Conceptual graphs and first-order logic

Conference or Workshop Item

Conceptual Graphs and First-Order Logic

Michel Wermelinger

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

Abstract. Conceptual Structures (CS) Theory is a logic-based knowledge representation formalism. To show that conceptual graphs have the power of first-order logic, it is necessary to have a mapping between both formalisms. A proof system, i.e. axioms and inference rules, for conceptual graphs is also useful. It must be sound (no false statement is derived from a true one) and complete (all possible tautologies can be derived from the axioms). This paper shows that Sowa's original definition of the mapping is incomplete, incorrect, inconsistent, and unintuitive, and the proof system is incomplete too. To overcome these problems a new translation algorithm is given and a complete proof system is presented. Furthermore, the framework is extended for higher-order types.
Key phrases: logical foundations of Conceptual Structures; ϕ operator; inference rules; logical axioms; higher-order types; meta-level reasoning.

1 Introduction

The logical foundation of CS Theory, as presented in [Sowa, 1984], is based on the definition of the ϕ operator, which translates conceptual graphs into first-order formulas, and on the definition of rules of inference. On page 142 it is claimed that "any formula in first-order logic can be expressed with simply nested contexts and lines of identity", and Theorem 4.4.7 on page 173 states that the inference rules for conceptual graphs are complete. However, as will be shown, the formal definition of ϕ doesn't fulfill the claim and the theorem—which is not proven—is false. Moreover, Sowa has been advocating the use of meta-level graphs. To that end, higher-order types are needed, although they are just as useful to specify finer-grained ontologies.

The purpose of the work to be described in this paper is therefore twofold: on one hand to correct the original definitions, on the other hand to extend them in order to accomodate higher-order types, thus providing a first step towards meta-level reasoning with conceptual graphs. The full reformulation and extension of the (first-order) logical foundations of conceptual graphs is made up of the following steps:

1. Define a first-order language L and an interpretation for it.
2. Define an algorithm to translate conceptual graphs into formulas of L.
3. Define inference rules and logical axioms for conceptual graphs.
4. Prove that they form a sound and complete proof system for first-order logic.

Due to space limitations, steps 1 and 4 have been omitted[1]. They can be found in [Wermelinger, 1995] which also includes the higher-order type framework to be used by ϕ and the inference rules. That framework is a simplified and yet more expressive formulation of the formal proposal for incorporating higher-order types into CS Theory presented in [Wermelinger and Lopes, 1994].

The structure of the paper is straightforward. The next section presents an overview of the adopted higher-order type system, and the other two main sections deal with the ϕ operator and the inference rules, respectively. The reader is expected to have some knowledge of conceptual graphs and logic. Most examples are adapted from [Sowa, 1984; Sowa, 1992].

2 Higher-Order Types

The building units of conceptual graphs are types. There is a concept type hierarchy $\mathcal{T}_{\mathcal{C}}$ and a relation type hierarchy $\mathcal{T}_{\mathcal{R}}$. Both of them are lattices. The top element of $\mathcal{T}_{\mathcal{C}}$ is the universal concept type $\top_c$, and the bottom element is the absurd concept type $\bot_c$. Similarly, the universal relation type $\top_r$ and the absurd relation type $\bot_r$ are the top and bottom elements of $\mathcal{T}_{\mathcal{R}}$. Concept types are classified according to their kind and order, and relation types are classified according to their arity and order.

There are two kinds of concept types: relational and non-relational ones. The former denote relations, the latter do not. The set of all relational concept types is written T^{rc}, and T^{nc} represents all non-relational ones. Concept types can also be classified according to their order: T_i^{rc} is the set of ith-order relational concept types, and the symbol T_i^{nc} stands for the set of all ith-order non-relational concept types. Both $\top_c$ and $\bot_c$ can be of any kind and order. Therefore, they stand apart from the other concept types and aren't included in T^{rc} or T^{nc}. As for relation types, each has an associated arity and order. The set of all n-ary relation types is written as $T^r_{\langle n \rangle}$ and the set of all ith-order relation types is represented by T^r_i. Again, $\top_r$ and $\bot_r$ do not belong to any of those sets.

Example 1. Following are some types and their classification according to the above scheme:

- `CAT`, `FELINE`, `ANIMAL`, `SQUARE`, `RECTANGLE`, `RHOMBUS` $\in T_1^{nc}$;
- `SPECIES`, `GENUS`, `SHAPE` $\in T_2^{nc}$;
- `CATEGORY`, `CHARACTERISTIC` $\in T_3^{nc}$;
- `AGNT`, `OBJ`, `LOC` $\in T^r_{\langle 2 \rangle} \cap T^r_1$;
- `BETW` $\in T^r_{\langle 3 \rangle} \cap T^r_1$;
- `INVERSE-OF` is a relation between two first-order order relations, hence it is an element of $T^r_{\langle 2 \rangle} \cap T^r_2$;
- `RELATION`, `BINARY`, `TRANSITIVE`, `REFLEXIVE`, `ANTI-SYM`, `SYMMETRIC`, `PARTIAL-ORDER` $\in T_2^{rc}$ because there is a second-order relation type.

[1] The completeness proof consists mainly in showing how the axioms and inference rules given in [Hamilton, 1988] can be translated to conceptual graphs.

Simply put, a higher-order type denotes a set of lower-order types, and if t_1 is a subtype of t_2 then the denotation of t_1 must be a subset of t_2's denotation. More specifically, relational concept types denote relation types, non-relational concept types denote other non-relational concept types, and relation types denote tuples of concept types[2]. Therefore, if t_1 is a subtype of t_2 then both must be of the same kind. Furthermore, if t_1 and t_2 are relation types they must have the same arity. That way relation nodes can be generalized or specialized without removing or adding concept nodes to the graph.

Example 2. The only subtype relationships (represented by $<$) among the types of the previous example are:

- `CAT` $<$ `FELINE` $<$ `ANIMAL`;
- `SQUARE` is the maximal common subtype of `RECTANGLE` and `RHOMBUS`;
- `PARTIAL-ORDER` is the maximal common subtype of `TRANSITIVE`, `REFLEXIVE`, and `ANTI-SYM` which in turn are subtypes of `BINARY`;
- `SYMMETRIC` $<$ `BINARY` $<$ `RELATION`.

A concept $\boxed{t : m}$ indicates that m is an entity of type t. In other words, m is an element of the denotation of t. Therefore, if t is a relational concept type then m must be a relation type. Otherwise, i.e. if t is a non-relational concept type, then so is m. To sum up, relation types and non-relational concept types can be used as markers, too. However, as first-order non-relational concept types denote individuals (and not types), a new set T_0^{nc} of "zero-order types" is needed. The elements of T_0^{nc} are mutually incomparable since they represent individuals. All other markers are organized into (disjoint) lattices since they are types. The marker set $\mathcal{M}$ is therefore a partially ordered set. If we add the generic marker `*` as a top element and the absurd marker $\overline{*}$ as a bottom element, then $\mathcal{M}$ becomes a lattice too, like $\mathcal{T_C}$ and $\mathcal{T_R}$.

Example 3. The following concepts show the denotation relationships between the types of Example 1. When a type is used as a marker, it is written in lower case and prefixed with #.

- $\boxed{\mathrm{CAT}\colon \texttt{\#Garfield}}$ where `#Garfield` $\in T_0^{nc}$;
- $\boxed{\mathrm{SPECIES}\colon \texttt{\#cat}}$ and $\boxed{\mathrm{GENUS}\colon \texttt{\#feline}}$;
- $\boxed{\mathrm{SHAPE}\colon \texttt{\#square}}$ $\boxed{\mathrm{SHAPE}\colon \texttt{\#rectangle}}$ $\boxed{\mathrm{SHAPE}\colon \texttt{\#rhombus}}$;
- $\boxed{\mathrm{CATEGORY}\colon \texttt{\#species}}$ $\boxed{\mathrm{CATEGORY}\colon \texttt{\#genus}}$ $\boxed{\mathrm{CHARACTERISTIC}\colon \texttt{\#shape}}$;
- $\boxed{\mathrm{SYMMETRIC}\colon \texttt{\#inverse-of}}$ which implies $\boxed{\mathrm{BINARY}\colon \texttt{\#inverse-of}}$ and $\boxed{\mathrm{RELATION}\colon \texttt{\#inverse-of}}$;
- $\boxed{\mathrm{BINARY}\colon \texttt{\#agnt}}$ $\boxed{\mathrm{BINARY}\colon \texttt{\#obj}}$ $\boxed{\mathrm{BINARY}\colon \texttt{\#loc}}$.

[2] The arguments of a first-order relation are non-relational types, and a higher-order relation has as arguments lower-order relations. But as conceptual graphs are bipartite, those arguments must be represented by concepts. Hence the need for relational concept types.

3 Translation

To show the logical foundations of conceptual graphs, the first step consists in finding a correspondence, i.e. a translation algorithm, between graphs and closed formulas[3] of some first-order language. The latter is implicitly defined by the transformation process. In CS Theory, the translation is given by the ϕ operator. Let us recall its definition as given in [Sowa, 1984].

Assumption 3.3.2. The operator ϕ maps conceptual graphs into formulas in the first-order predicate calculus. If u is any conceptual graph, then ϕu is a formula determined by the following construction:

- If u contains k generic concepts, assign a distinct variable symbol $x_1, \ldots, x_k$ to each one.
- For each concept c of u, let $identifier(c)$ be the variable assigned to c if c is generic or $referent(c)$ if c is individual.
- Represent each concept c as a monadic predicate whose name is the same as $type(c)$ and whose argument is $identifier(c)$.
- Represent each n-adic conceptual relation r of u as an n-adic predicate whose name is the same as $type(r)$. For each i from 1 to n, let the ith argument of the predicate be the identifier of the concept linked to the ith arc of r.
- Then ϕu has a *quantifier prefix* $\exists x_1 \ldots \exists x_k$ and a *body* consisting of the conjunction of all the predicates for the concepts and conceptual relations of u.

Assumption 4.2.3. If p is a proposition asserting the graphs $u_1, \ldots, u_n$, then ϕp is the formula $(\phi u_1 \wedge \ldots \wedge \phi u_n)$. If c is a negative context consisting of (NEG) linked to a proposition p, then ϕc is $\neg \phi p$. All generic concepts that occur in p or any context nested in p must be assigned distinct variable symbols by the formula operator ϕ.

Assumption 4.2.6. If u is a conceptual graph containing one or more lines of identity, compute the formula ϕu by first transforming the graph u according to the following algorithm:

```
assign a unique variable name to every generic concept of $u$;
for $a$ in the set of dominant concepts of $u$ loop
    $x$ := $identifier(a)$;
    append "=$x$" to the referent field of every
        concept dominated by $a$;
end loop;
erase all coreference links in $u$;
```

The formula ϕu is the result of applying ϕ to the transformed version of u with the following rule for mapping concepts with multiple referents: if b is a concept of u of the form [t : $x_1 = x_2 = \ldots = x_n$], then ϕb has the form $t(x_1) \wedge x_1 = x_2 \wedge \ldots \wedge x_1 = x_n$.

[3] A formula with free variables can be regarded as equivalent to its universal closure.

Although these definitions seem trivial, there are several things to notice about them. In the first place, the universal type $\top$ (using Sowa's notation) and the absurd type $\bot$ aren't handled in any special way. Furthermore, a negation sign can never appear immediately before a predicate; there must be always an existential quantifier between them. Notice also that each context corresponds to a closed formula except for one case. If a concept c_1 in context p_1 is dominated by a generic concept c_2 which is in $p_2 \neq p_1$, then the variable corresponding to c_2 appears free in the translation of p_1, but it is bounded in the formula $\phi(p_2)$.

Having these particularities in mind, consider the formula $\forall x\ P(x)$ where P is any unary predicate. Since ϕ only uses the existential quantifier, the formula must be rewritten as $\neg\exists x\ \neg P(x)$. Because of the negated literal, further transformation is necessary in order to get $\neg\exists x\ \neg\exists y\ P(y) \wedge y = x$ which can be represented by the incomplete graph `¬[ ¬[ [P: *y=*x] ] ]` or, in graphical notation, $\neg\boxed{\cdot\cdot\neg\boxed{\cdots\boxed{\text{P}}}}$. But what concept should be linked to the loose end of the coreference link? Sowa's answer is

Definition 4.2.8. If t is a type label for some concept, the *negated type* $\neg t$ is defined by a type definition of the form **type** `¬`t`(x)` **is** `[T: *x] ¬[[`t`: *x]]`.

Therefore, $\neg\exists x\ \neg P(x)$ should be written as $\neg\boxed{\boxed{\neg\text{P}}}$ which gets expanded into $\neg\boxed{\boxed{\text{T}}\cdot\cdot\neg\boxed{\cdots\boxed{\text{P}}}}$. But if we apply the formal definition of ϕ to that graph we obtain $\neg(\exists x\ T(x) \wedge \neg(\exists y\ P(y) \wedge y = x))$ which is only equivalent to the original formula if $T(x)$ is true for any x.

To sum up, the formal definition of the ϕ operator is inconsistent, incomplete, and not intuitive. The fundamental reason is just one: the translation process doesn't reflect the usual interpretation of $\top$ as 'true' and $\bot$ as 'false'. But negated types convey that special meaning of the universal type. Therefore, Assumption 3.3.2, besides not being intuitive, is not consistent with Definition 4.2.8. Furthermore, that interpretation of $\top$ is absolutely necessary for conceptual graphs to be able to represent any closed first-order formula, hence the incompleteness of ϕ.

Adding to the problems mentioned above, the definition of ϕ is not totally correct. On one hand, the empty context gets translated simply into () which is not a well-formed first-order formula because there is no predicate. On the other hand, Assumption 4.2.3 doesn't impose any ordering for the translation of graphs in the same context. That might lead to a formula different from the intended one, if there are coreference links. Consider the graph $\neg\boxed{\boxed{\text{CAT}}\cdots}\cdots\boxed{\text{DOG}}$ which is supposed to state "there is a dog which is not a cat". Applying Assumption 4.2.6, one gets `¬[CAT: *y=*x] [DOG:*x]` which can be translated into $\neg(\exists y\ Cat(y) \wedge y = x) \wedge \exists x\ Dog(x)$ or $\exists x\ Dog(x) \wedge \neg(\exists y\ Cat(y) \wedge y = x)$ depending on the chosen order. The formulas are not logically equivalent because x is free in the first formula. Thus, only the second one corresponds to the intuitive meaning of the graph.

The new translation algorithm given by the ten rules of Definition 1 overcomes all these problems and it also handles higher-order types. However, the basic mechanism remains the same as in Sowa's approach: each concept is assigned a unique variable (rule 2) which is existentially quantified (rule 5); those variables are copied from the dominating to the dominated concepts (rules 4 and 6); the formula corresponding to a graph consists of an existential quantifier prefix followed by the conjunction of the predicates generated by the concepts and relations (rule 8); and negative contexts translate into negated formulas (rule 10).

Let us first see how the above mentioned problems are dealt with. In order to be able to represent any closed first-order formula, the universal type must be translated into a true predicate which simultaneously introduces a new variable. The equality predicate is an obvious candidate. For clarity, it will be written as an infix operator. Furthermore, the symbol $\doteq$ was chosen to avoid any confusion with the meta-level equality $=$ used in definitions. Therefore, $\boxed{\top_c}$ will be translated as $x \doteq x$ (rule 6) where x is the variable associated to the concept. Concepts with the absurd type or the absurd marker are always false and correspond thus to the formula $\neg x \doteq x$[4] (rule 6). Similarly for the universal and absurd relation types (rule 7). Handling empty contexts, the second problem, is just as easy. According to [Sowa, 1984, p. 151], "an empty set of graphs makes no assertion whatever. By convention, it is assumed to be true." This means that having no graph is the same as having $\boxed{\top_c}$. Hence, by inserting this concept into each empty context (rule 1), the usual translation process will take care of the rest. Finally, to prevent ϕ from generating incompatible formulas for graphs in the same context, the quantifier prefix of each graph must be moved to the front of the whole formula (rule 9).

Having fixed ϕ's definition, let us extend it to handle higher-order types. In the new framework, types can be used as markers, and therefore there is also a partial order over markers. Furthermore, higher-order graphs are mainly used for meta-level statements. This means that the interpretation of coreference links should be slightly different. Consider the graph

$$g = \boxed{\text{SHAPE: \texttt{\#rhombus}}} \cdots \boxed{\text{SHAPE: \texttt{\#rectangle}}}$$

If the markers represent single individuals, and therefore `SHAPE` is a first-order type, g states that "a rhombus is the same shape as a rectangle". In logic, the equivalent statement is

$$Shape(rhombus) \wedge rhombus \doteq rectangle \wedge Shape(rectangle) \qquad (1)$$

However, if one considers `RHOMBUS` and `RECTANGLE` to be first-order types, and `SHAPE` to be second-order, then the intuitive reading should be "there is a shape which is both a rhombus and a rectangle". The new translation should be

$$\exists x \; Shape(x) \wedge x \sqsubseteq rhombus \wedge x \sqsubseteq rectangle \qquad (2)$$

[4] This is the infix form of $\neg \doteq (x, x)$ because only predicates can be negated, not variables.

where $\sqsubseteq$ is a special predicate (written as an infix operator) denoting the partial order among markers. Notice that formula 1 is false, but 2 is true since x can be substituted by SQUARE.

However, the translation generated by ϕ won't be exactly as formula 2. Let us see why. Both relation types and non-relational concept types can appear to the left or to the right of ':' in a concept. In other words, most types can be used as markers too. This means that they can be translated as predicates or constants. For example, $\phi(\boxed{\text{CAT: \#Garfield}}) = Cat(garfield)$ but $\phi(\boxed{\text{SPECIES: \#cat}}) = Species(cat)$ where *Cat* and *cat* are different logical symbols. With the purpose of using as few symbols as possible, the form $Holds(cat, garfield)$ will be used instead of $Cat(garfield)$. The "meta-predicate" *Holds* (similar to the one used in the KIF language [Genesereth and Fikes, 1992]) can also be applied to relations. For example, $Agnt(x, y)$ will be written as $Holds(agnt, x, y)$. Formally, as each predicate must have a fixed arity, there is not a single *Holds* but a set $\{Holds_i | i > 0\}$ where i is the arity of the relation type that appears as the first argument of the predicate. Concept types can be seen as unary relation types and therefore $\boxed{t : m}$ won't be translated to $T(m)$ anymore but to $Holds_1(t, m)$ instead (rule 6). Similarly the atomic formula $R(x_1, \ldots, x_n)$ will be rewritten as $Holds_n(r, x_1, \ldots, x_n)$ (rule 7). The predicate $Holds_i$ has therefore arity $i + 1$.

The utilization of $Holds_i$ makes the logical vocabulary even smaller, since $\sqsubseteq$ becomes unnecessary. In fact, if m and m' are types, $m \sqsubseteq m'$ can be restated as $\forall x \; Holds(m, x) \rightarrow Holds(m', x)$ which in turn can be written as $\neg\exists x \; Holds_1(m, x) \wedge \neg Holds_1(m', x)$ (rule 6). Otherwise, i.e. if m and m' are individuals, $m \sqsubseteq m'$ is simply the same as $m \doteq m'$ (rule 6), because neither m nor m' have any subtypes.

Notice that relational concept types can't appear in the referent field of concepts. Therefore, they can't be translated to logical constants and as such can't be quantified over or appear as arguments of some $Holds_i$. This means that $\boxed{t : m}$ will generate $t(m)$ when t is a relational concept type.

Definition 1. The translation of conceptual graphs to first-order logic is done according to the rules that follow. The functions ϕ, ϕ_p, ϕ_b return for each conceptual graph a sequence of logical symbols. The sequence $\phi(g)$ is the *first-order formula* for graph g, and it consists of the *quantifier prefix* $\phi_p(g)$ and the *body* $\phi_b(g)$. For each concept c, the auxiliary functions *id*, *cl*, and *dom* return, respectively, a variable that uniquely identifies c, a boolean that indicates if c is attached to a coreference link, and the set of identifiers of the concepts that dominate c. When necessary, the operator $\odot$ explicity represents the concatenation of symbol sequences.

1. In each empty context of g insert a concept $\boxed{\top_c : *}$.
2. For each concept c let $id(c) = x$, where x is a unique variable.
3. For each concept c let $cl(c) =$ `true` if c is attached to some coreference link, otherwise $cl(c) =$ `false`.
4. For each concept c let $dom(c) = \{id(c') | c' \text{ dominates } c\}$.
5. For each concept c let $\phi_p(c) = \exists id(c)$.

6. For each concept c with $type(c) = t$ and $referent(c) = m$, the formula $\phi_b(c)$ is obtained by the conjuntion of all the following sub-formulas that apply:
   - $\neg id(c) \doteq id(c)$ if $t = \bot_c$ or $m = \bar{*}$;
   - $id(c) \doteq id(c)$ if $t = \top_c$;
   - $t(id(c))$ if $t \in T^{rc}$;
   - $Holds_1(t, id(c))$ otherwise;
   - $\bigwedge_{x \in dom(c)} id(c) \doteq x$;
   - $m \doteq id(c)$ if $m \in T_0^{nc}$ or $cl(c) = \texttt{false}$ and $m \notin \{*, \bar{*}\}$;
   - $(\neg\exists x\ Holds_1(id(c), x) \wedge \neg Holds_1(m, x))$, where $x \neq id(c)$, if $m \notin T_0^{nc} \cup \{*, \bar{*}\}$ and $cl(c) = \texttt{true}$.
7. Let r be a relation with concepts $c_1, \ldots, c_n$ as arguments. If $type(r) = \top_r$ or $type(r) = \bot_r$ then $\phi_b(r) = id(c_1) \doteq id(c_1)$ or $\phi_b(r) = \neg id(c_1) \doteq id(c_1)$, respectively. Otherwise $\phi_b(r) = Holds_n(type(r), id(c_1), \ldots, id(c_n))$.
8. If g is a conceptual graph without contexts and with concepts C and relations R, then $\phi(g) = \phi_p(g)\phi_b(g)$ where

$$\phi_p(g) = \bigodot_{c \in C} \phi_p(c) \qquad\qquad \phi_b(g) = \bigwedge_{c \in C} \phi_b(c) \wedge \bigwedge_{r \in R} \phi_b(r)$$

9. If p is a proposition containing the set of graphs G then $\phi_p(p)$ is the empty sequence and $\phi(p) = \phi_b(p) = (\bigodot_{g \in G} \phi_p(g) \bigwedge_{g \in G} \phi_b(g))$.
10. If c is a context formed by the negation of proposition p, then $\phi_p(c)$ is the empty sequence and $\phi(c) = \phi_b(c) = \neg\phi(p)$.

Several translation examples follow. They show the difference between the old and the new definition of ϕ, and illustrate how some previously problematic cases are now handled. Table 2 gives further examples.

Example 4. The translation of $\boxed{\text{CAT: \#Garfield}} \leftarrow (\text{AGNT}) \leftarrow \boxed{\text{CHASE}} \rightarrow (\text{OBJ}) \rightarrow \boxed{\text{DOG}}$ is

$$\exists x \exists y\ Cat(garfield) \wedge Chase(x) \wedge Dog(y) \wedge Agnt(x, garfield) \wedge Obj(x, y)$$

according to Assumption 3.3.2. Applying Definition 1 instead, one has

$$\exists x \exists y \exists z\ Holds_1(cat, z) \wedge z \doteq garfield \wedge Holds_1(chase, x) \wedge Holds_1(dog, y) \wedge$$
$$Holds_2(agnt, x, z) \wedge Holds_2(obj, x, y)$$

Example 5. The formula $\forall x\ P(x)$ states basically that "if x is some entity then $P(x)$ is true". Let P be any relational type[5]. Then

$$\phi(\neg\boxed{\boxed{\top_c}\cdots\neg\boxed{\cdots\boxed{\text{P}}}}) = \neg(\exists x\ x \doteq x \wedge \neg(\exists y\ P(y) \wedge y \doteq x))$$

[5] The result would be similar if P were a non-relational type. Just substitute $Holds_1(p, x)$ for $P(x)$.

This formula is equivalent to $\forall x\ x \doteq x \rightarrow \exists y\ P(y) \wedge y \doteq x$. Due to the properties of equality, $x \doteq x$ is always true and $P(y) \wedge y \doteq x$ corresponds to $P(x)$. Thus one gets $\forall x\ P(x)$ as expected.

Example 6. According to the formulation of rule 9, the graph $\neg\boxed{\boxed{\text{CAT}}\cdots}\cdots\boxed{\text{DOG}}$ shown before is correctly translated as

$$\exists x\ \neg(\exists y\ Holds_1(cat, y) \wedge y = x) \wedge Holds_1(dog, x)$$

Example 7. The graph $\boxed{\text{SHAPE: \#rectangle}}\cdots\boxed{\text{SHAPE: \#rhombus}}$ has a coreference link between higher-order concepts. The corresponding formula is therefore

$$\begin{aligned}\exists x \exists y\ Holds_1(shape, x) \wedge x \doteq y \wedge (\neg\exists z\ Holds_1(x, z) \wedge \neg Holds_1(rectangle, z)) \wedge \\ Holds_1(shape, y) \wedge y \doteq x \wedge (\neg\exists z\ Holds_1(y, z) \wedge \neg Holds_1(rhombus, z))\end{aligned}$$

Example 8. The coreference link in $\boxed{\text{PERSON: Rosalie}}\cdot\cdot\neg\boxed{\cdots\boxed{\text{PERSON: Rosann}}}$ connects two first-order concepts. Applying rule 6 in this case leads to

$$\exists x Holds_1(person, x) \wedge x \doteq rosalie \wedge \neg(\exists y\ Holds_1(person, y) \wedge y \doteq rosann \wedge y \doteq x)$$

The ϕ operator just translates a sequence of symbols of some language (Conceptual Graphs) into another sequence of symbols (called formula) of some other language (first-order logic). For this process to have any meaning, the resulting formulas must have an interpretation. Classically, an interpretation of a first-order language L is a pair $\langle D, \delta \rangle$ where the denotation function δ maps constants of L into elements of the domain D and predicates into tuples of elements of D. The new definition of interpretation [Wermelinger, 1995] just adds the constraints presented informally in Section 2.

4 Inference

Theoretically, the translation and interpretation of conceptual graphs is important to show the formalism's expressiveness. But the main goal is to have inference rules that operate directly on conceptual graphs, instead of translating the graphs to formulas, do the proofs with them and then translating back to the graphical form.

The proof system given in [Sowa, 1984] consists of a single axiom, the empty set of graphs, and several first-order rules of inference. These are mainly based on the depth of a graph, i.e. on how many negative contexts one must traverse to reach the graph starting from the outer context. Depending on the depth, the graph is said to be in an evenly enclosed or oddly enclosed context. Even contexts contain true graphs and odd contexts contain false graphs. Therefore, conditions (i.e, graphs and coreference links) can be removed from the former and added to the latter. Moreover, a context c dominating a context c' (that is, $c' = c$ or c' is enclosed in a context dominated by c) corresponds to an implication and therefore the graphs in c (the antecedent) can be copied to c' (the consequent).

As simple and elegant it is, Sowa's system must be changed, even if one considers the corrected version of ϕ and no higher-order types. In fact, there are now several ways of representing truth, and each true graph that can't be derived from others must be an axiom. Otherwise the system won't be complete. Moreover, a new rule must be added: axioms may be inserted and removed from any context. Without these changes the universal instantion rule can't be applied to conceptual graphs. Using the old notation for clarity, consider the example $\forall x\ Cat(x) \vdash Cat(garfield)$. In graphical form the hypothesis is $\neg\boxed{\boxed{\top_c}\cdots\neg\boxed{\cdots\boxed{\text{CAT}}}}$. Restricting the referent in the oddly enclosed context one gets $\neg\boxed{\boxed{\top_c\text{: }\texttt{\#Garfield}}\cdots\neg\boxed{\cdots\boxed{\text{CAT}}}}$. By the individuation rule, a individual marker can be iterated from a dominating concept to a dominated one[6] and the coreference link may be erased, provided the dominated concept is generic. We thus get $\neg\boxed{\boxed{\top_c\text{: }\texttt{\#Garfield}}\ \neg\boxed{\boxed{\text{CAT: }\texttt{\#Garfield}}}}$ but can't proceed any further because graphs can't be removed from odd contexts.

The remaining of this section presents therefore a new formal proof system. For the most part it is similar to Sowa's. The changes that were done (including the above mentioned) are due to the type and marker hierarchies, the new interpretation of universal and absurd types, and the new meaning of coreference links resulting from the use of higher-order types.

In [Sowa, 1984] only concept types formed a hierarchy. Relation types and markers were incomparable. Therefore, the inference rules only enabled one to restrict concept types, i.e. to substitute them by subtypes, and to replace the generic marker by an individual marker, or the other way round. In this framework, relation types and markers may also be (un)restricted but there are some limitations. Let t and t' be any concept or relation types, such that t is a subtype of t'. Therefore, if $Holds_n(t, x_1, \ldots, x_n)$ is true, then $Holds_n(t', x_1, \ldots, x_n)$ is also true, and if the latter is false, so is the former. Thus any type may be unrestricted in evenly enclosed contexts and it may be substituted by a subtype in oddly enclosed contexts. In this respect higher-order types don't change the original inference rules.

However, markers can't be changed at will. Consider Examples 1 and 3: `CAT` is a subtype of `FELINE` which is a `GENUS` while `CAT` is a `SPECIES`. If the graph $\boxed{\text{SPECIES: }\texttt{\#cat}}$ is in an evenly enclosed context it is true, but it can't be generalized to the false graph $\boxed{\text{SPECIES: }\texttt{\#feline}}$. Similarly, if the latter is in an oddly enclosed context, it can't be specialized to the former. To sum up, individual markers can't be (un)restricted to other individual markers but they can be transformed into the generic or absurd markers. For example, the true graph $\neg\boxed{\boxed{\text{SPECIES: }\texttt{\#feline}}}$ can be specialized to the equally true $\neg\boxed{\boxed{\text{SPECIES: }\bar{*}}}$.

There is however one case where the marker hierarchy can be put to use,

[6] A concept c_1 dominates a concept c_2 if there is a coreference link between them and the context of c_1 dominates the context of c_2.

Context of c	Coreference link?	m	Action
even	no	$\bar{*}$	unrestrict
even	no	$\neq\bar{*}$	unrestrict to $*$
even	yes	any	unrestrict
odd	no	$*$	restrict
odd	no	$\neq *$	restrict to $\bar{*}$
odd	yes	any	restrict

Table 1. Conditions for changing referent m of concept c

namely if a coreference link is present. Let m be the marker of some dominating or dominated concept c whose identifier is the variable x. If m is a type, the condition $x \sqsubseteq m$ belongs to the context of c. If that context is even, the condition is true and so is $x \sqsubseteq m'$ where m' is a supertype of m. If the context is odd, the condition is false and restricting m to some subtype doesn't make it true. Table 1 summarizes all these conditions.

There is one more situation where markers can be restricted. Consider a concept c_1, with marker m_1 and identifier x_1, dominating a concept c_2 with referent m_2 and associated variable x_2. Then, the condition $x_2 \doteq x_1$ enables one to iterate any condition on x_1 from c_1's context to c_2's context. This corresponds to the replacement of m_2 by m_{12}, the greatest lower bound of m_1 and m_2: if $x_1 \sqsubseteq m_1$ and $x_2 \sqsubseteq m_2$ then $x_2 \sqsubseteq m_{12}$[7] (assuming $x_2 \doteq x_1$). Notice however that the restriction on m_2 can only be done if the result isn't the absurd marker, because a false graph might be obtained if c_2 is evenly enclosed. If c_2 were oddly enclosed the last line of Table 1 would apply and therefore this new rule, which finds an upper limit for the value x_2 that satisfies the formula, wouldn't be needed.

As for logical axioms, Sowa only uses the empty set of graphs. As seen in the previous section, some predicate *true(x)* is needed in order to be able to represent all closed formulas of first-order logic. That predicate turned out to be the equality $\doteq$. Therefore, the new axioms are graphs whose translation is some tautology based on $x \doteq x$. Looking at Definition 1 the possibilities listed in Table 2 are obtained, where m and m' are any markers different from $\bar{*}$ and t, t' are any concept types (although the given translations assume that they are non-relational).

It is obvious that the graphs involving $\top_r$ and $\bot_r$ may have any arity. However, there is a subtle difference. The mere presence of the absurd type $\bot_r$ automatically makes the graph false, and therefore the axiom true. The concepts used as relation arguments are thus irrelevant. But the same does not happen with the universal type $\top_r$. The concepts to which it is linked must be true too for the whole graph to be true. It is also worth noticing that ϕ translates the empty context in the same way as $\boxed{\top_c}$. The rules of inference will of course

[7] See Example 9.

$\phi(\boxed{})$	$= \exists x\ x \doteq x$
$\phi(\boxed{\top_c : m})$	$= \exists x\ x \doteq x \wedge x \doteq m$
$\phi(\neg\boxed{\boxed{\bot_c : m}})$	$= \neg\exists x\ \neg x \doteq x \wedge x \doteq m$
$\phi(\neg\boxed{\boxed{t : \bar{*}}})$	$= \neg\exists x\ \mathit{Holds}_1(t, x) \wedge \neg x \doteq x$
$\phi(\neg\boxed{(\bot_r) \rightarrow \boxed{t : m}})$	$= \neg\exists x\ \neg x \doteq x \wedge \mathit{Holds}_1(t, x) \wedge x \doteq m$
$\phi(\neg\boxed{\boxed{t : m} \rightarrow (\bot_r) \rightarrow \boxed{t' : m'}})$	$= \neg\exists x \exists y\ \neg x \doteq x \wedge \mathit{Holds}_1(t, x) \wedge x \doteq m \wedge \mathit{Holds}_1(t', y) \wedge y \doteq m'$
$\phi((\top_r) \rightarrow \boxed{\top_c : m})$	$= \exists x\ x \doteq x \wedge x \doteq x \wedge x \doteq m$
$\phi(\boxed{\top_c : m} \rightarrow (\top_r) \rightarrow \boxed{\top_c : m'})$	$= \exists x \exists y\ x \doteq x \wedge x \doteq x \wedge x \doteq m \wedge y \doteq y \wedge y \doteq m'$

Table 2. Logical axioms

allow one to insert and erase logical axioms from any context. The empty context becomes therefore obsolete because it can be derived from any other axiom by erasure. However, it will be kept for convenience. There are other redundant graphs in the above table. For example, any $\neg\boxed{\boxed{\bot_c : m}}$ can be obtained from $\neg\boxed{\boxed{\bot_c : *}}$ by restricting the referent (see the fourth line of Table 1). In the same way, $\neg\boxed{(\bot_r) \rightarrow \boxed{t : m}}$ can be derived from $\neg\boxed{(\bot_r) \rightarrow \boxed{\top_c : *}}$. The final set of logical axioms can be found in Definition 2.

Finally, the rules for handling coreference links are basically the same as in [Sowa, 1984] when the dominated concept is first-order. Otherwise a coreference link can't be inserted or removed in the general case. Let us see why. Consider again concepts c_1 and c_2 mentioned before. When a coreference link is drawn the following happens:

1. The condition $x_2 \doteq x_1$ is added to the context of c_2.
2. The condition $x_2 = m_2$ becomes $x_2 \sqsubseteq m_2$ if $m_2 \notin T_0^{nc} \cup \{*, \bar{*}\}$.
3. The condition $x_1 = m_1$ becomes $x_1 \sqsubseteq m_1$ if $m_1 \notin T_0^{nc} \cup \{*, \bar{*}\}$.

The erasure of a coreference link consists in doing the opposite actions. Due to step 1, coreference links can't be inserted when c_2 is evenly enclosed, and they can't be removed if c_2 is in an odd context. Additionally, if c_2 is a higher-order concept, step 2 applies. In this case inserting a coreference link relaxes the condition (i.e., it could become true), and therefore c_2 can't be oddly enclosed. Inversely, erasing a coreference link makes the condition stronger ($x_2 \sqsubseteq m_2$

Context of c_1	Context of c_2	$m_1 \in T_0^{nc} \cup \{*, \bar{*}\}$	$m_2 \in T_0^{nc} \cup \{*, \bar{*}\}$	Action
any	even	yes	yes	erasure
any	odd	yes	yes	insertion
odd	even	no	yes	erasure
even	odd	no	yes	insertion
odd	even	no	no	erasure if $m_1 = m_2$
even	odd	no	no	insertion if $m_1 = m_2$

Table 3. Conditions for changing coreference links

becomes $x_2 = m_2$) which prevents c_2 from being evenly enclosed[8]. Steps 1 and 2 thus impose contradictory restrictions on the context of c_2.

Fortunately, there is an exception, namely if $m_1 = m_2$. Consider the case where c_2 is evenly enclosed and the coreference link has therefore been removed. The new conditions are $x_2 \doteq m_2$ and $x_1 \doteq m_1$. Due to our assumption, the conditions are equivalent and therefore the erasure corresponds to the iteration of a condition from the dominating to the dominated context. The other possibility is to insert a coreference link if c_2 is oddly enclosed. The new conditions are $x_1 \sqsubseteq m_1$ in c_1's context and $x_2 \sqsubseteq m_2 \wedge x_2 \doteq x_1$ for c_2. Again, due to their equivalence, insertion of a coreference link corresponds to an iteration.

Table 3 summarizes the preceding observations. No action is possible for the unlisted cases. It should be obvious that it is not necessary to check any restrictions if the coreference link to be inserted or removed is an exact copy of another existing one. Also, since coreference links represent equalities, they may be inserted or removed according to the transitivity rule.

The inference rules can at last be presented.

Definition 2. Let S be a set of conceptual graphs in the outer context. Any graph derived from S by the following *first-order rules of inference* is said to be *provable* from S.

Equivalence In any context, a logical axiom may be inserted or removed, a double negation may be drawn or erased around any set of graphs.

Generalization In an evenly enclosed context, any type or marker may be unrestricted, and any graph or coreference link may be deleted, as long as the conditions in Tables 1 and 3 are obeyed.

Specialization In an oddly enclosed context, any type or marker may be restricted, and any graph or coreference link may be inserted, as long as the conditions in Tables 1 and 3 are obeyed.

Iteration A graph may be copied from context c to any context dominated by c, and coreference links may be drawn between the original concepts and their copies.

Deiteration The result of some possible iteration may be deleted.

[8] The same reasoning applies to c_1 if it isn't a first-order concept.

Transitivity If concept c_1 dominates concept c_2 which in turn dominates $c_3 \neq c_1$, then a coreference link between c_1 and c_3 may be drawn or erased. If it is inserted, then the coreference link between c_2 and c_3 may be erased.

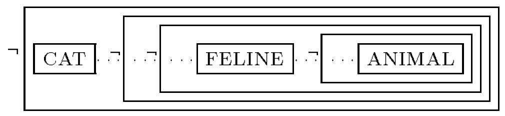
Individuation If concept c_1 dominates concept c_2 then *referent*(c_2) may be replaced by the greatest lower bound of *referent*(c_1) and *referent*(c_2) if the result is different than $\bar{*}$.

A graph provable from the following *logical axioms* is called a *theorem*.

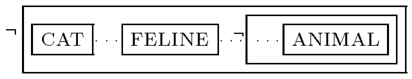
- The empty set of graphs $\{\}$;
- $\boxed{\top_c\colon m}$ for any $m \in \mathcal{M} - \{\bar{*}\}$;
- $\neg\boxed{\boxed{\bot_c\colon *}}$;
- $\neg\boxed{\boxed{\top_c\colon \bar{*}}}$;
- $\neg\boxed{(\bot_r)\rightarrow\boxed{\top_c\colon *}}$ and for any other arity;
- $(\top_r)\rightarrow\boxed{\top_c\colon m}$ for any $m \in \mathcal{M} - \{\bar{*}\}$ and any arity.

Example 9. Applying the individuation rule twice, and considering Example 2, the graph $\boxed{\text{SHAPE: \texttt{\#rectangle}}}\cdots\boxed{\text{SHAPE: \texttt{\#rhombus}}}$ is first transformed to $\boxed{\text{SHAPE: \texttt{\#rectangle}}}\cdots\boxed{\text{SHAPE: \texttt{\#square}}}$ and then $\boxed{\text{SHAPE: \texttt{\#square}}}\cdots\boxed{\text{SHAPE: \texttt{\#square}}}$ is derived. Notice that this graph doesn't necessarily imply $\boxed{\text{SHAPE: \texttt{\#square}}}$ because the former states that there exists a *subtype* of `SQUARE` which is a shape while the latter states that `SQUARE` *itself* is a shape.

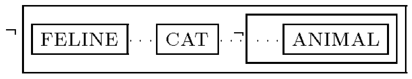
Example 10. The subtype relationships `CAT` < `FELINE` < `ANIMAL` can be stated by the graph



Erasing the double negation (an equivalence rule), it can be simplied to



and applying the transitivity rule one gets



The first graph corresponds indeed to the given type hierarchy fragment, as can be easily seen by the translation of it:

$$\neg\exists x\ Holds_1(cat, x) \wedge \neg\neg\exists y\ Holds_1(feline, y) \wedge y \doteq x \wedge$$
$$\neg\exists z\ Holds_1(animal, z) \wedge z \doteq y$$

or more simply

$\forall x\ Holds_1(cat, x) \rightarrow \forall y Holds_1(feline, y) \wedge x \doteq y \rightarrow \exists z\ Holds_1(animal, z) \wedge z \doteq y$

This formula is equivalent to

$\forall x \forall y\ Holds_1(cat, x) \wedge Holds_1(feline, y) \wedge x \doteq y \rightarrow \exists z\ Holds_1(animal, z) \wedge z \doteq y$

which is the translation of the second graph. Obviously, it can be rewritten as

$\forall x \forall y\ Holds_1(cat, x) \wedge Holds_1(feline, y) \wedge x \doteq y \rightarrow \exists z\ Holds_1(animal, z) \wedge z \doteq x$

corresponding to the last graph.

5 Conclusions

This paper has provided a closer look at the logical foundations of Conceptual Structures Theory. It was shown that the original formal definitions of [Sowa, 1984] are incomplete: on one hand, some closed first-order formulas can't be represented with conceptual graphs, on the other hand the universal instantiation rule is missing. Therefore, the definitions of the ϕ operator and of the first-order inference rules have been corrected. Furthermore, they have been extended to handle higher-order types.

It is hoped that this paper provides a first step towards a meta-level reasoning engine and a deeper investigation of the model-theoretic and proof-theoretic properties of Conceptual Structures.

References

[Genesereth and Fikes, 1992] 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.

[Hamilton, 1988] A. G. Hamilton. *Logic for Mathematicians.* Cambridge University Press, 1988. Revised edition.

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

[Sowa, 1992] John F. Sowa. Conceptual graph summary. In 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, pages 3–51. Ellis Horwood, 1992.

[Wermelinger and Lopes, 1994] Michel Wermelinger and José Gabriel Lopes. Basic conceptual structures theory. In William M. Tepfenhart, Judith P. Dick, and John F. Sowa, editors, *Conceptual Structures: Current Practices — Proceedings of the Second International Conference on Conceptual Structures*, number 835 in Lecture Notes in Artificial Intelligence, pages 144–159, College Park MD, USA, 16–19 August 1994. University of Maryland, Springer-Verlag.

[Wermelinger, 1995] Michel Wermelinger. Teoria Básica das Estruturas Conceptuais. Master's thesis, Universidade Nova de Lisboa, 1995.

This article was processed using the LaTeX macro package with LLNCS style