Negations in Simple Concept Graphs

Frithjof Dau

Technische Universität Darmstadt, Fachbereich Mathematik
Schloßgartenstr. 7, D-64289 Darmstadt, dau@mathematik.tu-darmstadt.de

Abstract. The aim of this paper is to mathematically introduce negation to concept graphs (which are a mathematical modification of conceptual graphs) as a well-defined syntactical construct. First off, we discuss some questions which arise when negations for conceptual graphs are defined. In our view, a solution for these questions is to express negations by cuts in the sense of Peirce's theory of existential graphs. A set-theoretical semantics for (nonexistential) concept graphs with cuts is developed in the framework of contextual logic. A modification of Peirce's alpha-calculus, which is sound and complete, is presented.

1 Motivation

Conceptual graphs are based on the *existential graphs* of Charles Sanders Peirce. These graphs consist of lines called *lines of identities*, predicate names of arbitrary arity and ovals around subgraphs which are used to negate the enclosed subgraph. The following three examples are well known:

The meanings of these graphs are: 'a cat is on a mat', 'no cat is on any mat' and 'there is a cat and there is a a mat such that the cat is not on the mat'.

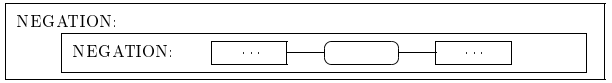
As Peirce says, "That a proposition is false is a *logical* statement about it, and therefore in a logical system deserves special treatment." ([Pe98]). The graphical element *oval* which Peirce used to negate its enclosure has been transferred to context boxes in conceptual graphs. These boxes are used to express that some information is valid in specific contexts or situations. Hence, the character of negation as logical operator in existental graphs changed to a metalevel character in conceptual graphs. Of course, in knowledge representation and natural language, negations are unavoidable. So the feature to express negations is desirable in concept graphs.

To handle negation in concept graphs, we need to achieve the following aims: For the mathematical treatment, formation rules for the well-formed formulas must exist that can express negations, and negation has to be covered by rules of inference. To do this in the spirit of Peirce, the semantics of negation has to be intelligible, and the graphical representation of negations must be easily readable and intuitively understandable (which has been an important goal in the theory of conceptual graphs from the very beginning, too).

Negations occur in several approaches for conceptual graphs. Why we do not adopt and mathematizise one of these approaches shall be explained in the rest of this section.

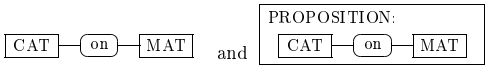
In order to handle negations, a specific syntactical element of the well-formed formulas has to be declared to express them. For this purpose, the standard approach use a context box of type `Proposition` which is linked to a unary relation of type `neg` ([So99]). Sometimes, these special context boxes are abbreviated by context boxes of type `Negation` (e.g. [So00]) or by drawing a simple rectangle with the mathematical negation symbol ¬ ([So99]). Some approaches use these rectangles without declaring whether the box is a specific syntactical element or just an abbreviation for context boxes of a specific type (e.g. [We95]). But both a calculus and a translation of conceptual graphs into other formal languages (like the translation to first order logic with the Φ-operator), have to respect the logical role of negation.

So, if negation is expressed just by special context boxes, any calculus and any tranlation has to treat these special context boxes differently from all other context boxes. For example, if negation is expressed by context boxes of type `Negation`, a calculus should allow the nested boxes in the following graph to be erased (and vice versa, to be introduced again):



This seems to be not possible in any calculus which does not treat the negation boxes separately (like the calculus of Prediger ([Pr98b]) or any calculus which is based on projections).

If negation is expressed with contexts of type `Proposition`, linked to a unary relation `neg`, another difficulty appears. This shall be shown by the following two conceptual graphs:

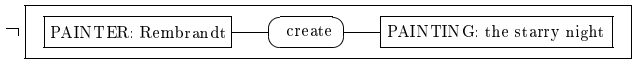


The first graph is well known: Its meaning is 'a cat is on a mat'. In particular, the graph claims to be true. The meaning of the second graph is, strictly speaking, 'there exists a proposition, which states that a cat is on a mat' ([So99]), and therefore different to the meaning of the first graph. Indeed; in none of the common calculuses, one graph can be derived from the other one. Hence it is problematic to express the negation of the first graph by the second graph with a relation `neg`.

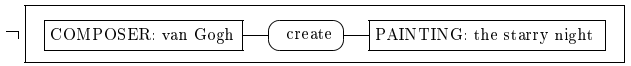
To summarize: It seems to be difficult to introduce negation as a special context box. These context boxes have to be treated differently to other boxes in the calculus and in any translation from conceptual graphs to other formal languages (like the operator Φ). This yields the following conclusion: For the mathematical treatment of negation in concept graphs, in the definition of their well-formed formulas there should be a specific syntactical element which is used to express negation.

The next step has to clarify the semantics, i.e. the meaning, of negation. To prepare this, we will discuss a small example. Consider the true proposition 'the painter Rembrandt created the painting 'the nightwatch", which can be translated to the following conceptual graph:

This graph represents not only the information that Rembrandt created 'the nightwatch', but also that Rembrandt is a painter and 'the nightwatch' is a painting. Now, consider the painting 'a starry night' instead of 'the nightwatch'. This painting was created by van Gogh, so the proposition 'the painter Rembrandt did not create the painting 'the starry night" is true. How can this proposition be transformed to a conceptual graph? The following graph is a first attempt:



This graph is *not* the translation of the former proposition: In the proposition, only the verb 'to create' is negated, but in the graph, the negation box also encloses the information that Rembrandt is a painter and 'the starry night' is a painting. The information in the concept boxes can fail, too, as can be seen in the following graph:



This graph is true although van Gogh did create 'the starry night'. In particular, this graph should not be read as

The composer Van Gogh did not *create the painting 'a starry night'.*

But this understanding is suggested when Sowa in [So00] says, that the meaning of the graph `[Negation: [Cat: Yoyo]→(On)→[Mat]]` is 'the graph denies that the cat Yoyo is on a mat'. Now, the goal is to negate only the verb 'to create' in the false proposition 'the painter Rembrandt created the painting 'the starry night". This problem has already been addressed, one approach for its solution is the following graph:

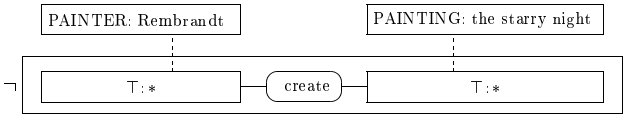


Fig. 1. CG for 'the painter Rembrandt did not create the painting 'a starry night"

Indeed, this graph expresses the proposition 'the painter Rembrandt did not create the painting 'the starry night'. But obviously, the aim of making conceptual graphs easily readable and intuitively understandable is not fulfilled.

Expressing identity in conceptual graphs with coreference-links or coreference-sets leads to another class of difficulties. In particular, the meaning of coreference-links connected to a concept box within a negation is not straightforward. This shall be explained next:

In conceptual graphs, coreference-links (which are used to express coreference-sets ([So99])) are used to express the identity of two entities: "Two concepts that refer to the same individual are coreferent. [...] To show that they are coreferent, they are connected with a dotted line, called a *coreference link*." ([So92]). Consider the following graph:

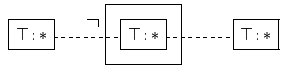
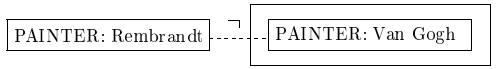


Fig. 2. conceptual graph for $\exists x \exists y . x \neq y$

According to Sowa ([So00]), the operator Φ translates this graph into the first order logic formula $\exists x \exists y \neg \exists z (z = x \wedge z = y)$, which is equivalent to $\exists x \exists y . x \neq y$. In particular, the three concept boxes cannot refer to the same individual. Note that Φ assigns different variables to the generic markers of different concept boxes, even if they are connected with a coreference-link. These variables are explicitly set to be equal in the formula, and they are equated *inside the negated part* of the formula. But since the links in the graph looks symmetric, it is not clear to a reader why the equating in the formula is placed *inside* and not *outside* of the negated subformula. This ambiguity can be seen even better in the following example:

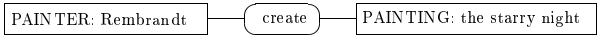


If Φ translates this graph to $PAINTER(R) \wedge \neg(R = VG \wedge PAINTER(VG))$, the resulting formula is true, but if Φ translates this graph to $PAINTER(R) \wedge Rembrandt = VG \wedge \neg(PAINTER(VG))$ the resulting formula is not true (the names in the formulas are abbreviated by R and VG). Hence, in order to understand the right meaning of this graph, the reader must have in mind the implicit agreement that equality is always placed in the inner context.

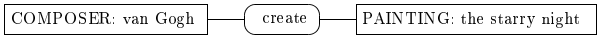
If we accept this meaning of corerence links, the next step is is to make clear which syntactical element in the well-formed formulas is used for them, and how they are handled by a calculus. In the abstract syntax of conceptual graphs ([So99]), coreference-links are generalized to coreference-sets. For example, Figure 2 has two coreference sets which are represented by coreference-links. Coreference-sets are sufficient to handle coreference in conceptual graphs with negations. But still rules are needed that treat these sets in a sound and complete way (for example, there have to be rules which allow a link to be drawn or erased from a concept box to itself). Some calculuses lack rules like this.

To summarize again: Introducing coreference sets to express identity may lead to misunderstandings of their meaning and to gaps in their syntactical implementation.

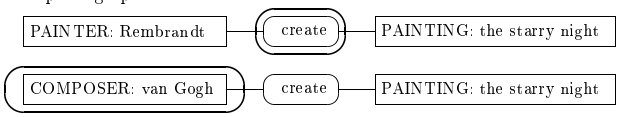
To cope with all the mentioned problems, we suggest the following: First, negations should be introduced as a new syntactical element, namely the ovals of Peirce, which can be drawn around arbitrary parts of a conceptual graph. To distinguish these ovals from the ovals which are drawn around relation names, we propose drawing them in bold. For example, in



only the relation `create` has to be negated, and in



only the concept box COMPOSER: van Gogh has to be negated. The resulting conceptual graphs are:



Because the present interpretation of coreference-links is not intuitive in some sense, we suggest to introduce a special binary relation id (as in first order logic). The advantages of this approach are

1. id can be trated like other relations and
2. the identity can be negated without loss of readability.

In our view this yields a more understandable notion of identity (as understood in mathematics). For example, the meaning of the graph

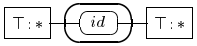
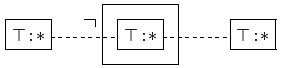


Fig. 3. CG with negation ovals for $\exists x \exists y . x \neq y$

is 'there exist at least two things'. Thus, it has the same meaning as the graph in Figure 2, but is much simpler. Furthermore, it shows that id is a proper syntactical extension and not a direct mathematization of coreference links.

Since the syntactical elements which allow negations and identity are extended and in this approach, every conceptual graph with negation boxes and coreference-links can be translated into a concept graph with negation ovals and the relation id. Of course, negation boxes (for example, context boxes of type `Negation`) are translated to negation ovals. A coreference-link between two context boxes is translated into a relation id between the boxes such that the relation node id is placed in the negation oval of the dominated context box. We will exemplify this with the following: The conceptual graph



is translated to the following conceptual graph with negation ovals and the relation id:

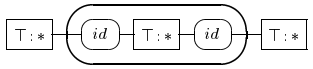


Fig. 4. another CG with negation ovals for $\exists x \exists y . x \neq y$

This graph can be transformed to the graph in Figure 3, which has the same meaning, but looks much simpler. On the other hand, concept graphs with negation ovals and the relation id can be translated to graphs with negation boxes and coreference-links. But because negation ovals need not include subgraphs, but arbitrary subsets of concept nodes and relations nodes, this translation is more complicated than the translation in the other direction. For example: Before the graph in Figure 3 can be translated, it has to be transformed into the graph in Figure 4.

The approach we present here is closely related to the original ideas of Peirce. It is easier to mathematize than approaches based on concept boxes of a specific type. In this paper, our approach shall be elaborated for simple concept graphs without generic markers, but with negation ovals and the relation id. In particular, the syntax for these graphs is defined, an extensional semantics for these graphs is introduced (which is based on power context families), and a sound and complete calculus is presented. Furthermore, this approach allows to define mathematically the operator Φ on simple concept graphs (which maps graphs to first order logic formulas) and its inverse operator Ψ (which maps first order logic formulas to graphs) such that both respect the (syntactical or semantic) entailment relation on graphs and formulas, respectively. In particular, the expressiveness of simple graphs and first order logic formulas is the same. This will be elaborated in a work which is in progress now.

2 Basic Definitions

Simple concept graphs are introduced by Prediger in [Pr98b] as mathematically defined syntactical constructs. We take into account her approach and extended it to include the possibily to express negations by using cuts and the possibility to express identity by using a special binary relation id.

First we have to start with ordered sets of names for objects, names and relations. These orders represent the conceptual ontology of the domain we consider.

Definition 1. *An* alphabet of conceptual graphs *is a tripel* $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ *such that*

- $\mathcal{G}$ *is a finite set whose elements are called* object names
- $(\mathcal{C}, \leq_{\mathcal{C}})$ *is a finite ordered set with a greatest element* $\top$ *whose elements are called* concept names
- $(\mathcal{R}, \leq_{\mathcal{R}})$ *is a union of finite ordered sets* $(\mathcal{R}_k, \leq_{\mathcal{R}_k})$, $k = 1, \ldots, n$ *(for an* $n \in \mathbb{N}$ *with* $n \geq 1$*) whose elements are called* relation names. *Let* $id \in \mathcal{R}_2$.

Now we can define the underlying structures of concept graphs with cuts. This definition extends the definition of directed multi-hypergraphs given in [Pr98b] by cuts, so that negations can be expressed.

Definition 2. *A* directed multi-hypergraph with cuts (of type n) *is a structure* $(V, E, \nu, Cut, area)$ *such that*

- V *and* E *are finite sets whose elements are called* vertices *and* edges, *respectively,*
- $\nu : E \to \bigcup_{k=1}^{n} V^k$ *(for a* $n \in \mathbb{N}, n \geq 1$*) is a mapping,*
- Cut *is a finite set whose elements are called* cuts *and*
- $area : Cut \to \mathfrak{P}(V \cup E \cup Cut)$ *is a mapping such that* $c \notin area(c)$ *for each* $c \in Cut$ *and, for two cuts* c_1, c_2 *with* $c_1 \neq c_2$, *exactly one of the following conditions holds:*
  i) $\{c_1\} \cup area(c_1) \subseteq area(c_2)$,
  ii) $\{c_2\} \cup area(c_2) \subseteq area(c_1)$,
  iii) $(\{c_1\} \cup area(c_1)) \cap (\{c_2\} \cup area(c_2)) = \emptyset$.

For an edge $e \in E$ *with* $\nu(e) = (v_1, \ldots, v_k)$ *we define* $|e| := k$ *and* $\nu(e)\big|_i := v_i$. *For each* $v \in V$, *let* $E_v := \{e \in E \mid \exists i \colon \nu(e)\big|_i = v\}$, *and analogously for each* $e \in E$, *let* $V_e := \{v \in V \mid \exists i \colon \nu(e)\big|_i = v\}$. *If it cannot be misunderstood, we write* $e\big|_i$ *instead of* $\nu(e)\big|_i$.

The notion of *cuts* and *areas* is closely related to the ideas of Peirce, as they are described in the work of Roberts (see [Ro73]). Peirce negated parts of an existential graph just by drawing an oval around it. This oval (more exactly just the line which is drawn on the sheet of assertion) is called a *cut*. In particular, a cut is not a graph. The space within a cut is called its *close* or *area*. So the area of a cut c contains vertices, edges and other cuts, even if they are deeper nested inside other cuts, but not the cut c itself. All the edges, vertices and cuts in the area of c are said to be *enclosed by* c.

Cuts do not intersect each other by the definition of Peirce. So for two different cuts c_1, c_2, exactly one of the following cases occurs:

- c_1 and its area is entirely enclosed by c_2,
- c_2 and its area is entirely enclosed by c_1,
- c_1 and its area and c_2 and its area have nothing in common.

Obvioulsy, these three cases coincide with the three conditions for the mapping $area$ in Definition 2. Now, let us first mention some simple properties for the mapping $area$ which can be shown easily:

- $c_1 \neq c_2 \wedge area(c_1) = area(c_2) \Longrightarrow area(c_1) = area(c_2) = \emptyset$
- $\emptyset \subsetneq area(c_1) \subsetneq area(c_2) \Longrightarrow c_1 \in area(c_2)$
- $c_1 \in area(c_2) \Longrightarrow area(c_1) \subseteq area(c_2)$

In many cases it makes sense to treat the outermost context, the sheet of assertion, as an (additional) cut. If we abbreviate the sheet of assertion by $\top$, we immediately come to the following definition:

Definition 3. *If* Cut *is a set of cuts of a directed multi-hypergraph with cuts and if* $area$ *is the appropriate mapping, then let* $Cut^\top := Cut \dot\cup \{\top\}$ *and* $area(\top) := V \cup E \cup Cut$.

It is easy to see that this extension still satisfies the conditions for the mapping $area$ which are given in Definition 2. This means that the properties we have just shown for Cut hold for $Cut^\top$, too.

By $c_1 \leq c_2 :\Longleftrightarrow c_1 \in area(c_2)$ a canonical ordering on $Cut^\top$, which is a tree with $\top$ as greatest element, is defined. This can be verified with the properties for the mapping $area$.

Obviously, each edge and vertex is enclosed *directly* (and not deeper nested) in a uniquely given cut c. For the further work, the notion of a *subgraph* is needed. It seems to be evident that a subgraph is enclosed directly in a uniquely given cut c, too. The notions of being *directly enclosed* and *subgraph* shall become precise through the following definition:

Definition 4. *Let* $\mathfrak{G} = (V, E, \nu, Cut, area)$ *be a directed multi-hypergraph with cuts.*

- *For each* $k \in V \cup E \cup Cut$ *we define*

$$cut(k) := min\{c \in Cut^\top \mid k \in area(c)\}$$

  $cut(k)$ *is called* the cut of k *and* $cut(k)$ *is said to* enclose directly *the vertex (the edge, the cut)* k.
- *The graph* $\mathfrak{G}' = (V', E', \nu', Cut', area')$ *is called a* subgraph of $\mathfrak{G}$ in the cut c *if* $c \in Cut^\top$ *is the smallest cut such that the following conditions hold:*
  - $V' \subseteq V, E' \subseteq E, Cut' \subseteq Cut$ *and the mappings* ν' *and* $area'$ *are just the restrictions of* ν *and* $area$ *to* E' *resp.* Cut' *(and are therefore well defined),*
  - $area(c') \subseteq V' \cup E' \cup Cut'$ *for each* $c' \in Cut'$,
  - $cut(k') \in Cut' \cup \{c\}$ *for each* $k' \in V' \cup E' \cup Cut'$,
  - $v \in V'$ *for each edge* $e' \in E'$ *and every vertex* $v \in V_e$.

  We write: $\mathfrak{G}' \subseteq \mathfrak{G}$ *and* $cut(\mathfrak{G}') = c$.

Note, that for each vertex (or edge, cut or subgraph), the set of all cuts containing the vertex forms a chain. If the number of cuts enclosing the vertex is even, the edge is said to be *evenly enclosed*, and analogously, if the number is odd, the vertex is said to be *oddly enclosed*. More formally:

Definition 5. *Let* $\mathfrak{G} = (V, E, \nu, Cut, area)$ *be a directed multi-hypergraph with cuts, let* k *be a subgraph or an element of* $V \cup E \cup Cut^\top$. *Let* n *be the number of cuts which enclose* k *(*$n := |\{c \in Cut \mid c \in area(c)\}|$*). If* n *is even,* k *is said to be* evenly enclosed, *otherwise* k *is said to be* oddly enclosed. *An evenly enclosed cut is called* positive, *an oddly enclosed cut is called* negative.

Now, the structure of simple concept graphs with cuts is derived from the structure of directed multi-hypergraphs with cuts by additionally labeling the

vertices and edges with concept names and relation names, respectively, and by assigning a reference to each vertex. In particular all definitions concerning directed multi-hypergraphs with cuts can be transferred to concept graphs. So in the following we will deal with subgraphs of concept graphs etc.

Definition 6. *A* (nonexistential) simple concept graph with cuts over the alphabet $\mathcal{A}$ *is a structure* $\mathfrak{G} := (V, E, \nu, Cut, area, \kappa, \rho)$, *where*

- $(V, E, \nu, Cut, area)$ *is a directed multi-hypergraph with cuts*
- $\kappa : V \cup E \to \mathcal{C} \cup \mathcal{R}$ *is a mapping such that* $\kappa(V) \subseteq \mathcal{C}$ *and* $\kappa(E) \subseteq \mathcal{R}$, *and all* $e \in E$ *with* $v(e) = (v_1, \ldots, v_k)$ *satisfy* $\kappa(e) \in \mathcal{R}_k$
- $\rho : V \to \mathcal{G}$ *is a mapping*

It is not clear what a graph containing vertices with more than one object, enclosed by a cut, means, and this might lead to misunderstandings. For this reason, the mapping ρ maps vertices only to elements of $\mathcal{G}$, not to subsets of $\mathcal{G}$ (in contrast to the definition of Prediger in [Pr98b]). Furthermore, ρ can be naturally extended to the edges: If e is an edge with $v(e) = (v_1, \ldots, v_k)$, let $\rho(e) := (\rho(v_1), \ldots, \rho(v_n))$.

3 Semantics

Usually, a semantics for conceptual graphs is given by a translation of graphs into formulas of first order logic, hence into formulas of another syntactically given structure. In Prediger (cf. [Pr98a], [Pr98b]), a different approach is presented. There, an extensional semantics which is based on power context families as model structures is introduced. The motivation for this *contextual semantics* can be read in [Pr98a]. With this semantics, Prediger develops a semantical entailment relation between concept graphs, and a sound and complete calculus for this entailment relation is presented. Now this approach shall be extended to concept graphs with cuts.

In concept graphs without cuts, only the conjunction of positive information can be expressed. For this reason it was possible for Prediger to construct for each concept graph a standard model in which all the information of the concept graph is encoded. Standard models have been an additional possibility (besides the entailment relation and the calculus) for doing reasoning with concept graph. If negations are used, one can express with concept graphs the disjunction of pieces of information. But disjunction of information can not be canonically encoded in standard models. Thus if we introduce negations to concept graphs, unfortunately the construction of standard models has to be dropped.

Now, let us recall the basic definitions of Prediger.

Definition 7. *A* power context family $\vec{\mathbb{K}} := (\mathbb{K}_0, \ldots, \mathbb{K}_n)$ *of type* n *(for an* $n \in \mathbb{N}$*) is a family of contexts* $\mathbb{K}_k := (G_k, M_k, I_k)$ *that satisfies* $G_k \subseteq (G_0)^k$ *for each* $k = 1, \ldots, n$. *Then we write* $\vec{\mathbb{K}} := (G_k, M_k, I_k)_{k=0,\ldots,n}$. *The elements of the set* $\mathfrak{R}_{\vec{\mathbb{K}}} := \bigcup_{k=1}^{n} \mathfrak{B}(\mathbb{K}_k)$ *are called* relation-concepts.

Interpreting a concept graph in a power context family, the object names will be interpreted by objects, e.g. by elements of the set G_0. The concept names of our alphabet will be interpreted by concepts in the context $\mathbb{K}_0$, and relation names of arity k will be interpreted by relation-concepts in the context $\mathbb{K}_k$. Of course, every reasonable interpretation has to respect the orders on the names. This leads to the following definition:

Definition 8. *For an alphabet* $\mathcal{A} := (\mathcal{G}, \mathcal{C}, \mathcal{R})$ *and a power context family* $\vec{\mathbb{K}}$, *we call the union* $\lambda := \lambda_{\mathcal{G}} \dot\cup \lambda_{\mathcal{C}} \dot\cup \lambda_{\mathcal{R}}$ *of the mappings* $\lambda_{\mathcal{G}} : \mathcal{G} \to G_0$, $\lambda_{\mathcal{C}} : \mathcal{C} \to \underline{\mathfrak{B}}(\mathbb{K}_0)$ *and* $\lambda_{\mathcal{R}} : \mathcal{R} \to \mathfrak{R}_{\vec{\mathbb{K}}}$ *a* $\vec{\mathbb{K}}$-interpretation *of* $\mathcal{A}$, *if* $\lambda_{\mathcal{C}}$ *and* $\lambda_{\mathcal{R}}$ *are order-preserving,* $\lambda_{\mathcal{C}}(\top) = \top$, $\lambda_{\mathcal{R}}(\mathcal{R}_k) \subseteq \underline{\mathfrak{B}}(\mathbb{K}_k)$ *for all* $k = 1, \ldots, n$, *and* $(g_1, g_2) \in \lambda_{\mathcal{R}}(id) \Leftrightarrow g_1 = g_2$ *for all* $g_1, g_2 \in \mathcal{G}$ *hold. The tupel* $(\vec{\mathbb{K}}, \lambda)$ *is called* context-interpretation *of* $\mathcal{A}$ *or, according to classical logic,* $\mathcal{A}$-structure.

Recall that we defined $\rho(e) := (\rho(v_1), \ldots, \rho(v_n))$ for edges e with $v(e) = (v_1, \ldots, v_k)$. Because $\lambda_{\mathcal{G}}$ is a mapping on the set $\mathcal{G}$ of object names, it can be naturally extended to tupels of object names. In particular we get $\lambda_{\mathcal{G}}(\rho(e)) := (\lambda_{\mathcal{G}}(\rho(v_1)), \ldots, \lambda_{\mathcal{G}}(\rho(v_n)))$.

Now we can define whether a concept graph is valid in an $\mathcal{A}$-structure. This is done in a canonical way:

Definition 9. *Let* $\vec{\mathbb{K}}$ *be a power context family and let* $\mathfrak{G}$ *be a concept graph. Inductively over* $c \in Cut^{\top}$, *we define* $\vec{\mathbb{K}} \models \mathfrak{G}[c]$ *in a canonical way:*

$\vec{\mathbb{K}} \models \mathfrak{G}[c] \Longleftrightarrow$

- $\lambda_{\mathcal{G}}(\rho(v)) \in Ext(\lambda_{\mathcal{C}}(\kappa(v)))$ *for each* $v \in V$ *with* $cut(v) = c$ *(vertex condition)*
- $\lambda_{\mathcal{G}}(\rho(e)) \subseteq Ext(\lambda_{\mathcal{R}}(\kappa(e)))$ *for each* $e \in E$ *with* $cut(e) = c$ *(edge condition)*
- $\vec{\mathbb{K}} \not\models \mathfrak{G}[c']$ *for each* $c' \in Cut$ *with* $cut(c') = c$ *(iteration over* $Cut^{\top}$*)*

For $\vec{\mathbb{K}} \models \mathfrak{G}[\top]$ *we write* $\vec{\mathbb{K}} \models \mathfrak{G}$.

If we have two concept graphs $\mathfrak{G}_1$, $\mathfrak{G}_2$ *such that* $\vec{\mathbb{K}} \models \mathfrak{G}_2$ *for each* $\mathcal{A}$*-structure with* $\vec{\mathbb{K}} \models \mathfrak{G}_1$, *we write* $\mathfrak{G}_1 \models \mathfrak{G}_2$.

Intuitively, $\vec{\mathbb{K}} \models \mathfrak{G}[c]$ can be read as $\vec{\mathbb{K}} \models \mathfrak{G}\big|_{area(c)}$. But note that generally $area(c)$ is not a subgraph of $\mathfrak{G}$. Therefore this should not be understood as a precise definition.

4 Calculus

The following calculus is based on the α-calculus of Peirce for existential graphs without lines of identity. These existential graphs consist only of propositional variables and ovals and are equivalent to propositional calculus.

For the sake of intelligibility, the whole calculus is described using common spoken language. Only the rules 'erasure', 'iteration', and 'merging two vertices' will be described in a mathematically precise manner to show that using full

sentences does not imply the loss of precision. This precision is definitely necessary because there must not be any possibility for misunderstandings of the rules. The rule 'iteration' for example, says that a subgraph of a graph can be copied into the same or a nested context. If this is to have a unique meaning, one requires a precise definition of 'subgraph' and 'same or nested context'.

First, we present the whole calculus. The first five rules of the calculus are the original rules of Peirce's α-calculus. The further rules are needed to encompass the orders on the concept- and relation names, to encompass the special properties of the concept name $\top$ and the relation name id and to deal with the possibility that different vertices can have the same reference.

Definition 10. *The calculus for (nonexistential) simple concept graph with cuts over the alphabet $\mathcal{A}$.*

- **erasure**
  In positive cuts, any directly enclosed edge, isolated vertex and closed subgraph may be erased.
- **insertion**
  In negative cuts, any directly enclosed edge, isolated vertex and closed subgraph may be inserted.
- **iteration**
  Let $\mathfrak{G}_\circ := (V_0, E_0, \nu_0, \kappa_0, \rho_0, Cut_0)$ be a subgraph of $\mathfrak{G}$ and let $\bar{c} \leq cut(\mathfrak{G}_\circ)$ be a cut such that $\bar{c} \notin Cut_0$. Then a copy of $\mathfrak{G}_0$ may be inserted into $\bar{c}$.
- **deiteration**
  If $\mathfrak{G}_0$ is a subgraph of $\mathfrak{G}$ which could have been inserted by rule of iteration, then it may be erased.
- **double negation**
  Double cuts (two cuts c_1, c_2 with $cut^{-1}(c_2) = \{c_1\}$) may be inserted or erased.
- **isomorphism**
  A graph may be substituted by an isomorphic copy of itself.
- **generalization**
  For evenly enclosed vertices and edges the concept names respectively relation names may be generalized.
- **specialization**
  For oddly enclosed vertices and edges the concept names respectively relation names may be specialized.
- **$\top$-rule**
  For each object name g, an isolated vertex $\boxed{\top : g}$ may be inserted or erased in arbitrary cuts.
- **merging two vertices**
  For each object name g, a vertex $\boxed{\top : g}$ may be merged into a vertex $\boxed{P : g}$ (i.e. $\boxed{\top : g}$ is erased and, for every edge e, $e(i) = \boxed{\top : g}$ is substituted by $e(i) = \boxed{P : g}$).
  Two vertices in the same cut and with the same reference may be merged.
- **reverse merging of two vertices**
  A merging of two vertices may be reversed.
- **rules of identity**
  - *reflexivity*
    For arbitrary vertices v edges e with $\kappa(e) = id$, $cut(e) = cut(v)$ and $e\big|_1 = e\big|_2 = v$ may be inserted or erased.
  - *symmetry*
    If e is an edge with $\kappa(e) = id$, then e may be substituted by an edge e' which fulfills $e'\big|_1 = e\big|_2$, $e'\big|_2 = e\big|_1$ and $cut(e') = cut(e)$.
  - *transitivity*
    If e_1, e_2 are two edges with $\kappa(e_1) = \kappa(e_2) = id$, $cut(e_1) = cut(e_2)$ and $e_1\big|_2 = e_2\big|_1$, then edges e with $\kappa(e) = id$, $cut(e) = cut(e_1)$, $e\big|_1 = e_1\big|_1$ and $e\big|_2 = e_2\big|_2$ may be inserted or erased.
  - *congruence*
    If e is an edge with $\rho(e\big|_1) = g_1$, $\rho(e\big|_1) = g_2$ and $\kappa(e) = id$, then $\rho(e\big|_1) = g_1$ may be substituted by $\rho(e\big|_1) = g_2$.

To see how these rules can be written down mathematically, here are the precise definitions for the rules 'erasure', 'iteration' and 'merging two vertices'.

- If $\mathfrak{G} := (V, E, \nu, \kappa, \rho, Cut)$ is a concept graph with the closed subgraph $\mathfrak{G}_\circ := (V_0, E_0, \nu_0, \kappa_0, \rho_0, Cut_0)$ and if $\bar{c}$ is a cut with $\bar{c} \notin Cut_0$, then let $\mathfrak{G}'$ be the following graph:
  - $V' := V \times \{1\} \cup V_0 \times \{2\}$
  - $E' := E \times \{1\} \cup E_0 \times \{2\}$
  - $\nu'((e,i)) = ((v_1,i), \ldots, (v_n,i))$ for $(e,i) \in E'$ and $\nu(e) = (v_1, \ldots, v_n)$
  - $\kappa'((e,i)) := \kappa(e)$ and $\kappa'((v,i)) := \kappa(v)$ for all $(e,i) \in E'$, $(v,i) \in V'$
  - $\rho'((v,i)) = \rho(v)$ for all $(v,i) \in V'$
  - $Cut' := Cut \times \{1\} \cup Cut_0 \times \{2\}$
  - $area'$ is defined as follows: Let $c \in Cut$.
    for $c \in Cut_0$ let $area'((c,2)) := area(c) \times \{2\}$
    for $c \not\geq \bar{c}$ let $area'((c,1)) := area(c) \times \{1\}$
    for $c \geq \bar{c}$ let $area'((c,1)) := area(c) \times \{1\} \cup (V_0 \cup E_0 \cup Cut_0) \times \{2\}$

  Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *iterating the subgraph $\mathfrak{G}_0$ into the cut $\bar{c}$*.
- If $\mathfrak{G} := (V, E, \nu, \kappa, \rho, Cut)$ is a concept graph with the closed subgraph $\mathfrak{G}_\circ := (V_0, E_0, \nu_0, \kappa_0, \rho_0, Cut_0)$, then let $\mathfrak{G}'$ be the following graph:
  - $V' := V \backslash V_0$
  - $E' := E \backslash E_0$
  - $\nu' := \nu\big|_{E'}$
  - $\kappa' := \kappa\big|_{V' \cup E'}$
  - $\rho' := \rho\big|_{V'}$
  - $Cut' := Cut \backslash Cut_0$
  - $area'(c') := area(c')\big|_{V' \cup E' Cut'}$

  Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *erasing the subgraph $\mathfrak{G}_0$*.
- If $\mathfrak{G} := (V, E, \nu, \kappa, \rho, Cut)$ is a concept graph with two vertices $v_1, v_2 \in V$, then let $\mathfrak{G}'$ be the following graph:
  - $V' := V \backslash \{v_1\}$

- $E' := E$
- ν' is defined as follows: For $\nu(e)\big|_i = v$ let $\nu'(e)\big|_i = \begin{cases} v & v \neq v_1 \\ v_2 & v = v_1 \end{cases}$.
- $\kappa' := \kappa|_{V' \cup E}$
- $\rho' := \rho|_{V'}$
- $Cut' = Cut$
- $area'(c') := area(c')\big|_{V' \cup E \cup Cut}$

Then we say that $\mathfrak{G}'$ is derived from $\mathfrak{G}$ by *merging* v_2 *into* v_1.

These rules are sound and complete with respect to the given semantics (see Theorem 1). Instead of proving this theorem formally, some heuristics for the rules are presented.

First note, that all the rules are in some sense dually symmetric with respect to positive and negative cuts. More precisely, every rule which can be applied in one direction in positive cuts can be applied in the opposite direction in negative cuts, and vice versa. So if a rule can only be applied in positive contexts, this rule has a counterpart for negative contexts (like erasure and insertion or like generalization and specialization). All other rules apply both to positive and negative contexts.

The first five rules are sound and complete concerning the classical propositional calculus. If all vertices and edges would be understood as logically independent propositional variables, these rules would be enough. The rules 'generalization', 'specialization' and '$\top$-rule' encompass the orders on the concept and relation names. Note that $\top$ is not only the greatest element of all concepts: The semantics for $\top$ implies that every object belongs to the extension of the concept $\top$. Thus the generalization rule does not encompass all properties of the concept $\top$, and the $\top$-rule is necessary. The same is true for the relation id. In fact it is a congruence relation by definition. This is encompassed by the id-rules. The specialization rule can be derived from the other rules, but it is added to keep the calculus symmetric. The rules 'merging two vertices' and 'reverse merging of two vertices' deal with the fact that one object may be the reference for different vertices. With these rules it is possible to transform every concept graph into an equivalent graph in which no cut intersects a relation line. More precisely:

Definition 11. *A concept graph is called* free of intersections, *if it fulfills the following condition:* $\forall e \in E\, \forall v \in V : v \in V_e \Longrightarrow cut(v) = cut(e)$

It follows from the rules 'merging two vertices' and 'reverse merging of two vertices' that every concept graph is equivalent to a graph free of intersections. And these graphs are easy to read: They have a form which is closely related to the existential graphs without lines of identity, and the soundness and completeness of the first five rules concerning existential graphs can be applied now. This leads to the following essential theorem:

Theorem 1 (soundness and completeness of the calculus).
Two nonexistential, simple concept graph $\mathfrak{G}_1$, $\mathfrak{G}_2$ *with cuts over* $\mathcal{A}$ *satisfy*

$$\mathfrak{G}_1 \vdash \mathfrak{G}_2 \quad \Longleftrightarrow \quad \mathfrak{G}_1 \models \mathfrak{G}_2$$

5 Future Work

How to procede with this work is clear. First the approach has to be extended to include graphs with generic markers. The Φ-operator for these graphs has to be elaborated and it has to be proven that simple concept graphs with negation ovals and identity are equivalent to first order logic. In part, this has already been done (e.g. [BMT98]). Afterwards, the approach should be extended to the nested case. It seems reasonable that nested graphs are equivalent to a certain class of formulas of modal logic in such a way that nestings will be interpreted as different possible worlds, which are connected by the structure of the nestings. And again, a semantics and a sound and complete calculus have to be developed.

References

[BMT98] F. Baader, R. Molitur, S. Tobies: The Guarded Fragment of Conceptual Graphs. RWTH LTCS-Report. `http://www-lti.informatik.rwth-aachen.de/Forschung/Papers.html`

[CMS98] M. Chein, M.-L. Mugnier, G. Simonet: Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics, Rapport de Recherche, LIRMM, Université Montpellier II, 1998.

[GW99a] B. Ganter, R. Wille: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin-Heidelberg-New York 1999.

[GW99b] B. Ganter, R. Wille: Contextual attribute logic, in: W. Tepfenhart, W. Cyre (Eds.): Conceptual Structures: Standards and Practices, Springer Verlag, Berlin–New York 1999, 377-388.

[LK96] D. Lukose, R. Kremer: Knowledge Engineering: PART A, Knowledge Representation. `http://www.cpsc.ucalgary.ca/~kremer/courses/CG/`

[Pe98] C. S. Peirce: Reasoning and the Logic of Things. The Cambridge Conferences Lectures of 1898. Ed. by K. L. Kremer, Harvard Univ. Press, Cambridge 1992

[Pr98a] S. Prediger: Kontextuelle Urteilslogik mit Begriffsgraphen. Ein Beitrag zur Restrukturierung der mathematischen Logik, Shaker Verlag 1998.

[Pr98b] S. Prediger: Simple Concept Graphs: A Logic Approach, in: M, -L. Mugnier, M. Chein (Eds.): Conceptual Structures: Theory, Tools and Applications, Springer Verlag, Berlin–New York 1998, 225–239.

[Ro73] D. D. Roberts: The Existential Graphs of Charles Sanders Peirce, Mouton The Hague – Paris 1973.

[So84] J. F. Sowa: Conceptual Structures: Information Processing in Mind and Machine. Addison Wesley Publishing Company Reading, 1984.

[So92] J. F. Sowa: Conceptual Graphs Summary, in: T. E. Nagle, J. A. Nagle, L. L. Gerholz, P. W. Eklund (Eds.): Conceptual Structures: current research and practice, Ellis Horwood, 1992, 3–51.

[So99] J. F. Sowa: Conceptual Graphs: Draft Proposed American National Standard, in: W. Tepfenhart, W. Cyre (Eds.): Conceptual Structures: Standards and Practices, Springer Verlag, Berlin–New York 1999, 1-65.

[So00] J. F. Sowa: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[We95] M. Wermelinger: Conceptual Graphs and First-Order Logic, in: G. Ellis et al. (Eds.): Conceptual Structures: Applications, Implementations and Theory, Springer Verlag, Berlin–New York 1995, 323–337.