2007

A Peirce-style calculus for ALC

Frithjof Dau
dau@uow.edu.au

Peter Eklund
University of Wollongong, peklund@uow.edu.au

A Peirce-style calculus for ALC

Abstract

Description logics (DLs) are a well-understood family of knowledge representation (KR) languages. The notation of DLs has the flavour of a variable-free first order predicate logic. In this paper, a diagrammatic representation of the DL *ALC*, based on Peirce's existential graphs, is presented, and a set of transformation rules on these graphs is provided. It is proven that these rules form a sound and complete diagrammatic calculus for *ALC*.

Publication Details

Dau, F. & Eklund, P. W. (2007). A Pierce Style Calculus for ALC. In P. Cox (Eds.), Proceedings of the VLL 2007 workshop on Visual Languages and Logic (pp. 55-71). USA: CEUR-WS.

A Peirce-style calculus for $\mathcal{ALC}$

Frithjof Dau and Peter Eklund *
Faculty of Informatics
University of Wollongong
Wollongong, Australia

Abstract

Description logics (DLs) are a well-understood family of knowledge representation (KR) languages. The notation of DLs has the flavour of a variable-free first order predicate logic. In this paper, a diagrammatic representation of the DL $\mathcal{ALC}$, based on Peirce's existential graphs, is presented, and a set of transformation rules on these graphs is provided. It is proven that these rules form a sound and complete diagrammatic calculus for $\mathcal{ALC}$.

Keywords: Existential graphs, relation graphs, description logics, diagrammatic reasoning, calculus, soundness, completeness

1 Introduction

Description logics (DLs) are a well-understood family of knowledge representation (KR) languages tailored to express knowledge about concepts and concept hierarchies that have gained widespread use. The basic building blocks of DLs are concepts, roles and sometimes individuals, which can be composed by language constructs such as intersection, union, value or number restrictions and more to build more complex well-formed formulas that themselves represent more complex concepts and roles. For example, if MAN, FEMALE, MALE, RICH, HAPPY are concepts and if HASCHILD is a role, we can define

$$\begin{aligned} \text{HM} \quad \equiv \quad & \text{MAN} \sqcap \exists \text{HASCHILD}.\text{FEMALE} \sqcap \exists \text{HASCHILD}.\text{MALE} \\ & \sqcap \forall \text{HASCHILD}.(\text{RICH} \sqcup \text{HAPPY}) \end{aligned}$$

which defines the concept of men who have both male and female children, and where all children are rich or happy (HM abbreviates HAPPYMAN).

The formal notation of DLs has the flavour of a variable-free first order predicate logic (FOL). In fact, DLs correspond to decidable fragments of FOL. Like FOL, DLs have a well-defined, formal syntax and Tarski-style semantics, and they provide sound and complete inference facilities. The variable-free notation of DLs makes them easier to comprehend than the common FOL formulas that include variables. Nevertheless, without training, the symbolic notation of FOL can be hard to learn and comprehend.

It has been argued that diagrams are useful for KR systems [9, 12, 13], a fact that has been acknowledged by the DLs authors (see introduction of [1]). Therefore one

*Email:dau,peklund@uow.edu.au

alternative to DLs symbolic notation is the development of a diagrammatic representation. A first attempt can be found in [9], where a graph-based representation for the textual DL CLASSIC is elaborated. In [3], a specific DL is mapped to the diagrammatic system of conceptual graphs [17]. In [2], a UML-based representation for a DL is provided. In these approaches, the focus is on a graphical *representation* of DL, however, as emphasized in many works on DL, *reasoning* is a distinguishing feature of DL. Correspondences between graphical representation of the DL and the DL reasoning system are therefore important inclusions in any graphical representation but to date they have remain largely unelaborated.

This paper presents a diagrammatic representation of the DL $\mathcal{ALC}$ in the style of Peirce's existential graphs (EGs) [18, 15, 16, 6] (the reasons for choosing Peirce's graphs as a diagrammatic framework for DL are presented [8]). An adequate diagrammatic calculus for $\mathcal{ALC}$, based on Peirce's calculus for his graphs, is provided.

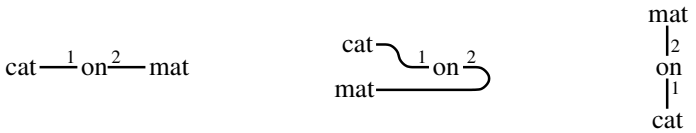
Reasoning with DLs is usually carried out by means of tableau algorithms. The calculus of this paper differs significantly from this approach in two respects. First, the rules of the calculus are *deep-inference* rules, as they modify deep nested subformulas, whereas tableau algorithms (similar to other common calculi) only modify formulas at their top-level (some interesting aspects of Peirce's rules in terms of proof-theory are investigated in [7]). More importantly for this workshop, the rules can be best understood to modify the diagrammatic Peirce-style representations of $\mathcal{ALC}$, i.e., the calculus is a *diagrammatic* calculus.

The paper is structured as follows. We assume that the reader has some familiarity with the system of EGs. In Section 1.1, only a very short introduction is provided. Thorough introductions can be found in [18, 15, 16, 6]. In Section 2 the syntax and semantics of the DL $\mathcal{ALC}$ as we use it in this paper is introduced. In Section 3, the diagrammatic calculus for $\mathcal{ALC}$ is presented, and its soundness and completeness is proven. The final section provides a discussion of this research.

1.1 Existential Graphs

Existential graphs [10] are a diagrammatic logic invented by C.S. Peirce (1839-1914) in the last two decades of his life. Existential graphs are divided into three parts called Alpha, Beta and Gamma. The three parts build on one another, Beta builds upon Alpha, and Gamma builds on both Alpha and Beta. Alpha corresponds to propositional logic, Beta corresponds to FOL (to be precise: first order logic with predicates and equality, but without functions or constants). Gamma encompasses features of higher order logic, including modal logic, self-reference and more. In contrast to Alpha and Beta, Gamma was never finished by Peirce, and even now, only fragments of Gamma (mainly the modal logic part) are elaborated to contemporary mathematical standards. In this section, only Beta is introduced.

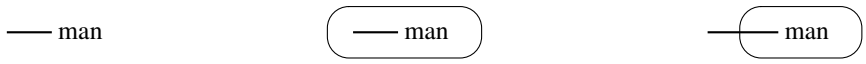
The EGs of Beta consist of predicate names of arbitrary arities, of heavily drawn lines which are both used to express existential quantification and identity, and of closed, doublepoint-free curves which are called CUTS (or sometimes SEPS) and used to negate the enclosed subgraph. We start with a very simple graph expressing 'a cat is on a mat'. Below, three different diagrams of the graph are depicted.



Each diagram contains two heavily drawn lines. In this case, they do not cross cuts or do not have branching points (see below for an explanation of cuts and branching points). This simplest form of a heavily drawn line is called LINE OF IDENTITY. The two lines of identity denote two (not necessarily different) objects. The first line of identity is attached to the unary predicate 'cat', hence the first object denotes a cat. Analogously, the second line of identity denotes a mat. Both lines are attached to the dyadic predicate 'on', i.e. the first object (the cat) stands in the relation 'on' to the second object (the mat). The meaning of the graph is therefore 'there is a cat and a mat such that the cat is on the mat', or in short: A cat is on a mat.

The three different diagrams are different representations of the same EG. In order to distinguish graphs from their diagrams, Peirce coined the term *graph* and *graph replica*, i.e., the above diagrams are different graph replicas of the same EG. Similar distinctions are made between *types* and *tokens*, known from philosophy, or *abstract* and *concrete* syntax, used widely in Computer Science. As discussed in [11, 4], for a formally precise elaboration of any logic by means of diagrams, this distinction is vital. This approach is adopted in this paper as well. In Section 2, a fragment of Peirce's graphs is used as a diagrammatic system for the DL $\mathcal{ALC}$. In this section, the syntax of this fragment is defined on a abstract level which prescinds from the topological properties of the diagrammatic representations.

In the next graphs, the cuts which are used to express negation are introduced.



The meaning of the first graph is 'there is a man'. The second graph is built from the first graph by drawing a cut around it, i.e. the first graph is denied. Hence the meaning of the second graph is 'it is not true that there is a man', i.e. 'there is no man'. In the third graph, the heavily drawn line (which is not a line of identity, as it crosses a cut) begins on the sheet of assertion. Hence, the existence of the object is asserted, not denied. For this reason the meaning of the third graph is 'there is something which is not a man'.

Peirce writes in [10], in 4.116 (we adopt the usual convention to refer to his collected papers), a "line of identity is [. . .] a heavy line with two ends and without other topical singularity (such as a point of branching or a node), not in contact with any other sign except at its extremities." So lines of identity do not have any branching points, nor are they are allowed to *cross* cuts. However, by connecting them at their endpoints, we can obtain networks of lines of identity, which are termed LIGATURES. Peirce allows only two or three lines of identity to be connected. If three lines of identity are connected, the point where they meet is called a BRANCHING POINT. Moreover, it is possible to connect lines of identity connect directly on a cut. Due to this possibility, ligatures are permitted which cross a cut.

Let us now consider the three EGs of Fig. 1, where ligatures are used.

In the first graph, the ligature consists of three lines of identity, which meet in a branching point, in the second graph, the ligature consists of two lines of identity meeting on the cut, and the ligature in the third graph is composed of seven lines of identity. Nonetheless, in all these graphs, a ligature can, similar to a line of identity, be understood to denote a single object. The meaning of the graphs of Figure 1 'there exists a male, human african', 'there exists a man who will not die', and 'it is not true that there is a pet cat such that it is not true that it is not lonely and owned by somebody', i.e., 'every pet cat is owned by someone and is not lonely'.

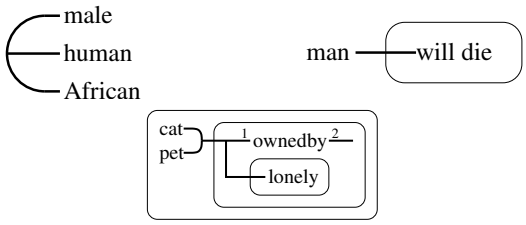


Figure 1: Four Peirce graphs with ligatures which do not traverse cuts

Nonetheless, other examples show that this interpretation of ligatures is not so simple in every case: namely a ligature may stand for more than one object. Let us consider the three EGs of Figure 2. These graphs have the meanings 'there are at least two suns', 'there are (not necessarily distinct) objects which are blue, red, large and small, respectively', and 'the blue and large or the red and small object are distinct', and 'there are objects o_1, o_2, o_3 with the properties S, P, and T resp, and these objects are not all identical' (i.e., $o_1 = o_2 = o_3$ does not hold). In every graphs, there is not a single ligature that can be understood to denote a single object.

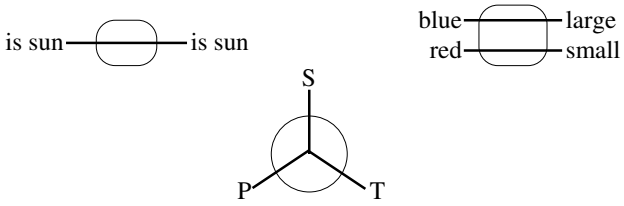


Figure 2: Three Peirce graphs with ligatures which traverse cuts

In every graph in Figure 2 a part of a ligature traverses a cut (i.e., there is a cut c and a heavily drawn line l which is part of the ligature such that both endpoints of l are placed on c and the remainder of l is enclosed by c). Such a device denotes non-identity of the endpoints of l. A complete discussion of ligatures in existential graphs goes beyond the scope of this paper, see [5] for a more detailed discussion. It will turn out that in the EGs we have to deal with in this paper, no ligature traverses a cut, so we will not run into problems caused by the kind of ligatures of we see in Figure 2.

We now have all the necessary elements to express existential quantification, predicates of arbitrary arities, conjunction and negation. As such we see that the Beta part of EGs corresponds to FOL (without object names and without function names). Moreover, Peirce equipped EGs with a set of five sound and complete inference rules. We do not address these rules in this section. For the $\mathcal{ALC}$-fragment of EGs, they will be introduced in Section 3.

2 The Description Logic $\mathcal{ALC}$

The vocabulary $(\mathcal{A}, \mathcal{R})$ of a DL consists of a set $\mathcal{A}$ of (ATOMIC) CONCEPTS, which denote sets of individuals, and a set $\mathcal{R}$ (ATOMIC) ROLES, which denote binary relationships between individuals. Moreover, we usually consider vocabularies that include the universal concept $\top$. From these atomic items, more complex concepts and roles are

built with constructs such as intersection, union, value and number restrictions, etc. For example, if C, C_1, C_2 are concepts, then so are $C_1 \sqcap C_2$, $\neg C$, $\forall R.C$, and $\exists R.C$ (these constructors are called conjunction, negation, value restriction, and exists restriction).

In this paper, we focus on the description logic $\mathcal{ALC}$, which is the smallest propositionally closed description logic. For our purpose, we consider $\mathcal{ALC}$ to be composed of conjunction, negation and existential restriction. In contrast to the usual approach, in this paper the concepts of $\mathcal{ALC}$ are introduced as labelled trees. This is more convenient for defining the rules of the calculus, and the trees are already close to Peirce's notion of graphs.

An INTERPRETATION is a pair $(\Delta^{\mathcal{I}}, \mathcal{I})$, consisting of an nonempty DOMAIN OF THE INTERPRETATION $\Delta^{\mathcal{I}}$ and INTERPRETATION FUNCTION $\mathcal{I}$ which assigns to every $A \in \mathcal{A}$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to role $R \in \mathcal{R}$ a relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We require $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$.

Trees can be formalized either as rooted and acyclic graphs, or as special posets. We adopt the second approach, i.e., a tree is a poset $(T, \geq)$, where $s \geq t$ can be understood as 's is an ancestor of t'. A LABELLED TREE is a structure $\mathbf{T} := (T, \leq, \nu)$, where $(T, \leq)$ is a tree and $\nu : T \to L$ is a mapping from the set of nodes to some set L of labels. The greatest element of T is the ROOT of the tree. As usual, each node v gives rise to a SUBTREE $\mathbf{T}_v$ (formally, $\mathbf{T}_v = (T_v, \geq \big|_{T_v \times T_v}, \nu|_{T_v})$ with $T_v := \{w \in T \mid v \geq w\}$). We write $\mathbf{T}' \subseteq \mathbf{T}$, if $\mathbf{T}'$ is a subtree of $\mathbf{T}$. Isomorphic labelled trees are implicitly identified.

Next we introduce some operations to inductively construct labelled trees. These operations will be used to define the syntax and semantics of $\mathcal{ALC}$ based on labelled trees. We assume to have a set L of labels.

Chain: Let $l_1, \ldots, l_n \in L$. With $l_1\, l_2\, \ldots l_n$ we denote the labelled tree $\mathbf{T} := (T, \geq, \nu)$ with $T := \{v_1, \ldots, v_n\}$, $v_1 > v_2 > \ldots > v_n$ and $\nu(v_1) = l_1, \ldots, \nu(v_n) = l_n$. That is, $l_1\, l_2\, \ldots l_n$ denotes a CHAIN, where the nodes are labelled with l_1, l_2, $\ldots$, l_n, respectively. We extent this notation by allowing the last element to be a labelled tree: If $l_1\, l_2\, \ldots l_n \in L$ and if $\mathbf{T}'$ is a labelled tree, then $l_1\, l_2\, \ldots l_n \mathbf{T}'$ denotes the labelled tree $\mathbf{T} := (T, \geq, \nu)$ with $T := T' \cup \{v_1, \ldots, v_n\}$, $v_1 > v_2 > \ldots > v_n$ and $v_i > v$ for each $i = 1, \ldots, n$ and $v \in T'$, and $\nu := \nu' \cup \{(v_1, l_1), \ldots, (v_n, l_n)\}$. That is, $\mathbf{T}$ is obtained by placing the chain $l_1\, l_2\, \ldots l_n$ above $\mathbf{T}'$.

Subsitution: Let $\mathbf{T}_1, \mathbf{T}_2$ be labelled trees and $\mathbf{S} := (S, \geq_s, \nu_s)$ a subtree of $\mathbf{T}_1$. Then $\mathbf{T} := \mathbf{T}_1[\,\mathbf{T}_2 \,/\, \mathbf{S}\,]$ denotes the labelled tree obtained from $\mathbf{T}_1$ when $\mathbf{S}$ is substituted by $\mathbf{T}_2$. Formally, we set $\mathbf{T} := (T, \geq, \nu)$ with $T := (T_1 - S) \cup T_2$, $\geq := \geq_1 \big|_{T_1 - S} \cup \geq_2 \cup \{(w_1, w_2) \mid w_1 > v,\, w_1 \in T_1 - S,\, w_2 \in T_2\}$, and $\nu := \nu_1\big|_{(T_1 - S)} \cup \nu_2$.

Composition: Let $l \in L$ be a label and $\mathbf{T}_1, \mathbf{T}_2$ be labelled trees. Then $l(\mathbf{T}_1, \mathbf{T}_2)$ denotes the labelled tree $\mathbf{T} := (T, \geq, \nu)$, where we have $T := T_1 \cup T_2 \cup \{v\}$ for a fresh node v, $\geq := \geq_1 \cup \geq_2 \cup (\{v\} \times (T_1 \cup T_2))$, and $\nu := \nu_1 \cup \nu_2 \cup \{(v, l)\}$. That is, $\mathbf{T}$ is the tree having a root labelled with l and which has $\mathbf{T}_1$ and $\mathbf{T}_2$ as (direct) subtrees.

Strictly speaking, in the above operations we have sometimes to consider trees with *disjoint* sets of nodes (for example, we have to assume in $\mathbf{T}_1[\,\mathbf{T}_2 \,/\, \mathbf{S}\,]$ that T_1 and T_2 are disjoint). As we consider trees only up to isomorphism, this can always easily be achieved and is usually not explicitely mentioned.

Using these operations, we can now define the tree-style syntax for $\mathcal{ALC}$.

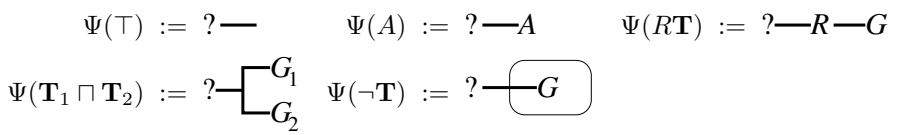
Definition 2.1 *Let a vocabulary $(\mathcal{A}, \mathcal{R})$ be given with $\top \in \mathcal{A}$. Let '$\sqcap$' and '$\neg$' be two further signs, denoting conjunction and negation. Let $(\Delta^{\mathcal{I}}, \mathcal{I})$ be a interpretation for the vocabulary $(\mathcal{A}, \mathcal{R})$. We inductively define the elements of $\mathcal{ALC}^{Tree}$ as labelled trees $\mathbf{T} := (T, \geq, \nu)$, as well as the interpretation $\mathcal{I}(\mathbf{T})$ of $\mathbf{T}$ in $(\Delta^{\mathcal{I}}, \mathcal{I})$.*

Atomic Trees: *For each $A \in \mathcal{A}$, the labelled tree A (i.e. the tree with one node labelled with A), as well as $\top$ are in $\mathcal{ALC}^{Tree}$. According to the interpretation of names in interpretations, we set $\mathcal{I}(A) = A^{\mathcal{I}}$ and $\mathcal{I}(\top) = \Delta^{\mathcal{I}}$.*
Negation: *Let $\mathbf{T} \in \mathcal{ALC}^{Tree}$. Then the tree $\mathbf{T}' := \neg\mathbf{T}$ is in $\mathcal{ALC}^{Tree}$. We set $\mathcal{I}(\mathbf{T}') = \Delta^{\mathcal{I}} - \mathcal{I}(\mathbf{T})$.*
Conjunction: *Let $\mathbf{T}_1, \mathbf{T}_2 \in \mathcal{ALC}^{Tree}$. Then the tree $\mathbf{T} := \sqcap(\mathbf{T}_1, \mathbf{T}_2)$ is in $\mathcal{ALC}^{Tree}$. We set $\mathcal{I}(\mathbf{T}) = \mathcal{I}(\mathbf{T}_1) \cap \mathcal{I}(\mathbf{T}_2)$.*
Exists Restriction: *Let $\mathbf{T} \in \mathcal{ALC}^{Tree}$, let R be a role name. Then $\mathbf{T}' := R\mathbf{T}$ is in $\mathcal{ALC}^{Tree}$. We set $\mathcal{I}(\mathbf{T}') = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : xR^{\mathcal{I}}y \wedge y \in \mathcal{I}(\mathbf{T})\}$.*

*The labelled trees of $\mathcal{ALC}^{Tree}$ are called $\mathcal{ALC}$-*TREES*. Let $\mathbf{T} := (T, \geq, \nu) \in \mathcal{ALC}^{Tree}$. An element $v \in T$ respectively the corresponding subtree $\mathbf{T}_v$ is said to be* EVENLY ENCLOSED*, iff $|\{w \in T \mid w > v \text{ and } \nu(w) = \neg\}|$ is even. The notation of* ODDLY ENCLOSED *is defined accordingly.*

Of course, $\mathcal{ALC}$-trees correspond to the formulas of $\mathcal{ALC}$, as they are defined in the usual linear fashion. For this reason, we will sometimes mix the notation of $\mathcal{ALC}$-formulas and $\mathcal{ALC}$-trees. Particularly, we sometimes write $\mathbf{T}_1 \sqcap \mathbf{T}_2$ instead of $\sqcap(\mathbf{T}_1, \mathbf{T}_2)$. Moreover, the conjunction of trees can be extended to an arbitrary number of conjuncts, i.e.: If $\mathbf{T}_1, \ldots, \mathbf{T}_n$ are $\mathcal{ALC}$-trees, we are free to write $\mathbf{T}_1 \sqcap \ldots \sqcap \mathbf{T}_n$. We agree that for $n = 0$, we set $\mathbf{T}_1 \sqcap \ldots \sqcap \mathbf{T}_n := \top$.

Next a diagrammatic representation of $\mathcal{ALC}$-trees in the style of Peirce's EGs is provided. EGs as such correspond to *closed* FOL-formulas: They are evaluated to true or false. Nonetheless, they can be easily extended to RELATION GRAPHS [14, 6] which are evaluated to relations instead. This is done by adding a syntactical device to EGs which corresponds to free variables. The diagrammatic rendering of free variables can be done via numbered question markers, which are attached to the lines of identity of EGs. As $\mathcal{ALC}$-concepts correspond to FOL-formulas with exactly one free variable, we we assign to each $\mathcal{ALC}$-tree $\mathbf{T}$ a corresponding relation graph $\Psi(\mathbf{T})$ with exactly one (now unnumbered) query marker. Let A be an atomic concept, R be a role name, let $\mathbf{T}$, $\mathbf{T}_1$, $\mathbf{T}_2$ be $\mathcal{ALC}$-trees where we already have defined $\Psi(\mathbf{T}) = ?\text{—}G$, $\Psi(\mathbf{T}_1) = ?\text{—}G_1$, and $\Psi(\mathbf{T}_2) = ?\text{—}G_2$, respectively. Now Ψ is defined inductively as follows:



Considering our HAPPYMAN-example given in the introduction, the corresponding $\mathcal{ALC}$-tree, and a corresponding Peirce graph, is provided in Fig. 3. Due to our choice of constructors, we replaced $\forall f$ by $\neg\exists\neg f$ and $f \vee g$ by $\neg(\neg f \wedge \neg g)$ (for DL-concepts f, g). The rules of the forthcoming calculus can be best understood to be carried out on the Peirce graphs. The ongoing formal proofs with $\mathcal{ALC}$-trees will be depicted this way.

Finally we define semantic entailment between $\mathcal{ALC}$-trees.

Definition 2.2 *Let $\{\mathbf{T}_i \mid i \in I\}$ be a set of $\mathcal{ALC}$-Trees and let $\mathbf{T}$ be an $\mathcal{ALC}$-Tree. We set*

$$\{\mathbf{T}_i \mid i \in I\} \models \mathbf{T} \ :\Longleftrightarrow \bigcap_{i \in I} \mathcal{I}(\mathbf{T}_i) \subseteq \mathcal{I}(\mathbf{T}) \text{ for each interpretation } (\Delta^{\mathcal{I}}, \mathcal{I})$$

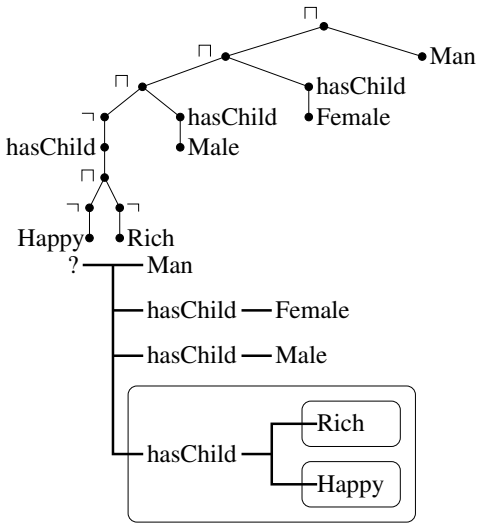


Figure 3: The $\mathcal{ALC}$-tree, and a corresponding Peirce graph for HAPPYMAN

For $I = \emptyset$, we set $\bigcap_{i \in I} \mathcal{I}(\mathbf{T}_i) := \Delta^{\mathcal{I}}$ for the respective model, and write $\models \mathbf{T}$. For $|I| = 1$, we write $\mathbf{T}' \models \mathbf{T}$.

3 The Calculus for $\mathcal{ALC}^{Tree}$

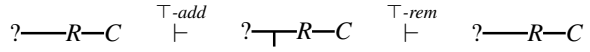
Peirce provided a set of five rules for the system of EGs, termed *erasure*, *insertion*, *iteration*, *deiteration*, *double cut*. These rules are formally elaborated and their soundness and completeness is proven in [6]. Moreover, it is discussed in [6] how these rules have to be extended for the system of relation graphs.

The class of relation graphs corresponding to $\mathcal{ALC}$ is a fragment of the full system of relation graphs. Of course, the rules for relation graphs are still sound rules for the $\mathcal{ALC}$-fragment. But it is not clear whether these rules are still complete. For two graphs G_1, G_2 of the $\mathcal{ALC}$-fragment with $G_1 \models G_2$, we have a proof for $G_1 \vdash G_2$ within the full system of relation graphs, i.e., a sequence of graphs starting with G_1, ending with G_2, where each graph in the sequence is derived from its predecessor by one of Peirce's five rules. However it may happen that we do not have a proof which consists only of graphs of the $\mathcal{ALC}$-fragment. In fact, in the calculus we provide, we need more than Peirce's five rules. Besides some trivial rules, like rules which capture the associativity of conjunction, we need special rules for handling of roles. The rules *iteration of roles into even* and *deiteration of roles from odd* (see below) are the most important example for this.

Next, the Peirce style rules for $\mathcal{ALC}^{Tree}$ are provided. These rules transform a given $\mathcal{ALC}$-tree into a new $\mathcal{ALC}$-tree. In order to make the calculus more understandable, we provide within the rule definitions some examples and diagrams that illustrate them. For each rule name, we provide an abbreviation which will be used in the proofs.

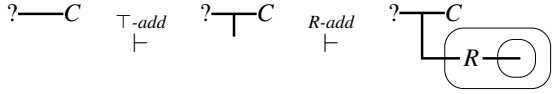
Definition 3.1 *The calculus for $\mathcal{ALC}$-Trees over a given vocabulary $(\mathcal{A}, \mathcal{R})$ consists of the following rules:*

Addition and Removal of $\top$ ***(*** $\top$***-add. and*** $\top$***-rem.):*** *Let* $\mathbf{T} := (T, \geq, \nu)$ *be an* $\mathcal{ALC}$*-tree, let* $\mathbf{S} \subseteq \mathbf{T}$ *be a subtree. For* $\mathbf{T}' := \mathbf{T}[\sqcap(\mathbf{S}, \top)) \,/\, \mathbf{S}\,]$ *we set* $\mathbf{T} \dashv\vdash \mathbf{T}'$ *(*$\mathbf{T} \dashv\vdash \mathbf{T}'$ *abbreviates* $\mathbf{T} \vdash \mathbf{T}'$ *and* $\mathbf{T}' \vdash \mathbf{T}$*). We say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* ADDING A $\top$-NODE*, and* $\mathbf{T}$ *is derived from* $\mathbf{T}'$ *by* REMOVING A $\top$-NODE*. For the Peirce graphs, this rule corresponds to adding a branch to a heavily drawn line, or removing it. A simple example is given below.*



These rules are "technical 'helper' rules that will be often combine with other rules that add or remove subtrees. Examples will be given below.

Addition and Removal of Roles (R-add. and R-rem.): *Let* $\mathbf{T}$ *be an* $\mathcal{ALC}$*-tree having* $\top$ *as a subtree. Let* R *be a role name. Then for* $\mathbf{T}[\,\neg R\neg\top \,/\, \top\,]$ *we set* $\mathbf{T} \dashv\vdash \mathbf{T}'$*. We say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* ADDING THE ROLE R*, and* $\mathbf{T}$ *is derived from* $\mathbf{T}'$ *by* REMOVING THE ROLE R*. A simple example for this rule with Peirce graphs is given below. Due to the symmetry of the rules, the inverse direction of this proof is a proof as well.*



Associativity of Conjunction (conj.): *Let* $\mathbf{T}$ *be an* $\mathcal{ALC}$*-tree with a subtree* $\mathbf{S}_1 \sqcap (\mathbf{S}_2 \sqcap \mathbf{S}_3)$*. For* $\mathbf{T}' := \mathbf{T}[\,(\mathbf{S}_1 \sqcap \mathbf{S}_2) \sqcap \mathbf{S}_3 \,/\, \mathbf{S}_1 \sqcap (\mathbf{S}_2 \sqcap \mathbf{S}_3)\,]$ *we set* $\mathbf{T} \dashv\vdash \mathbf{T}'$*. We say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *resp.* $\mathbf{T}$ *is derived from* $\mathbf{T}'$ *by* USING THE ASSOCIATIVITY OF CONJUNCTION.

Addition and Removal of a Double Negation (dn): *Let* $\mathbf{T} := (T, \geq, \nu)$ *be an* $\mathcal{ALC}$*-tree, let* $\mathbf{S} \subseteq \mathbf{T}$ *be a subtree. Then for* $\mathbf{T}' := \mathbf{T}[\,\neg\neg\mathbf{S} \,/\, \mathbf{S}\,]$ *we set* $\mathbf{T} \dashv\vdash \mathbf{T}'$*. We say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* ADDING A DOUBLE NEGATION *and* $\mathbf{T}$ *is derived from* $\mathbf{T}'$ *by* REMOVING A DOUBLE NEGATION.

Erasure from even, Insertion into odd (era. and ins.): *Let* $\mathbf{T} :=$ *be an* $\mathcal{ALC}$*-tree with a positively enclosed subtree* $\mathbf{S} \subseteq \mathbf{T}$*. Then for* $\mathbf{T}' := \mathbf{T}[\,\top \,/\, \mathbf{S}\,]$ *we set* $\mathbf{T} \vdash \mathbf{T}'$*. We say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* ERASING $\mathbf{S}$ FROM EVEN*. Vice versa, let* $\mathbf{T} =$ *be an* $\mathcal{ALC}$*-tree with an negatively enclosed subtree* $\top \subseteq \mathbf{T}$*. Let* $\mathbf{S} \in \mathcal{ALC}^{Tree}$*. Then for* $\mathbf{T}' := \mathbf{T}[\,\mathbf{S} \,/\, \top\,]$ *we set* $\mathbf{T} \vdash \mathbf{T}'$*. We say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* INSERTING $\mathbf{S}$ INTO ODD.

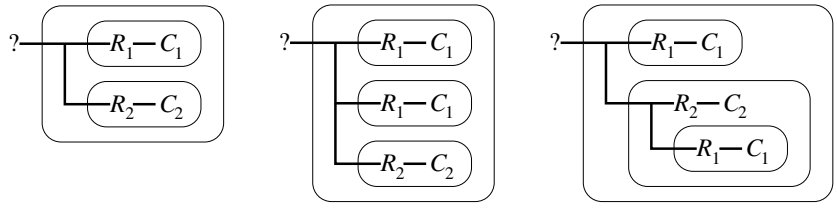
This is another set of rules which often go together with the addition and removal of $\top$*. Examples will be given later.*

Iteration and Deiteration (it. and deit.): *Let* $\mathbf{T} := (T, \geq, \nu)$ *be an* $\mathcal{ALC}$*-tree with with a subtree* $\mathbf{S} := (S, \geq_S, \nu_S) \subseteq \mathbf{T}$*. Let* s *be the greatest element of* $\mathbf{S}$*, let* t *be the parent node of* s *in* $\mathbf{T}$*. Let* $\nu(t) = \sqcap$ *(i.e.* t *is labelled with '*$\sqcap$*'), let* $v \in T$ *be a node with* $v < t$*,* $v \notin S$*,* $\nu(v) = \top$*, such that for each node* w *with* $t > w > v$ *we have* $\nu(w) = \neg$ *or* $\nu(w) = \sqcap$*. Then for* $\mathbf{T}' := \mathbf{T}[\,\mathbf{S} \,/\, \top\,]$ *we set* $\mathbf{T} \dashv\vdash \mathbf{T}'$*.*[1] *We say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* ITERATING $\mathbf{S}$ *and* $\mathbf{T}$ *is derived from* $\mathbf{T}'$ *by* DEITERATING $\mathbf{S}$*.*

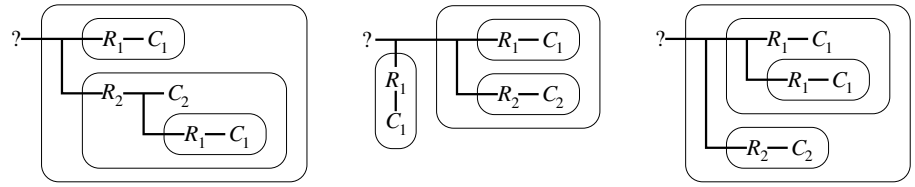
Iteration and Deiteration often combine with the addition and removal of $\top$*, and they are probably the most complex rules. To exemplify them, we consider the following six Peirce graphs. The second and the third graph can be derived from the first graph*

[1] More precisely and according to our convention, we set $\mathbf{T}' := \mathbf{T}[\,\mathbf{S}' \,/\, \top\,]$, where $\mathbf{S}'$ is an isomorphic copy of $\mathbf{S}$, having only fresh nodes.

by iterating the subgraph ?—(R1—C1) *(preceeded by the* $\top$*-addition rule).*

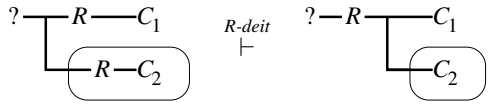


The next three graphs are not *results from the iteration rule. In the fourth graph, the condition that* $\nu(w) = \neg$ *or* $\nu(w) = \sqcap$ *holds for each node* w *with* $t > w > v$ *is violated. The fifth graph violates the condition* $v < t$*, and the sixth graph violates the condition* $v \notin S$*.*



Iteration of Roles into even, Deiteration of Roles from odd (R-it. and R-deit.): *Let* $\mathbf{T}$ *be an* $\mathcal{ALC}$*-tree. Let* $\mathbf{S}_a, \mathbf{S}_b, \mathbf{S}_1, \mathbf{S}_2$ *be* $\mathcal{ALC}$*-trees with* $\mathbf{S}_a := \sqcap(R\mathbf{S}_1, \neg R\mathbf{S}_2)$ *and* $\mathbf{S}_b := R \sqcap (\mathbf{S}_1, \neg\mathbf{S}_2)$*. Then, if* $\mathbf{S}_a \subseteq \mathbf{T}$ *is a positively enclosed subtree, for* $\mathbf{T}' := \mathbf{T}[\,\mathbf{S}_b \,/\, \mathbf{S}_a\,]$ *we set* $\mathbf{T} \vdash \mathbf{T}'$*, and we say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* DEITERATING THE ROLE R FROM ODD*. Vice versa, if* $\mathbf{S}_b \subseteq \mathbf{T}$ *is a negatively enclosed subtree, for* $\mathbf{T}' := \mathbf{T}[\,\mathbf{S}_a \,/\, \mathbf{S}_b\,]$ *we set* $\mathbf{T} \vdash \mathbf{T}'$*, and we say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* ITERATING THE ROLE R INTO EVEN*.*

Below, a simple example for the rule with Peirce's graphs is provided.



Based on these rules, we can now define formal proofs.

Definition 3.2 *Let* $\mathbf{T}_a, \mathbf{T}_b$ *be two* $\mathcal{ALC}$*-Trees. A* PROOF FOR $\mathbf{T}_a \vdash \mathbf{T}_b$ *is a finite sequence* $(\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n)$ *with* $\mathbf{T}_a = \mathbf{T}_1$*,* $\mathbf{T}_b = \mathbf{T}_n$*, where each* $\mathbf{T}_{i+1}$ *is obtained from* $\mathbf{T}_i$ *by applying one of the rules of the calculus.*

Now let $\{\mathbf{T}_i \mid i \in I\}$ *be a set of* $\mathcal{ALC}$*-Trees and let* $\mathbf{T}$ *be an* $\mathcal{ALC}$*-Tree. We set*

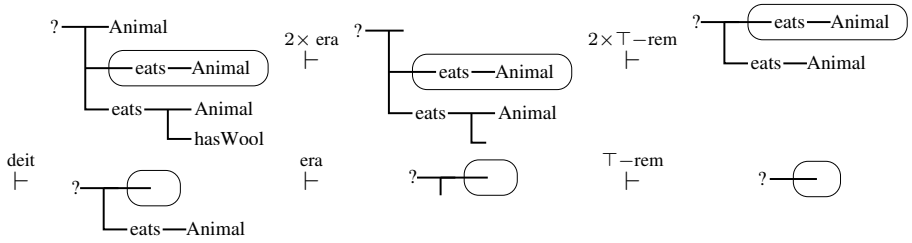
$$\{\mathbf{T}_i \mid i \in I\} \vdash \mathbf{T} \quad :\Longleftrightarrow \quad \text{there are } \mathcal{ALC}\text{-Trees } \mathbf{T}_1, \ldots, \mathbf{T}_n \in \{\mathbf{T}_i \mid i \in I\} \text{ with } \mathbf{T}_1 \sqcap \ldots \sqcap \mathbf{T}_n \vdash \mathbf{T}$$

Before the soundness and completeness of the calculus is proven, we first present an example of a proof, using the Peirce-style diagrams, and then derive some useful metarules. The example and the metarules will give some insights in how the calculus works.

A popular toy example for $\mathcal{ALC}$.reasoning is the mad cow ontology. Consider the following $\mathcal{ALC}$-definitions:

$$\begin{array}{rcl@{\qquad}rcl} Cow & \equiv & Animal \sqcap Vegetarian & Sheep & \equiv & Animal \sqcap hasWool \\ Vegetarian & \equiv & \forall eats.\neg Animal & MadCow & \equiv & Cow \sqcap \exists eats.Sheep \end{array}$$

The question to answer is whether this ontology is consistent. This question can be reduced to rewriting the ontology to a single concept $MadCow \equiv Animal \sqcap \forall eats.\neg Animal \sqcap \exists eats.(Animal \sqcap hasWool)$ and to investigate whether this concept is satisfiable, i.e., whether there exists as least one interpretation where this concept is interpreted by a non-empty set. We will show that this is not the case by proving with our calculus that the concept entails the absurd concept. The proof is given below.



We started with the Peirce graph for the given concept and derived the absurd concept, thus the ontology is not satisfiable.

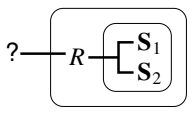
Next, we provide the above mentioned metarules, some of them will be used in the completeness proof.

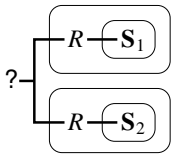
Each rule of the calculus is basically the substitution of a subtree of a given $\mathcal{ALC}$-tree by another subtree. Each rule can be applied to arbitrarily deeply nested subtrees. Moreover, if we have a rule which can be applied to positively enclosed subtrees, then we always have a rule in the converse direction which can be applied to negatively enclosed subtrees (and visa versa). Due to these structural properties of rules, we immediately obtain the following helpful lemma (it is adopted from [17]).

Lemma 3.1 *Let $\mathbf{S}_1, \mathbf{S}_2$ be two $\mathcal{ALC}$-trees with $\mathbf{S}_1 \vdash \mathbf{S}_2$. Let $\mathbf{T}$ be an $\mathcal{ALC}$-tree. Then if $\mathbf{S}_1 \subseteq \mathbf{T}$ is a positively enclosed subtree of $\mathbf{T}$, we have $\mathbf{T} \vdash \mathbf{T}[\,\mathbf{S}_2 \,/\, \mathbf{S}_1\,]$. Visa versa, if $\mathbf{S}_2 \subseteq \mathbf{T}$ is a negatively enclosed subtree of $\mathbf{T}$, we have $\mathbf{T} \vdash \mathbf{T}[\,\mathbf{S}_1 \,/\, \mathbf{S}_2\,]$.*

The next lemma corresponds to the equivalence of the $\mathcal{ALC}$-concepts $\forall R.(C \sqcap D)$ and $\forall R.C \sqcap \forall R.D$.

Lemma 3.2 (Splitting Roles) *Let $\mathbf{S}_1, \mathbf{S}_2$ be $\mathcal{ALC}$-trees, let $R \in \mathcal{R}$ be a role name. Then:*

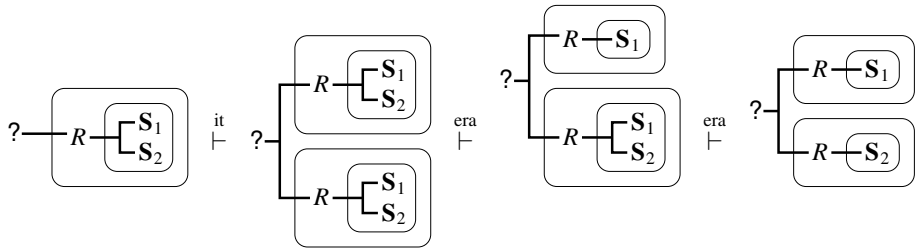
The trees 

and 

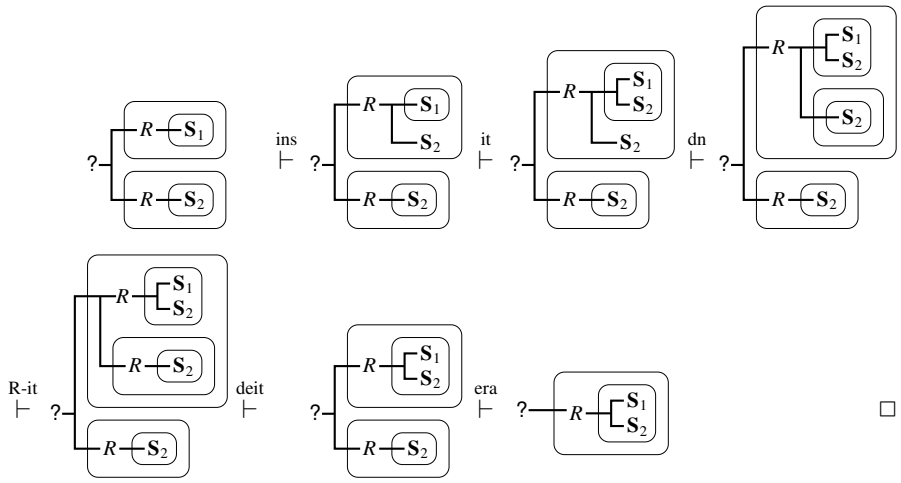
are equivalent.

Proof: We show the directions '$\Rightarrow$' and '$\Leftarrow$' separately. As we have already seen, the deiteration-rule and the erasure rule are usually followed by the $\top$-removal rule, and visa versa, the iteration rule and the insertion rule are usually preceeded by the $\top$-addition rule. In the proof, these two steps are combined without explicitely mentioning

the $\top$-removal/addition rule. Now '$\Rightarrow$' is proven as follows:



The other direction is as follows:

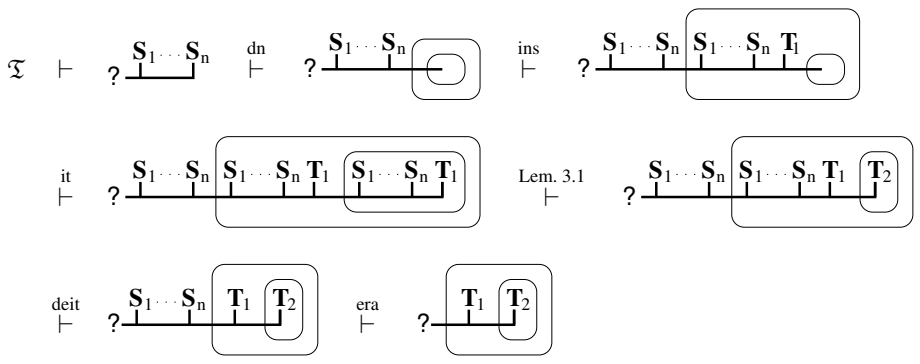


For $\mathcal{ALC}$, the full deduction theorem holds.

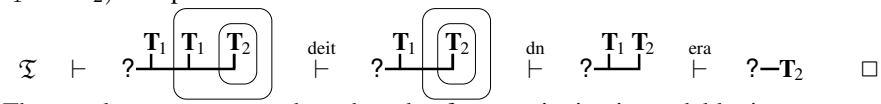
Theorem 3.1 (Deduction Theorem) *Let $\mathfrak{T}$ be a set of $\mathcal{ALC}$-trees, let $\mathbf{T}_1, \mathbf{T}_2$ be two $\mathcal{ALC}$-trees. Then we have:* $\mathfrak{T} \cup \{\mathbf{T}_1\} \vdash \mathbf{T}_2 \iff \mathfrak{T} \vdash \neg(\mathbf{T}_1 \sqcap \neg\mathbf{T}_2)$

Proof: Again applications of the $\top$-removal/addition rule are not explicitly mentioned.

'$\Rightarrow$': Let $\mathbf{S}_1, \ldots, \mathbf{S}_n \in \mathfrak{T}$ with $\mathbf{S}_1 \sqcap \ldots \sqcap \mathbf{S}_n \sqcap \mathbf{T}_1 \vdash \mathbf{T}_2$. We have:



'$\Leftarrow$': From $\mathfrak{T} \vdash \neg(\mathbf{T}_1 \sqcap \neg\mathbf{T}_2)$ and $\mathfrak{T} \cup \{\mathbf{T}_1\} \vdash \mathbf{T}_1$ we get $\mathfrak{T} \cup \{\mathbf{T}_1\} \vdash \mathbf{T}_1 \sqcap \neg(\mathbf{T}_1 \sqcap \neg\mathbf{T}_2)$. We proceed as follows:



The next lemma corresponds to the rule of neccessitation in modal logics.

Lemma 3.3 *If* $\mathbf{T}$ *is an* $\mathcal{ALC}$*-tree with* $\vdash \mathbf{T}$*, then we have* $\vdash \neg R \neg \mathbf{T}$ *as well.*

Proof: All rules of the calculus modify subtrees $\mathbf{S}$ of a given $\mathcal{ALC}$-tree $\mathbf{T}$, and their application depends only on whether $\mathbf{S}$ is positively or negatively enclosed. So if $(\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n)$ with $\mathbf{T}_1 = \top$ and $\mathbf{T}_n = \mathbf{T}$ is a proof for $\mathbf{T}$, then $(\top, \neg R \neg \mathbf{T}_1, \neg R \neg \mathbf{T}_2, \ldots, \neg R \neg \mathbf{T}_n)$ is a proof for $\vdash \neg R \neg \mathbf{T}$. The additional first step is an application of the rule 'addition of roles'. □

Please note that for this lemma, it is vital that $\mathbf{T}$ is derived from the empty set (the empty sheet of assertion in Peirce's terminology). The proof of the lemma does not work if $\mathbf{T}$ is derived from some set $\mathfrak{T}$, and it can easily be seen that we generally cannot conclude $\mathfrak{T} \vdash \neg R \neg \mathbf{T}$ from $\mathfrak{T} \vdash \mathbf{T}$.

In the following, the soundness and completeness of the calculus is proven. In contrast to rules of most common calculi, the rules presented here are 'deep' rules, as they modify deeply nested subtrees. For this reason, the following lemma is helpful for proving the soundness of the rules.

Lemma 3.4 *Let* $\mathbf{T}_1$*,* $\mathbf{T}_2$*,* $\mathbf{S}_1$*,* $\mathbf{S}_2$ *be* $\mathcal{ALC}$*-trees with* $\mathbf{T}_2 = \mathbf{T}_1[\,\mathbf{S}_2 \,/\, \mathbf{S}_1\,]$.

1. *If* $\mathbf{S}_1 \models \mathbf{S}_2$ *and the substitution takes place in an even, then* $\mathbf{T}_1 \models \mathbf{T}_2$.
2. *If* $\mathbf{S}_2 \models \mathbf{S}_1$ *and the substitution takes place in an odd, then* $\mathbf{T}_1 \models \mathbf{T}_2$.
3. *If* $\mathbf{S}_1$ *and* $\mathbf{S}_2$ *are semantically equivalent, then so are* $\mathbf{T}_1$ *and* $\mathbf{T}_2$.

Proof: The proof of this lemma is a straight-forward induction on $\mathcal{ALC}$-trees.

We are now prepared to prove the soundness of the calculus.

Theorem 3.2 *If* $(\Delta^{\mathcal{I}}, \mathcal{I})$ *is a model and if* $\mathbf{T}'$ *is obtained from* $\mathbf{T}$ *by one of the rules, we have* $\mathcal{I}(\mathbf{T}) \subseteq \mathcal{I}(\mathbf{T}')$.

Proof: The $\mathcal{ALC}$-trees $\mathbf{S}$ and $\sqcap(\mathbf{S}, \top)$ are obviously equivalent. So the soundness of the rule 'Addition or Removal of $\top$' follows immediately from Lemma 3.4(iii). The rules 'Addition or Removal of Roles', 'Associativity of Conjunction' and 'Addition or Removal of a Double Negation' are handled similarly.

Next, as we have $\mathbf{T} \models \top$, Lemma 3.4(i) yields the soundness of the erasure of a a positively enclosed subtree, and Lemma 3.4(ii) yields the soundness of the insertion negatively enclosed subtree.

Next we consider the iteration and deiteration of roles. Let $\mathbf{S}_a, \mathbf{S}_b$ be defined as in the rule. We will show $\mathbf{S}_a \models \mathbf{S}_b$. Let $a \in \mathcal{I}(\mathbf{S}_a)$. Then it follows $a \in \mathcal{I}(R\mathbf{S}_1)$ and $a \in \mathcal{I}(\neg R\mathbf{S}_2)$. Therefore there exists $b \in \Delta^{\mathcal{I}}$ with aRb and $b \in \mathcal{I}(\mathbf{S}_1)$, but there exists no $c \in \Delta^{\mathcal{I}}$ with aRc and $c \in \mathcal{I}(\mathbf{S}_2)$. Particularly we have $b \notin \mathcal{I}(\mathbf{S}_2)$. We conclude $b \in \mathcal{I}(\neg \mathbf{S}_2)$, so we have $b \in \mathcal{I}(\sqcap(\mathbf{S}_1, \neg \mathbf{S}_2))$ as well. Due to aRb, we finally obtain $a \in \mathcal{I}(\mathbf{S}_b)$. As we now have $\mathbf{S}_a \models \mathbf{S}_b$, Lemma 3.4(i) yields the soundness of deiterating a role R from an odd, and Lemma 3.4(ii) yields the soundness of iterating a role into an even.

Finally, we have to prove the soundness of the iteration and deiteration rule. First note the iteration rule removes v from T and adds the fresh nodes of S' to T, i.e., we have $T - \{v\} \subseteq T'$. To ease the technical presentation, let us assume that the greatest element of $\mathbf{S}'$ is v (instead of a fresh node), so that we have $T \subseteq T'$.

For a node $w \in T$, let $\mathbf{T}_w$ be the corresponding subtree of $\mathbf{T}$, and let $\mathbf{T}'_w$ be the corresponding subtree of $\mathbf{T}'$ (particularly, due to our assumption, we have $\mathbf{T}_v = \top$ and

$\mathbf{T}'_v = \mathbf{S}'$). We will prove that $\mathbf{T}_t$ and $\mathbf{T}'_t$ are semantically equivalent. So let $a \in \Delta^{\mathcal{I}}$. We have to show that,

$$a \in \mathcal{I}(\mathbf{T}_w) \quad \Longleftrightarrow \quad a \in \mathcal{I}(\mathbf{T}'_w) \tag{1}$$

holds for $w = t$. We have $\mathbf{T}' = \mathbf{T}[\,\mathbf{T}'_t \,/\, \mathbf{T}_t\,]$, so once Eqn. (1) is proven for $w = t$, we can now apply Lemma 3.4(iii) and obtain that $\mathbf{T}$ and $\mathbf{T}'$ are semantically equivalent, which yields the soundness of the iteration and deiteration rule. So it remains to show Eqn. (1).

In either $\mathbf{T}$ and $\mathbf{T}'$, the node t has two branches, one of which is $\mathbf{S}$. If we have $a \notin \mathcal{I}(\mathbf{S})$, we have $a \notin \mathcal{I}(\mathbf{T}_t)$ and $a \notin \mathcal{I}(\mathbf{T}'_t)$ as well, so Eqn. (1) holds. Now let us assume the alternate case, that we have $a \in \mathcal{I}(\mathbf{S})$. We will prove that Eqn. (1) holds for each w with $t \geq w \geq v$ by induction.

We have $\mathbf{T}_v = \top$, $\mathbf{T}'_v = \mathbf{S}$, $a \in \mathcal{I}(\top)$ and $a \in \mathcal{I}(\mathbf{S}')$, so Eqn. (1) holds for $w = v$. This proves the induction start.

For the induction step, let w be such that the induction hypothesis is proven for the child u of w with $u \geq v$. There are two cases to consider: $\nu(w) = \neg$ or $\nu(w) = \sqcap$.

For $\nu(w) = \neg$, we have $\mathbf{T}_w = \neg\mathbf{T}_u$ and $\mathbf{T}'_w = \neg\mathbf{T}'_u$. As we have $a \in \mathcal{I}(\mathbf{T}_u) \Leftrightarrow a \in \mathcal{I}(\mathbf{T}'_u)$ due to the induction hypothesis, we obtain that Eqn. (1) holds for w.

For $\nu(w) = \sqcap$, the node w has two children, one of them being u. Let u' be the child of w which is different from u. Then we have $\mathbf{T}_w = \sqcap(\mathbf{T}_u, \mathbf{T}_{u'})$ and $\mathbf{T}'_w = \sqcap(\mathbf{T}'_u, \mathbf{T}'_{u'})$. Moreover, we have $\mathbf{T}_{u'} = \mathbf{T}'_{u'}$, and $a \in \mathcal{I}(\mathbf{T}_u) \Leftrightarrow a \in \mathcal{I}(\mathbf{T}'_u)$ holds due to the induction hypothesis. From this we conclude that Eqn. (1) holds.

This finishes the induction, so we conclude Eqn. (1) for $w = t$, which in turn finishes the proof for the soundness of the iteration and deiteration rule. □

We are now prepared to prove the completeness of the calculus.

Theorem 3.3 *Let $\mathfrak{T} := \{\mathbf{T}_i \mid i \in I\}$ be a set of $\mathcal{ALC}$-Trees and let $\mathbf{T}$ be an $\mathcal{ALC}$-Tree. Then we have:*

$$\{\mathbf{T}_i \mid i \in I\} \models \mathbf{T} \quad \Longrightarrow \quad \{\mathbf{T}_i \mid i \in I\} \vdash \mathbf{T}$$

Proof: We assume that there is no derivation of $\mathbf{T}$ from $\mathfrak{T}$, and we show that $\mathfrak{T} \not\models \mathbf{T}$.

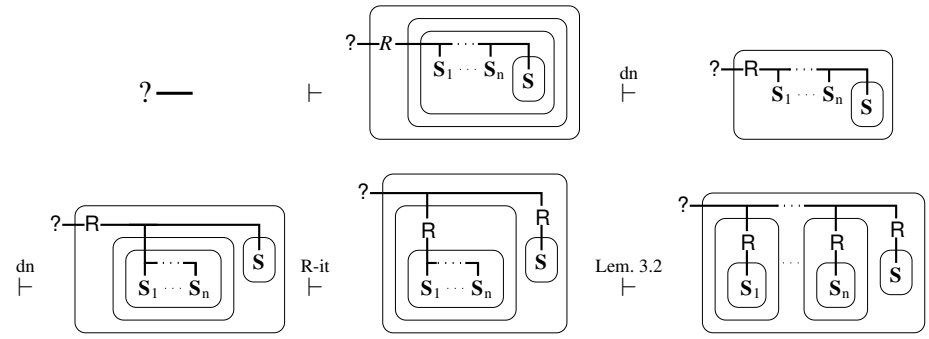
We call a set $\mathfrak{S}$ of $\mathcal{ALC}$-Trees INCONSISTENT, if we have $\mathfrak{S} \vdash \neg\top$. Due to the deduction theorem, $\mathfrak{S}$ is inconsistent if and only if there are $\mathbf{S}_1, \ldots, \mathbf{S}_n \in \mathfrak{S}$ with $\vdash \neg(\mathbf{S}_1 \sqcap \ldots \sqcap \mathbf{S}_n)$. We assume that there is no proof of $\mathbf{T}$ from $\mathfrak{T}$. Then $\mathfrak{T} \cup \{\neg\mathbf{T}\}$ is consistent.

For a set $\mathfrak{S}$ of $\mathcal{ALC}$-Trees and a role name R, let $\mathfrak{S}^R := \{\mathbf{S} \mid \neg R\neg\mathbf{S} \in \mathfrak{S}\}$. We first prove the following property:

$$\text{If } \mathfrak{S} \text{ is consistent, where } R\neg\mathbf{S} \in \mathfrak{S}, \text{ then } \mathfrak{S}^R \cup \{\neg\mathbf{S}\} \text{ is also consistent.} \tag{2}$$

It is easier to show the contraposition of Eqn. (2), so we assume that $\mathfrak{S}^R \cup \{\neg\mathbf{S}\}$ is not consistent. Then there exists finitely many elements $\mathbf{S}_1, \ldots, \mathbf{S}_n$ of $\mathfrak{S}^R$ such that there is a proof of $\neg(\mathbf{S}_1 \sqcap \ldots \mathbf{S}_n \sqcap \neg\mathbf{S})$ (from the empty set). Now Lemma 3.3 yields that

we have a proof of $\neg R\neg\neg(\mathbf{S}_1 \sqcap \ldots \mathbf{S}_n \sqcap \neg\mathbf{S})$ as well. We proceed as follows:



So $\mathfrak{S}$ is also $\mathfrak{T}$-inconsistent, thus Eqn. (2) holds.

Now, using a standard argument based on the axiom of choice, every consistent set can be extended to a maximal consistent set, i.e., a consistent set which cannot be properly extended to another consistent set.

Next, for a maximal consistent set $\mathfrak{S}_\mathrm{m}$, we have

$$\mathbf{S} \in \mathfrak{S}_\mathrm{m} \iff \neg\mathbf{S} \notin \mathfrak{S}_\mathrm{m} \tag{3}$$

$$\mathbf{S} \sqcap \mathbf{S}' \in \mathfrak{S}_\mathrm{m} \iff \mathbf{S} \in \mathfrak{S}_\mathrm{m} \text{ and } \mathbf{S}' \in \mathfrak{S}_\mathrm{m} \tag{4}$$

for arbitrary $\mathcal{ALC}$-Trees $\mathbf{S}, \mathbf{S}'$. We only prove Eqn. (3), the proof of Eqn. (4) is done similarly.

Due to $\mathbf{S} \sqcap \neg\mathbf{S} \overset{\text{deit}}{\vdash} \mathbf{S} \sqcap \neg\top \overset{\text{era}}{\vdash} \top \sqcap \neg\top \overset{\top\text{-rem}}{\vdash} \neg\top$, and as we can infer any tree from $\neg\top$, we cannot have both $\mathbf{S} \in \mathfrak{S}_\mathrm{m}$ and $\neg\mathbf{S} \in \mathfrak{S}_\mathrm{m}$. On the other hand, let us suppose we have $\neg\mathbf{S} \notin \mathfrak{S}_\mathrm{m}$. Then we have $\mathfrak{S}_\mathrm{m} \nvdash \neg\mathbf{S}$. Assume $\mathfrak{S}_\mathrm{m} \cup \{\mathbf{S}\}$ is inconsistent. Then we have $\mathbf{S}_1, \ldots, \mathbf{S}_n \in \mathfrak{S}_\mathrm{m}$ with $\vdash \neg(\mathbf{S}_1 \sqcap \ldots \sqcap \mathbf{S}_n \sqcap \mathbf{S})$. Thm. 3.1 yields $\mathbf{S}_1 \sqcap \ldots \sqcap \mathbf{S}_n \vdash \neg\mathbf{S}$, i.e., we have $\mathfrak{S}_\mathrm{m} \vdash \neg\mathbf{S}$. This contradicts $\mathfrak{S}_\mathrm{m} \nvdash \neg\mathbf{S}$. So if $\neg\mathbf{S} \notin \mathfrak{S}_\mathrm{m}$, then $\mathfrak{S}_\mathrm{m} \cup \{\mathbf{S}\}$ is consistent, thus $\mathbf{S} \in \mathfrak{S}_\mathrm{m}$ due to the maximality of $\mathfrak{S}_\mathrm{m}$. Hence Equation (3) is proven.

Now let $(\Delta^\mathcal{I}, \mathcal{I})$ be defined as $\Delta^\mathcal{I} := \{\mathfrak{S}_\mathrm{m} \mid \mathfrak{S}_\mathrm{m} \text{ is maximal consistent}\}$, $\mathcal{I}(A) := \{\mathfrak{S}_\mathrm{m} \mid \mathbf{T}_A \in \mathfrak{S}_\mathrm{m}\}$ and $\mathcal{I}(R) := \{(\mathfrak{S}_\mathrm{m}, \mathfrak{T}_\mathrm{m}) \mid \mathfrak{S}_\mathrm{m}^R \subseteq \mathfrak{T}_\mathrm{m}\}$. We prove by induction over the construction of $\mathcal{ALC}$-trees that for $\mathbf{S} \in \mathcal{ALC}^{Tree}$ and $\mathfrak{S}_\mathrm{m} \in \Delta^\mathcal{I}$ we have

$$\mathfrak{S}_\mathrm{m} \in \mathcal{I}(\mathbf{S}) \iff \mathbf{S} \in \mathfrak{S}_\mathrm{m} \tag{5}$$

For a concept name A, Eqn. (5) holds by the definition of the model. For a tree $\neg\mathbf{S}$, we have $\mathfrak{S}_\mathrm{m} \in \mathcal{I}(\neg\mathbf{S}) \overset{\text{Def. } \mathcal{I}}{\iff} \mathfrak{S}_\mathrm{m} \notin \mathcal{I}(\mathbf{S}) \overset{\text{I.H.}}{\iff} \mathbf{S} \notin \mathfrak{S}_\mathrm{m} \overset{\text{Eqn. (3)}}{\iff} \neg\mathbf{S} \in \mathfrak{S}_\mathrm{m}$. For a tree $\mathbf{S} \sqcap \mathbf{S}'$, Eqn. (5) is similarly proven using Eqn. (4). It remains to consider role names.

Let $R \in \mathcal{R}$. We first prove Eqn. (5) for $\mathcal{ALC}$-trees $\mathbf{S}' := \neg R\neg\mathbf{S}$ (instead of $\mathbf{S}' := R\mathbf{S}$). Due to our induction, we can assume that Eqn. (5) is proven for all $\mathcal{ALC}$-trees which have less occurrences of R than $\mathbf{S}'$. Def. 2.1 yields

$$\mathfrak{S}_\mathrm{m} \in \mathcal{I}(\neg R\neg\mathbf{S}) \iff \text{for all } \mathfrak{T}_\mathrm{m} \text{ with } (\mathfrak{S}_\mathrm{m}, \mathfrak{T}_\mathrm{m}) \in \mathcal{I}(R) \text{ we have } \mathfrak{T}_\mathrm{m} \in \mathcal{I}(\mathbf{S}) \tag{6}$$

Suppose first we have $\mathbf{S}' \in \mathfrak{S}_\mathrm{m}$. If $\mathfrak{T}_\mathrm{m} \in \Delta^\mathcal{I}$ is arbitrary with $(\mathfrak{S}_\mathrm{m}, \mathfrak{T}_\mathrm{m}) \in \mathcal{I}(R)$, then $\mathbf{S} \in \mathfrak{T}_\mathrm{m}$ by the definition of the model. The induction hypothesis yields $\mathfrak{T}_\mathrm{m} \in \mathcal{I}(\mathbf{S})$, so Eqn. (6) yields $\mathfrak{S}_\mathrm{m} \in \mathcal{I}(\mathbf{S}')$. Next suppose $\mathbf{S}' \notin \mathfrak{S}_\mathrm{m}$. Eqn. (3) yields $R\neg\mathbf{S} \in \mathfrak{S}_\mathrm{m}$. Let $\mathfrak{T}' := \mathfrak{S}_\mathrm{m}^R \cup \{\neg\mathbf{S}\}$. Then $\mathfrak{T}'$ is consistent due to Eqn. (2). Let $\mathfrak{T}_\mathrm{m} \supseteq \mathfrak{T}'$ be a maximal consistent set. Then $\mathfrak{T}_\mathrm{m} \in \Delta^\mathcal{I}$, and $(\mathfrak{S}_\mathrm{m}, \mathfrak{T}_\mathrm{m}) \in \mathcal{I}(R)$. Since $\neg\mathbf{S} \in \mathfrak{T}' \subseteq \mathfrak{T}_\mathrm{m}$,

we have $\mathbf{S} \notin \mathfrak{T}_\mathrm{m}$. The induction hypothesis yields $\mathfrak{T}_\mathrm{m} \notin \mathcal{I}(\mathbf{S})$, thus $\mathfrak{S}_\mathrm{m} \notin \mathcal{I}(\mathbf{S}')$ due to Eqn. (6). Hence Eqn. (5) is proven for $\mathbf{S}' = \neg R \neg \mathbf{S}$.

To finish the proof of Eqn. (5), let us finally observe that for $\mathcal{ALC}$-trees of the form $R\mathbf{S}$, we have $\mathfrak{S}_\mathrm{m} \in \mathcal{I}(R\mathbf{S}) \iff \mathfrak{S}_\mathrm{m} \notin \mathcal{I}(\neg R \neg \neg \mathbf{S}) \overset{\text{s.a.}}{\iff} \neg R \neg \neg \mathbf{S} \notin \mathfrak{S}_\mathrm{m} \iff R\mathbf{S} \in \mathfrak{S}_\mathrm{m}$. So Eqn. (5) is proven.

Now let $\mathfrak{S}_\mathrm{m} \in \Delta^\mathcal{I}$. We have: $\mathfrak{S}_\mathrm{m} \in \bigcap_{\mathbf{S} \in \mathfrak{T}} \mathcal{I}(\mathbf{S}) \iff \mathfrak{S}_\mathrm{m} \in \mathcal{I}(\mathbf{S})$ for all $\mathbf{S} \in \mathfrak{T} \overset{\text{Eqn. (5)}}{\iff} \mathbf{S} \in \mathfrak{S}_\mathrm{m}$ for all $\mathbf{S} \in \mathfrak{T} \iff \mathfrak{T} \subseteq \mathfrak{S}_\mathrm{m}$. This yields $\bigcap_{\mathbf{S} \in \mathfrak{T}} \mathcal{I}(\mathbf{S}) = \{\mathfrak{S}_\mathrm{m} \in \Delta^\mathcal{I} \mid \mathfrak{S}_\mathrm{m} \supseteq \mathfrak{T}\}$. As $\mathfrak{T} \cup \{\neg \mathbf{T}\}$ is consistent, there exist a maximal consistent set $\mathfrak{T}^0_\mathrm{m} \supseteq \mathfrak{T} \cup \{\neg \mathbf{T}\}$. On the one hand, we now have $\mathfrak{T}^0_\mathrm{m} \in \bigcap_{\mathbf{S} \in \mathfrak{T}} \mathcal{I}(\mathbf{S})$. On the other hand, we have $\mathbf{T} \notin \mathfrak{T}^0_\mathrm{m}$, thus $\mathfrak{T}^0_\mathrm{m} \notin \mathcal{I}(\mathbf{T})$ by Eqn. (5). So we obtain $\bigcap_{\mathbf{S} \in \mathfrak{T}} \mathcal{I}(\mathbf{S}) \not\subseteq \mathcal{I}(\mathbf{T})$, which means $\mathfrak{T} \not\models \mathbf{T}$. □

4 Conclusion and Further Research

This paper provides the first steps toward a diagrammatic representation of DLs, including diagrammatic inference mechanisms. To the best of our knowledge, this is the first attempt to providing *diagrammatic* reasoning facilities for DLs. The results presented in this paper show promise in investigating relation graphs further as diagrammatic versions of corresponding DLs.

The approach can to be extended to other variants of DL as well. For instance, a major task is to incorporate nominals, or number restrictions (either unqualified or qualified). Similarly, constructors on roles, like inverse roles or role intersection, have also to be investigated.

In the long term, our research advocates developing a major subset of DL as a mathematically precise diagrammatic reasoning system. While the intention is to render DL more user-friendly through a diagrammatic correspondence, such systems will need to be evaluated against the traditional textual form of DL in order to measure any readability improvement. Cognition experiments with such a evaluation are planned as future work.

5 Acknowledgements

We like to thank the anonymous referees for their valuable suggestions, and particularly to the ones who pointed out a gap in the former proof of the completeness.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] S. Brockmans, R. Volz, A. Eberhart, and P. Löffler. Visual modeling of owl dl ontologies using uml. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 198–213. Springer, Berlin – Heidelberg – New York, 2004.

[3] P. Coupey and C. Faron. Towards correspondence between conceptual graphs and description logics. In M.-L. Mugnier and M. Chein, editors, *ICCS*, volume 1453 of *LNAI*, pages 165–178. Springer, Berlin – Heidelberg – New York, 1998.

[4] F. Dau. Types and tokens for logic with diagrams: A mathematical approach. In K. E. Wolff, H. D. Pfeiffer, and H. S. Delugach, editors, *Conceptual Structures at Work: 12th International Conference on Conceptual Structures*, volume 3127 of *Lecture Notes in Computer Science*, pages 62–93. Springer, Berlin – Heidelberg – New York, 2004.

[5] F. Dau. Fixing shin's reading algorithm for peirce's existential graphs. In D. Barker-Plummer, R. Cox, and N. Swoboda, editors, *Diagrams*, volume 4045 of *LNAI*, pages 88–92. Springer, Berlin – Heidelberg – New York, 2006.

[6] F. Dau. Mathematical logic with diagrams, based on the existential graphs of peirce. Habilitation thesis. To be published. Available at: `http://www.dr-dau.net`, 2006.

[7] F. Dau. Some notes on proofs with alpha graphs. In P. Øhrstrøm, H. Schärfe, and P. Hitzler, editors, *ICCS*, volume 4068 of *Lecture Notes in Computer Science*, pages 172–188. Springer, Berlin – Heidelberg – New York, 2006.

[8] F. Dau and P. Eklund. Towards a diagrammatic reasoning system for description logics. Submitted to the Journal of Visual Languages and Computing, Elsevier. Available at `www.kvocentral.org`., 2006.

[9] B. R. Gaines. An interactive visual language for term subsumption languages. In *IJCAI*, pages 817–823, 1991.

[10] W. Hartshorne and Burks, editors. *Collected Papers of Charles Sanders Peirce*, Cambridge, Massachusetts, 1931–1935. Harvard University Press.

[11] J. Howse, F. Molina, S.-J. Shin, and J. Taylor. On diagram tokens and types. In M. Hegarty, B. Meyer, and N. H. Narayanan, editors, *Diagrams*, volume 2317 of *Lecture Notes in Computer Science*, pages 146–160. Springer, Berlin – Heidelberg – New York, 2002.

[12] R. Kremer. Visual languages for konwledge representation. In *Proc. of 11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98) Banff, Alberta, Canada*. Morgan Kaufmann, 1998.

[13] J. T. Nosek and I. Roth. A comparison of formal knowledge representation schemes as communication tools: Predicate logic vs semantic network. *International Journal of Man-Machine Studies*, 33(2):227–239, 1990.

[14] S. Pollandt. Relation graphs: A structure for representing relations in contextual logic of relations. In U. Priss, D. Corbett, and G. Angelova, editors, *Conceptual Structures: Integration and Interfaces*, volume 2393 of *LNAI*, pages 24–48, Borovets, Bulgaria, July, 15–19, 2002. Springer, Berlin – Heidelberg – New York.

[15] D. D. Roberts. *The Existential Graphs of Charles S. Peirce*. Mouton, The Hague, Paris, 1973.

[16] S.-J. Shin. *The Iconic Logic of Peirce's Graphs*. Bradford Book, Massachusetts, 2002.

[17] J. F. Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley, Reading, Mass., 1984.

[18] J. J. Zeman. *The Graphical Logic of C. S. Peirce*. PhD thesis, University of Chicago, 1964. Available at: `http://www.clas.ufl.edu/users/jzeman/`.