserved. Moreover, the denotation of absurd types is the empty set, and universal types represent all elements of the next lower order.

Assumption 10. The *denotation* (or *interpretation*) function δ assigns to each (relational) concept type a subset of the universe of discourse and to each relation type a set of tuples of domain elements, subject to the following conditions:

- $\delta : T_1 \rightarrow \wp(I)$
- $\delta : T_i \rightarrow \wp(T_{i-1})$ for $i = 2, \ldots, n$

- $\delta : R_i \to \wp(R^i)$ for $i = 1, \ldots, m$
- $\delta : R_{\langle x_1,\ldots,x_d\rangle} \to \wp(\delta(top(x_1)) \times \ldots \times \delta(top(x_d)))$
- for each lattice $L \in \mathcal{T_C} \cup \mathcal{T_R} \cup \mathcal{T_{RC}}$
  - $\delta(bottom(L)) = \emptyset$
  - if $\delta : L \to \wp(S)$ then $\delta(top(L)) = S$
  - if $x \leq y$ in L, then $\delta(x) \subseteq \delta(y)$

Example 8. Using Examples 1, 3, 4, 5, and 6 one gets:

- $\delta(\top) = I$ and $\delta(\bot) = \emptyset$
- $\delta(\texttt{TYPE}) = T_1 = \{\top, \bot, \texttt{PERSON}, \texttt{CAT}, \ldots\}$
- $\delta(\texttt{TYPE'}) = T_2 = \{\texttt{TYPE}, \texttt{GENUS}, \texttt{KINGDOM}, \texttt{SHAPE}, \ldots\}$
- $\texttt{KINGDOM} \in \delta(\texttt{RANK})$, $\texttt{SHAPE} \in \delta(\texttt{CHARACTERISTIC})$
- $\texttt{RANK < TYPE'}$ implies $\delta(\texttt{RANK}) = \{\texttt{KINGDOM},\texttt{GENUS},\texttt{SPECIES},\ldots\} \subseteq \delta(\texttt{TYPE'})$
- $\texttt{KIND} \in R_{\langle T_1,T_2\rangle}$ implies that $\delta(\texttt{KIND}) \in \wp(\delta(top(T_1)) \times \delta(top(T_2)))$ or, more simply, $\delta(\texttt{KIND}) \subseteq \delta(\top) \times \delta(\texttt{TYPE}) = I \times T_1$
- $\delta(\texttt{DYADIC}) = \bigcup_{x_1,x_2 \in \mathcal{T_C}} R_{\langle x_1,x_2\rangle} = \{\texttt{LINK}, \texttt{KIND}, \ldots\} \subseteq R^1$
- $\texttt{INVERSE-OF} \in \delta(\texttt{DYADIC-2})$

One might be tempted to include the following seemingly intuitive restriction in Assumption 10: for any (relational) concept types x, y, and z, if $x \in \delta(z)$ and $y \leq x$ then $y \in \delta(z)$. However, [17] presents a counter-example: $\texttt{FELIS} \in \delta(\texttt{GENUS})$ and $\texttt{FELIS-CATUS < FELIS}$ but $\texttt{FELIS-CATUS} \notin \delta(\texttt{GENUS})$ since FELIS-CATUS is a species.

The following theorem generalizes Theorem 3.2.6 for all type hierarchies.

Theorem 11. *For any $L \in \mathcal{T_C} \cup \mathcal{T_R} \cup \mathcal{T_{RC}}$ and any $x, y \in L$ the following holds:*

- $\delta(x \wedge y) \subseteq \delta(x) \cap \delta(y)$
- $\delta(x) \cup \delta(y) \subseteq \delta(x \vee y)$

Proof. Using Assumption 10 and some well-known lattice properties, one gets

- $(x \wedge y) \leq x \Rightarrow \delta(x \wedge y) \subseteq \delta(x)$ and $(x \wedge y) \leq y \Rightarrow \delta(x \wedge y) \subseteq \delta(y)$
- $x \leq (x \vee y) \Rightarrow \delta(x) \subseteq \delta(x \vee y)$ and $y \leq (x \vee y) \Rightarrow \delta(y) \subseteq \delta(x \vee y)$

which imply the theorem's statements. □

2.1 Individuals

The denotation of some type is a set of other types treated as individuals. To be able to use Conceptual Graphs as a meta-language, we must provide for individual markers that correspond to the individuals in the universe of discourse. Therefore, we must extend the *single* set of individual markers of Assumption 3.3.1 to *multiple* sets, one for each different order. Furthermore, the individual markers should reflect the ordering of the types they correspond to. For this purpose we will refine the τ and ρ operators that were presented informally in [18] to map the meta-level to the instance level.

Assumption 12. There is a set $I_C = \{IC_1, \ldots, IC_n\}$ of *individual concept marker* sets such that IC_1 is finite, there is an injective function $\tau : IC_1 \to I$, and for each $i \in \{2, \ldots, n\}$ there is an isomorphism $\tau : IC_i \to T_{i-1}$.

Notice that due to the isomorphism and to Assumption 1, IC_i is a finite lattice for $i \neq 1$. Moreover, any set can be viewed as a partially ordered set with the antichain order ($x \leq y \Leftrightarrow x = y$). This means that all members of I_C are finite posets, a necessary condition for the order-embedding of Assumption 15. Furthermore, an isomorphism is an injective function. This implies that all τ mappings are one-to-one, which is equivalent to the commonly used Unique Name Assumption (i.e., different markers represent different individuals of the domain).

Example 9. Assumption 3.3.1 states that the elements of IC_1 are `#1`, `#2`, ... However, since we provide several marker sets, and according to general usage, we will use more readable labels, like `#John`, `#Moby-Dick`, etc.

Example 10. In an actual implementation, there may be a default mapping `#`$x \to X$ to spare the user from tedious typing. For example τ(`#person`) = `PERSON`, where `#person` $\in IC_2$ because `PERSON` $\in T_1$.

Since the domain includes relations, we must also provide individual markers for them. They will be used in the referent field of relational concepts, which implies that there can only be m sets of such markers, not $m + 1$.

Assumption 13. There is a set $I_R = \{IR_1, \ldots, IR_m\}$ of *individual relation marker* sets such that for each $i \in \{1, \ldots, m\}$ there is an isomorphism $\rho : IR_i \to R^i$.

Notice that each IR_i is a finite poset whose top elements are the markers that correspond to the top elements of the ith-order relation type hierarchies.

Example 11. According to Examples 4 and 5 one has in IR_1 both `#agnt < #link` and `#kind < #chrc`, but `#kind` is unrelated to `#link`, which is one of the top elements of IR_1.

Example 12 (adapted from [17]). An implemented system may provide a default mapping which can be overruled by the user. For instance, ρ(`#greater-than`) = `GREATER-THAN` unless the user explicitly states that ρ(`#greater-than`) = `>`.

Now that the whole domain is covered, we introduce as in [11] an absurd marker which together with the usual generic marker enables us to classify the markers into lattices, as proposed in [15].

Assumption 14. There are two special markers called the *generic marker* (written $*$) and the *absurd marker* (written $\bar{*}$).

Assumption 15. There is a set of *markers* $\mathcal{M} = CM_1 \cup \ldots \cup CM_n \cup RM_1 \cup \ldots \cup RM_m$ such that for each $i \in \{1, \ldots, n\}$ and each $j \in \{1, \ldots, m\}$

- $CM_i = IC_i \cup \{*, \bar{*}\}$ and $RM_j = IR_j \cup \{*, \bar{*}\}$ are lattices;
- $top(CM_i) = top(RM_j) = *$ and $bottom(CM_i) = bottom(RM_j) = \bar{*}$;
- IC_i is embedded in CM_i and IR_j is embedded in RM_j.

Notice that $\mathcal{M} = \bigcup_{i=1}^{n} IC_i \cup \bigcup_{j=1}^{m} IR_j \cup \{*, \bar{*}\}$. Furthermore, since each IC_i is a finite lattice for $i > 1$, it is bounded and therefore has a top and a bottom element. This means that the only marker directly beneath the generic marker in CM_i is $top(IC_i)$ and conversely $bottom(IC_i)$ is the only marker above the absurd marker.

The conformity relation presented in Assumption 3.3.3 states which markers may be combined with which concept types. However, Assumption 3.3.3 is self-contradictory if taken literally. It states that (1) no individual marker may conform to the absurd type, and (2) if a marker conforms to two different types, it must conform to their maximal common subtype. We get a contradiction when the maximal common subtype is the absurd type. An obvious solution is to require as in [2] that no individual marker may conform to incompatible types. Our approach will be not to assume (2).

In our formalization, we make it clear that the conformity relation is just the explicitation of the denotation function. In fact, the assertion "`#John` conforms to `PERSON`" is equivalent to "John is a person" which formally means that the individual corresponding to the name John is a member of the denotation of the type `PERSON`. Having this relationship enables us to define the conformity relation in a much more compact, elegant, and general way than Assumption 3.3.3.

Assumption 16. The *conformity relation* $::$ between (relational) concept types and markers is such that

$$\forall t \in \bigcup_{i=1}^{n} T_i \cup \bigcup_{i=1}^{m} R_i \;\; \forall m \in \mathcal{M} \;\; t :: m \Leftrightarrow m = * \vee \tau(m) \in \delta(t) \vee \rho(m) \in \delta(t)$$

Assumption 3.3.3 states that if a marker conforms to some type t, it must conform to all of t's supertypes. With the new definition, this becomes a theorem.

Theorem 17. $\forall x, y \in \bigcup_{i=1}^{n} T_i \cup \bigcup_{i=1}^{m} R_i \;\; \forall m \in \mathcal{M} \;\; x :: m \wedge y \geq x \Rightarrow y :: m.$

Proof. If m is the generic marker, the theorem is trivially true. Otherwise, $x :: m$ means that $\tau(m) \in \delta(x) \vee \rho(m) \in \delta(x)$ by Assumption 16. Together with $x \leq y \Rightarrow \delta(x) \subseteq \delta(y)$ (Assumption 10) this implies $\tau(m) \in \delta(y) \vee \rho(m) \in \delta(y)$ which is equivalent to $y :: m$, as pretended. □

Assumption 3.3.3 also requires that no individual marker may conform to the absurd type. Again, our formulation implies and extends that constraint.

Theorem 18. *The absurd marker does not conform to any (relational) concept type, and an individual marker doesn't conform to any absurd (relational) concept type.*

Proof. The theorem states that $t :: m$ is false if m is the absurd marker or if t is an absurd type and m is an individual marker. In the first case, $m \neq *$ and $\tau(m)$ and $\rho(m)$ aren't defined (Assumptions 12 and 13). In the second case, $m \neq *$ and the denotation of t is empty (Assumption 10). So in both cases the conditions of Assumption 16 are false, which proves the theorem. □

However, the most important difference between our Assumption 16 and Assumption 3.3.3 is that we don't require $t :: m$ and $t' :: m$ to imply $(t \wedge t') :: m$. The reason is that it would amount to say that $\delta(t \wedge t') = \delta(t) \cap \delta(t')$. This is what is called the lattice-theoretic interpretation of a type hierarchy [1]. Intuitively, it means that for every pair of compatible types their intersection must be represented by an explicit type, even if it is not conceptually relevant. Therefore, the order-theoretic interpretation given by Theorem 11 is more appropriate for AI applications, as the next example shows.

Example 13 (adapted from [1]). Assume that historic landmarks are the maximal common subtype of churches and old buildings:

$$\texttt{HISTORIC-LANDMARK} = \texttt{CHURCH} \wedge \texttt{OLD-BUILDING}$$

This does not mean that every old church is necessarily an historic landmark. So it is possible to have $\delta(\texttt{CHURCH})$ = {St. Peter, St. Mary, St. Paul}, $\delta(\texttt{OLD-BUILDING})$ = {St. Peter, St. Mary, Town Hall}, but just $\delta(\texttt{HISTORIC-LANDMARK})$ = {St. Peter}. In other words, although `CHURCH::#st-mary` and `OLD-BUILDING::#st-mary`, we *don't* have `HISTORIC-LANDMARK::#st-mary`.

3 Concepts, Relations, and Conceptual Graphs

Having defined types and individuals, it is possible to have concepts and relations. Basically, a (relational) concept is a combination of an ith-order (relational) concept type with a (relation) marker denoting an $(i-1)$th-order individual.

Definition 19. The set of all *concepts* is $\mathcal{C} = C_1 \cup \ldots \cup C_n \cup RC_1 \cup \ldots \cup RC_m$ where $C_i = T_i \times CM_i$ for each $i \in \{1, \ldots, n\}$ and $RC_j = R_j \times RM_j$ for each $j \in \{1, \ldots, m\}$.

Relations are also tuples, such that a relation's arguments are concepts whose types conform to the signature given by the relation's type.

Definition 20. The set of all *relations* is $\mathcal{R} = \{\langle l, a_1, \ldots, a_d\rangle | l \in R_{\langle x_1,\ldots,x_d\rangle}\ \forall i \in \{1, \ldots, d\}\ (x_i = T_j \Rightarrow a_i \in C_j) \wedge (x_i = R_j \Rightarrow a_i \in RC_j)\}$.

Now it is possible to define the *type* and *referent* functions of Assumptions 3.2.1, 3.2.7 and 3.3.1.

Definition 21. The function *type* returns for each concept or relation its type:

- $type : \mathcal{C} \to \bigcup_{i=1}^{n} T_i \cup \bigcup_{i=1}^{m} R_i$ is defined as $type(c) = t \Leftrightarrow c = \langle t, m\rangle$;

- $type : \mathcal{R} \to \bigcup_{k=1}^{m+1} R^k$ is defined as $type(r) = t \Leftrightarrow r = \langle t, a_1, \ldots, a_d \rangle$.

Definition 22. The function $referent : \mathcal{C} \to \mathcal{M}$ returns for each concept its marker: $referent(c) = m \Leftrightarrow c = \langle t, m \rangle$.

Notice that we could have defined first the type function just for concepts and then $\mathcal{R}$ as $\{\langle l, a_1, \ldots, a_d \rangle | l \in R_{\langle x_1, \ldots, x_d \rangle} \ \forall i \in \{1, \ldots, d\} \ a_i \in \mathcal{C} \wedge type(a_i) \in x_i\}$. The next definition will be useful for the rest of the paper.

Definition 23. The partial function $arg : \mathcal{R} \times \mathbb{N} \to \mathcal{C}$ returns for a given relation and a given natural number i the ith argument of the relation:

$$arg(r, i) = c \Leftrightarrow r = \langle t, a_1, \ldots, a_i, \ldots, a_d \rangle \wedge c = a_i$$

The function $args : \mathcal{R} \to \wp(\mathcal{C})$ assigns to each relation its arguments:

$$args(r) = \bigcup_{i=1}^{degree(type(r))} arg(r, i)$$

The ordering of types and referents induces an ordering over concepts, and therefore over relations.

Definition 24. A concept c is a *restriction* (or *specialization*) of concept c' and conversely c' is a *generalization* of c (written $c \leq c'$) if and only if $type(c) \leq type(c')$ and $referent(c) \leq referent(c')$.

Definition 25. A relation r is a *restriction* (or *specialization*) of relation r' and conversely r' is a *generalization* of r (written $r \leq r'$) if and only if $type(r) \leq type(r')$ and $arg(r, i) \leq arg(r', i)$ for every $i \in \{1, \ldots, degree(type(r))\}$.

Proposition 26. *Both $\mathcal{C}$ and $\mathcal{R}$ are partitions of lattices.*

Proof. Each C_i and each RC_j is a product of lattices, hence a lattice (whose partial order is the specialization relation among concepts). Furthermore those lattices are disjoint by construction. A similar reasoning applies to $\mathcal{R}$. □

A simple conceptual graph (i.e. without contexts or coreference links) is defined as in Assumption 3.1.2.

Assumption 27. A *simple conceptual graph* is a finite, bipartite, connected graph $\langle V_C, V_R, E \rangle$ such that V_C is a non-empty bag of concepts, V_R is a bag of relations, and E is a subset of $V_C \times V_R$ satisfying

$$\forall c \in V_C \ \forall r \in V_R \ \langle c, r \rangle \in E \Leftrightarrow \exists i \in \mathbb{N} \ arg(r, i) = c$$

Notice that if V_R is empty, then V_C contains a single concept; otherwise the graph wouldn't be connected.

4 Canonical Graphs

A conceptual graph only imposes a minimum of meaningfulness. It is always well-formed in the sense that the concepts linked to a relation must conform to the relation's signature, but there are no other conditions. In order to restrict the possible combinations of concepts and relations, canonical graphs will be defined as being graphs that obey certain selectional constraints. Towards that goal, we start with the definition of canonical formation rules, which are identical to Assumption 3.4.3 except that it is now possible to restrict relation types. For a more detailed study of the canonical formation rules, especially the join operation, the reader is referred to [2].

Assumption 28. The *canonical formation rules* allow the transformation of the not necessarily different conceptual graphs u and v into a conceptual graph w:

copy w is an exact copy of u;

simplify if two relations are equal, then one of them may be removed from the graph;

restrict a concept c may be replaced by a concept c' if $c' < c$ and $type(c') :: referent(c')$, and a relation r may be replaced by relation r' if $type(r') < type(r)$ and $\forall i \in \{1, \ldots, degree(type(r))\}\ arg(r, i) = arg(r', i)$;

join if concept c of graph u is equal to concept c' in graph v, then w is obtained by taking the union of all vertices and edges of u and v and replacing all occurrences of c' by c.

Using these rules compositionally, it is possible to derive a conceptual graph u from a set of conceptual graphs S. However, for the definition of canonical graph (Assumption 36) it is important to guarantee that *every* graph of S is used in the derivation of u. For that purpose, we adopt the notion of derivation from [11] but we call it "canonical derivation", a term that is used in [11] to denote the derivation from a specific set (the canonical basis). We feel that our nomenclature is more in accordance with [16] and the common usage in the Conceptual Graphs community.

Definition 29. A *canonical derivation* of a conceptual graph G is a directed acyclic graph whose nodes are conceptual graphs such that

- G is the only sink;
- any node is either a source or it has exactly one or two predecessors;
- if node v has exactly one predecessor v' then v is a simplification or a restriction of v';
- if node v has exactly two predecessors v' and v'' then v is a join of v' and v''.

Definition 30. A conceptual graph u is *canonically derivable* from a set S of conceptual graphs if there is a canonical derivation of u whose sources are elements of S.

The canonical formation rules induce a further structuring mechanism, this time between conceptual graphs. Notice that Theorem 3.5.2 states that the relation between graphs is anti-symmetric, which is not true. In fact it is possible for two *different* graphs to be related to each other as Gerard Ellis and others [2] pointed out. Thus the notion of "equivalent graph" is necessary.

Definition 31. A conceptual graph u is a *specialization* of a conceptual graph v and conversely v is a *generalization* of u (written $u \leq v$) if and only if there is a derivation of u containing v.

Definition 32. Two graphs are *equivalent* if each one is a specialization of the other.

Definition 33. The *generalization hierarchy* is the poset of equivalence classes of conceptual graphs induced by the generalization relation.

Definition 34. Two graphs are *comparable* if one of them is a specialization of the other.

In [16] the canonical basis is defined as the initial set of conceptual graphs to which the canonical formation rules may be applied. All graphs thus obtained are called canonical graphs. Furthermore, in the Conceptual Catalog [16, Appendix B] a canonical graph is assigned to each concept and relation type, but this is not part of the formal definition of a canonical basis (Assumption 3.4.5). This led the Conceptual Structures community to use the term "canonical graph" in two different senses: (1) a graph that is derivable from the canonical basis, and (2) *the* graph that is associated to some type in the canonical basis. Of course, these two senses are not incompatible, since (2) implies (1). However, to make the distinction clear we will use the term "base graph" for sense (2).[3]

There is another dual view as Mark Willems pointed out. On one hand base graphs represent selectional constraints on the links between relations and concepts. On the other hand they state the mandatory "arguments" of each type (e.g. the concept type `GIVE` should involve two `PERSON`s and an `OBJECT`). We feel that this latter view is more appropriate of a lexicon.

We therefore use base graphs only for selectional restrictions. As concepts may be linked to many different relations, we take an approach similar to [2, 11] and only impose constraints on the relations. In other words, for each relation type there is at most one base graph associated with it, stating the maximal concepts that may be attached to it. Furthermore, if a relation r is a subrelation of r', then their base graphs must be related, as r can't be attached to more general concepts than r'.

Also, if an individual marker m conforms to type t, it makes no sense to say that the graph consisting only of concept $\langle t, m\rangle$ is non-canonical. By the canonical formation rules, $\langle t, m\rangle$ is derivable from $\langle t', *\rangle$ where t' is the top element of the type hierarchy to which t belongs. Therefore, all concept types of the same hierarchy will be associated to the same base graph.

Assumption 35. A *canon* is a tuple $\langle \mathcal{T}_\mathcal{C}, \mathcal{T}_\mathcal{RC}, \mathcal{T}_\mathcal{R}, \mathcal{M}, \tau, \rho, ::, \mathcal{B}, \gamma\rangle$ such that the first seven elements satisfy all the assumptions given before, the *canonical basis* $\mathcal{B}$ is a finite set of conceptual graphs (formed from the types and individuals given by the canon), and the surjective function $\gamma : \bigcup_{i=1}^{n} T_i \cup \bigcup_{j=1}^{m} R_j \cup \bigcup_{k=1}^{m+1} R^k \to \mathcal{B}$ associates to each type its *base graph*. Furthermore, the following conditions must be met:

- any concept occurring in the graphs of $\mathcal{B}$ satisfies $::$;
- each graph of $\mathcal{B}$ has at most one relation, and each relation type occurs at most once in the graphs of $\mathcal{B}$;

[3] The "star graphs" of [11] are a special case of our base graphs, namely those associated to relation types.

- for any relation type t, the graph $\gamma(t)$ has a relation r such that $type(r) \geq t$;
- $\forall L \in \mathcal{T_C} \cup \mathcal{T_{RC}}\ \forall t \in L\ \ \gamma(t) = \langle \{\langle top(L), * \rangle\}, \emptyset, \emptyset \rangle$;
- $\forall r, r' \in \bigcup_{k=1}^{m+1} R^k\ \ r \leq r' \Rightarrow \gamma(r) \leq \gamma(r')$.

Finally, we can define a canonical graph g as a graph that has been derived from all the base graphs of the concepts and relations that appear in g.

Assumption 36. A conceptual graph u is *canonical* with respect to a given canon $\langle \mathcal{T_C}, \mathcal{T_{RC}}, \mathcal{T_R}, \mathcal{M}, \tau, \rho, ::, \mathcal{B}, \gamma \rangle$ if there is a derivation for u whose sources S are members of $\mathcal{B}$ satisfying the following conditions:

- for each relation r of u, $\gamma(t) \in S$ for every $t \geq type(r)$;
- for each concept c of u, $\gamma(type(c)) \in S$;
- no other graph belongs to S.

It should be obvious that

Proposition 37. *Given a canon C with canonical basis $\mathcal{B}$, every graph of $\mathcal{B}$ is canonical with respect to C.*

5 Conclusions

In our opinion, there are several strong reasons for having a formal specification of Conceptual Structures Theory, and therefore this paper presented a precise account of its basic notions, especially regarding higher-order types, as they enable conceptual graphs to be used as the meta-language. The proposed formalization, which incorporates some of the recently published (informal) ideas, is highly structured: types, referents, and therefore concepts and relations, are all organized into lattices. This makes the theory simpler, more regular, and more elegant. It also allowed us to extend the restriction operation to relation types. Furthermore, our formalization clarifies several relationships, as between referents and types, and between the denotation function and the conformity relation. Finally, the meaning of canonical graph as a selectional constraint has been precisely defined.

The basic notions presented in this paper will be implemented in the second version of the Conceptual Graph Tools [21]. Furthermore, they can be combined with a mechanism to structure knowledge bases as presented in [20].

Acknowledgements

We would like to thank Luís Caires, Margarida Mamede, and an anonymous referee for several useful comments and suggestions.

References

[1] C. Beierle, U. Hedstück, U. Pletat, P. H. Schmitt, and J. Siekmann. An order-sorted logic for knowledge representation systems. Technical Report 113, IWBS, April 1990.

[2] Michel Chein and Marie-Laure Mugnier. Conceptual graphs: fundamental notions. *Révue d'Intelligence Artificielle*, 6(4):365–406, 1992.

[3] B. A. Davey and H. A. Priestley. *Introduction to Order and Lattices*. Cambridge University Press, 1990.

[4] Gerard Ellis and Robert A. Levinson, editors. *Proceedings of the First International Workshop on PEIRCE: A Conceptual Graphs Workbench*, Las Cruces, New Mexico, 10 July 1992. University of Queensland Technical Report 241.

[5] John W. Esch. Contexts as white box concepts. In Mineau et al. [10], pages 17–29.

[6] David A. Gardiner, Bosco S. Tjan, and James R. Slagle. Extending conceptual structures: Representation issues and reasoning operations. In Nagle et al. [12], pages 67–85.

[7] Michael R. Genesereth and Richard E. Fikes. Knowledge interchange format version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, June 1992. "Living document" of the Interlingua Working Group of the DARPA Knowledge Sharing Effort.

[8] Bikash Chandra Ghosh and Vilas Wuwongse. Declarative semantics of conceptual graph semantics. In Robert Levinson and Gerard Ellis, editors, *Proceedings of the Second International Workshop on PEIRCE: A Conceptual Graphs Workbench*, Quebec, Canada, 7 August 1993. Laval University.

[9] Guy W. Mineau, Bernard Moulin, and John F. Sowa, editors. *Conceptual Graphs for Knowledge Representation*, number 699 in Lecture Notes in Artificial Intelligence, Québec City, Canada, 4–7 August 1993. Springer-Verlag. Proceedings of the First International Conference on Conceptual Structures.

[10] Guy W. Mineau, Bernard Moulin, and John F. Sowa, editors. *International Conference on Conceptual Structures: Theory and Applications*, Québec City, Canada, 4–7 August 1993. Complementary proceedings.

[11] M.L. Mugnier and M. Chein. Characterization and algorithmic recognition of canonical conceptual graphs. In Mineau et al. [9], pages 294–311.

[12] Timothy E. Nagle, Janice A. Nagle, Laurie L. Gerholz, and Peter W. Eklund, editors. *Conceptual Structures: Current Research and Practice*. Ellis Horwood Series in Workshops. Ellis Horwood, 1992.

[13] S. Perez and A. Sarris, editors. Information resource dictionary system conceptual schema. Technical Report X3H4/92-003 and ISO/IEC JTC1/SC21 N7486, American National Standards Institute and International Organisation for Standardization, 1993.

[14] Gérard Sabah and Anne Vilnat. Hierarchy of relational types in conceptual graphs to handle natural language parsing. In Mineau et al. [10], pages 198–215.

[15] Jan Schmidt and Pavel Kocura. Generalized referents: a neat interface for the scruffy work. In Mineau et al. [10], pages 1–16.

[16] John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. The System Programming Series. Addison-Wesley Publishing Company, 1984.

[17] John F. Sowa. Conceptual graph summary. In Nagle et al. [12], pages 3–51.

[18] John F. Sowa. Relating diagrams to logic. In Mineau et al. [9], pages 1–35.

[19] Bosco S. Tjan, David A. Gardiner, and James R. Slagle. Representing and reasoning with set referents and numerical quantifiers. In Nagle et al. [12], pages 53–66.

[20] Michel Wermelinger and Alex Bejan. Conceptual structures for modeling in CIM. In Mineau et al. [9], pages 345–360.

[21] Michel Wermelinger and José Gabriel Lopes. An X-Windows toolkit for knowledge acquisition and representation based on Conceptual Structures. In Heather D. Pfeiffer and Timothy E. Nagle, editors, *Conceptual Structures: Theory and Implementation*, number 754 in Lecture Notes in Artificial Intelligence, pages 262–271. Springer Verlag, 1993. Proceedings of the Seventh Annual Workshop on Conceptual Graphs, Las Cruces, New Mexico, USA, 8–10 July 1992.