A Diagrammatic Reasoning System for $\mathcal{ALC}$

Frithjof Dau, Peter Eklund

Faculty of Informatics
University of Wollongong
Wollongong, NSW, 2522 Australia
dau,peklund@uow.edu.au

Abstract. Description logics (DLs) are a well-known family of knowledge representation (KR) languages. The notation of DLs has the style 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 provided. As the transformation rules modify the diagrammatic representation of $\mathcal{ALC}$ this produces a diagrammatic calculus. Some examples present in the paper illustrate the use and properties of this calculus.

1 Introduction

Description logics (DLs) [1] are a well-known and understood family of knowledge representation (KR) languages tailed to express knowledge about concepts and concept hierarchies. The basic building blocks of DLs are atomic concepts, atomic roles and individuals that can be composed by language constructs such as intersection, union, value or number restrictions (and more) to build more complex concepts and roles. For example, if Man, Female, Male, Rich, Happy are concepts and if hasChild is a role, we can define

$$\textsc{Man} \sqcap \exists \textsc{hasChild}.\textsc{Female} \sqcap \exists \textsc{hasChild}.\textsc{Male} \sqcap \forall \textsc{hasChild}.(\textsc{Rich} \sqcup \textsc{Happy})$$

as the concept of men who have both male and female children where all children are rich or happy. Let us call this concept HappyMan.

The formal notation of DLs has the style of a variable-free first order predicate logic (FOL) and DLs correspond to decidable fragments of FOL. Like FOL, DLs have a well-defined, formal syntax and Tarski-style semantics, and they exhibit sound and complete inference features. 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 difficult to comprehend.

A significant alternative to symbolic logic notation has been the development of a diagrammatic representation of DLs. It is well accepted that diagrams are, in many cases, easier to comprehend than symbolic notations [2–4], and in particular it has been argued that they are useful for knowledge representation systems [5, 6]. This has been acknowledged by DL researchers and is a common view among the broader knowledge representation community [7].

A first attempt to experiment with diagrammatic KR can be found in [5], where a graph-based representation for the textual DL CLASSIC is elaborated. In [8], a specific DL is mapped to the diagrammatic system of conceptual graphs [9]. In [10], a UML-based representation for a DL is provided. In these treatments the focus is on a graphical *representation* of DL, however *reasoning* is a distinguishing feature of DLs. Correspondences between graphical representation of the DL and the DL reasoning system are therefore important inclusions in any graphical representation. However, to date they remain largely unelaborated.

On the other hand there are diagrammatic reasoning systems that have the expressiveness of fragments of FOL or even full FOL. Examples are the various classes of spider- and constraint diagrams [11, 12], which are based on Euler circles and Venn-Peirce-diagrams, or the system of Sowa's conceptual graphs [9], which are based on Peirce's existential graphs. Contemporary elaborations of these systems include a well-defined syntax, extensional semantics and/or translations to formulas of FOL, and – most importantly for the goal of this paper – sound and complete calculi which can be best understood as manipulations of the diagrams.

This paper presents a diagrammatic representation of the DL $\mathcal{ALC}$ in the style of Peirce's existential graphs (EGs) [13, 4]. An diagrammatic calculus for $\mathcal{ALC}$, based on Peirce's transformation rules, is provided. The DL $\mathcal{ALC}$ is the smallest propositionally closed DLwhich renders it a good starting point for developing DLs as diagrammatic reasoning systems. More expressive DLs are targeted for further research. Note also that it is well known that $\mathcal{ALC}$ is a syntactical variant of the multi-modal logic **K**, thus the results of this paper can be reused in modal logics. The reasons for choosing EGs are given in [14].

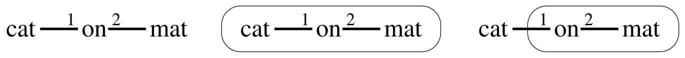
Reasoning with DLs is usually carried out by means of tableau algorithms. The calculus of this paper differs significantly from this approach in two respects. First, the rules of the calculus are *deep-inference* rules, as they modify deep nested sub-formulas, whereas tableau algorithms (similar to other common calculi) only modify formulas at their top-level. Secondly, the rules can be best understood to modify the diagrammatic Peirce-style representations of $\mathcal{ALC}$, i.e., the calculus is a genuine *diagrammatic* calculus.

The paper is structured as follows. First, an introduction existential graphs is provided in Section 2. In Section 3 the syntax and semantics of the DL $\mathcal{ALC}$ as we use it in this paper is introduced. In Section 4, the diagrammatic calculus for $\mathcal{ALC}$ is presented. Due to space limitations, the proof of its soundness and completeness is omitted: this result can be found in [14]. In Section 5, some examples and meta-rules for the calculus are provided. Finally, Section 6 provides a summary of this research and its significance.

2 Existential and Relation Graphs

Existential graphs (EGs) are a diagrammatic logic invented by C.S. Peirce (1839-1914) at the turn of the 20th century. We briefly introduce a fragment of EGs called Beta, corresponding to first order logic.

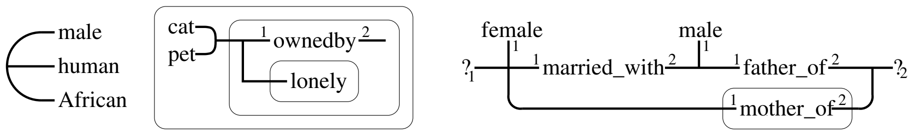
The graphs of Beta are composed of predicate names of arbitrary arity, heavily drawn lines, called LINES OF IDENTITY, are used to denote both the existence of objects and the identity between objects. Closed curves called CUTS are used to negate the enclosed subgraph. The area where the graph is written or drawn is what Peirce called the SHEET OF ASSERTION. Consider the following EGs.



The first graph contains two lines of identity, hence it denotes two (not necessarily different) objects. They are attached to the unary predicates 'cat' and 'mat', respectively, and both are attached to the binary predicate 'on'. 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'. In the second graph, the cut encloses our first example completely. Hence its meaning is 'it is not true that there is a cat on a mat'. In the third graph, the left line of identity begins on the sheet of assertion. Hence, the existence of the object is asserted and not denied. For this reason this graph is read 'there is a cat which is not on any mat'.

Lines of identity may be connected to networks that are called LIGATURES. In the two left-most graphs in the figures below ligatures are used. The meaning of these graphs is 'there exists a male, human african', 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'.

EGs are evaluated to true or false. Nonetheless, they can be easily extended to RELATION GRAPHS (RGs) [15, 16] which are evaluated to relations instead. This is done by adding a syntactical device that corresponds to free variables. The diagrammatic rendering of free variables can be done via numbered question markers. The rightmost graph below is a relation graph with two free variables. This graph describes the binary relation *is_stepmother_of*.



3 The Description Logic $\mathcal{ALC}$

The vocabulary $(\mathcal{A}, \mathcal{R})$ of a DL consists of a set $\mathcal{A}$ of (ATOMIC) CONCEPTS, denoting sets of individuals, and a set $\mathcal{R}$ (ATOMIC) ROLES, denoting binary relationships between individuals. Moreover, we 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 $C_1 \sqcap C_2$, $\neg C$, $\forall R.C$, $\exists R.C$, or $\leq nR.C$ (these constructors are called conjunction, negation, value restriction, existential restriction, and qualified number restriction).

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

An INTERPRETATION is a pair $(\Delta^{\mathcal{I}}, \mathcal{I})$, consisting of an nonempty DOMAIN $\Delta^{\mathcal{I}}$ 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 special 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 LABELED 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 is usual, each node v gives rise to a SUBTREE $\mathbf{T}_v$ ($\mathbf{T}_v = (T_v, \geq \big|_{T_v \times T_v}, \nu|_{T_v})$ 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 labeled trees are implicitly identified.

Next we introduce some operations to inductively construct labeled 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 labeled 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 labeled with l_1, l_2, $\ldots$, l_n, respectively. We extend this notation by allowing the last element to be a labeled tree: If $l_1\, l_2\, \ldots l_n \in L$ and if $\mathbf{T}'$ is a labeled tree, then $l_1\, l_2\, \ldots l_n\mathbf{T}'$ denotes the labeled 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 $\mathbf{T}'$.
Substitution: Let $\mathbf{T}_1, \mathbf{T}_2$ be labeled 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 labeled 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 labeled trees. Then $l(\mathbf{T}_1, \mathbf{T}_2)$ denotes the labeled 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 labeled with l and having $\mathbf{T}_1$ and $\mathbf{T}_2$ as subtrees.

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

Definition 1 ($\mathcal{ALC}$-Trees). *Let a vocabulary $(\mathcal{A}, \mathcal{R})$ be given with $\top \in \mathcal{A}$. Let '$\sqcap$' and '$\neg$' be two further signs. 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 labeled 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 labeled tree A (i.e. the tree with one node labeled with A), as well as $\top$ are in $\mathcal{ALC}^{Tree}$. 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* $\mathcal{I}(\mathbf{T}') = \{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : xRy \wedge y \in \mathcal{I}(\mathbf{T})\}$.

The labeled 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.*

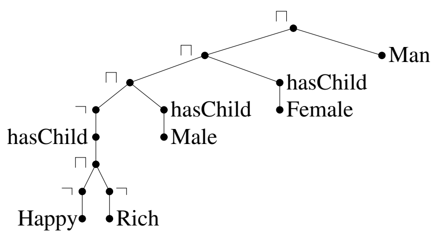
Of course, $\mathcal{ALC}$-trees correspond to the formulas of $\mathcal{ALC}$, as they are defined in the usual linear fashion. For this reason, we will sometimes mix the notation of $\mathcal{ALC}$-formulas and $\mathcal{ALC}$-trees. Particularly, we sometimes write $\mathbf{T}_1 \sqcap \mathbf{T}_2$ 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 $\mathbf{T}_1 \sqcap \ldots \sqcap \mathbf{T}_n := \top$.

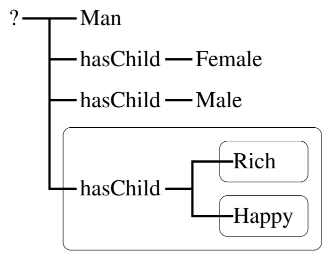
Next, a diagrammatic representation of $\mathcal{ALC}$-trees in the style of Peirce's RGs is provided. As $\mathcal{ALC}$-concepts correspond to FOL-formulas with exactly one free variable, we we assign to each $\mathcal{ALC}$-tree $\mathbf{T}$ a corresponding RG $\Psi(\mathbf{T})$ with exactly one (now 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:

$$\Psi(\top) := ?\text{—} \qquad \Psi(A) := ?\text{—}A \qquad \Psi(R\mathbf{T}) := ?\text{—}R\text{—}G$$

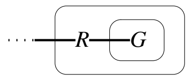
$$\Psi(\mathbf{T}_1 \sqcap \mathbf{T}_2) := ?\text{—}\left[\begin{array}{l}\text{—}G_1\\ \text{—}G_2\end{array}\right. \qquad \Psi(\neg\mathbf{T}) := ?\text{—}\boxed{\text{—}G}$$

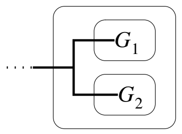
Considering our HAPPYMAN-example given in the introduction, the corresponding $\mathcal{ALC}$-tree, and a corresponding RG, is provided below. The rules of the forthcoming calculus can be best understood to be carried out on Peirce RGs.





It might be argued that having only constructors for conjunction, negation and existential restriction is a downside of this system. Although we argued in the introduction that diagrams are easier to comprehend than the symbolic notation for DLs, reading the diagrams also requires effort. Some reading heuristics help in the understanding RGs. For example, subgraphs of the form





are literally read $\ldots \neg\exists R.\neg C$ and $\ldots \neg(\neg C_1 \sqcap \neg C_2)$, respectively, but it is more convenient to read them as $\forall\exists R.C$ resp., $C_1 \sqcup C_2$. Shin [4] coined the term *multiple reading* for this approach. In her book, she elaborates this for EGs by proving a translation from EGs to FOL that assigns to each EG a *set* of (equivalent) FOL-formulas (unfortunately, her translations are slightly flawed: see [17] for a discussion and correction of her reading algorithm). Shin argues thoroughly in [4] that this multiple reading is a unique feature of the Peirce's graphs, a feature that distinguishes them from the usual symbolic notation in mathematical logic. This feature is not a drawback, but an advantage of Peirce's system.

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

Definition 2 (Semantics). *Let $\mathfrak{T} \subseteq \mathcal{ALC}^{Tree}$ and let $\mathbf{T} \in \mathcal{ALC}^{Tree}$. We set*

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

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 $\mathbf{T}' \models \mathbf{T}$, omitting the curly set brackets, or we write $\mathbf{T}' \sqsubseteq \mathbf{T}$ (adopting the common DL-notation), and we say that that $\mathbf{T}'$ SUBSUMES $\mathbf{T}$ resp. that $\mathbf{T}$ IS SUBSUMED BY $\mathbf{T}'$.

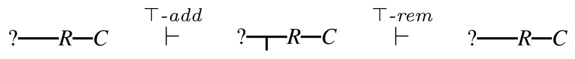
4 The Calculus for $\mathcal{ALC}^{Tree}$

Peirce provided a set of five rules for the system of existential graphs, termed *erasure*, *insertion*, *iteration*, *deiteration*, *double cut*. They form a sound and complete diagrammatic calculus for EGs. Moreover, they can be extended for the system of Relational Graphs (RGs).

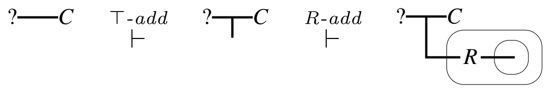
The class of RGs corresponding to $\mathcal{ALC}$ is a fragment of the full system of RGs. Naturally, the rules for RGs are still sound rules for the $\mathcal{ALC}$-fragment, but it is less clear whether these rules remain complete. For two graphs G_1, G_2 of the $\mathcal{ALC}$-fragment with $G_1 \models G_2$, we have a proof for $G_1 \vdash G_2$ within the full system of RGs, but it might happen that the proof needs graphs that do not belong to the $\mathcal{ALC}$-fragment. In the calculus we provide we require additional rules of this type. Besides trivial rules, like rules that capture the associativity of conjunction, we need special rules for handling roles. The rules *iteration of roles into even* and *deiteration of roles from odd* are the most important examples.

Next in the presentation 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 illustrating them. For each rule name we provide an abbreviation that will be used in the proofs.

Definition 3 (Calculus). *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 $\mathbf{T}$ be an $\mathcal{ALC}$-tree, let $\mathbf{S} \subseteq \mathbf{T}$ be a subtree. For $\mathbf{T}' := \mathbf{T}[\,\mathbf{S} \sqcap \top \,/\, \mathbf{S}\,]$ 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 or removing a branch to/from a heavily drawn line. 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.*

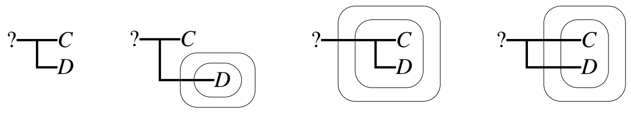


Addition and Removal of Roles (R-add. and R-rem.): *Let $\mathbf{T}$ be an $\mathcal{ALC}$-tree with a subtree $\mathbf{S} \subseteq \mathbf{T}$. Let R be a role name. Then for $\mathbf{T}[\,\neg R \neg \top \,/\, \top\,]$ we set $\mathbf{T} \dashv\vdash \mathbf{T}'$. We say that $\mathbf{T}'$ is derived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

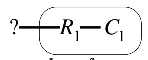
Consider the four graphs below. The second and the third graph can be derived from the first by adding a double negation. Inferences in the opposite direction can also be carried out. The fourth graph is a result of adding a double negation in the general *theory of RGs, but in the system of $\mathcal{ALC}$-trees, this is even not a diagram of an $\mathcal{ALC}$-tree, as the cuts cross more than one heavily drawn line.*



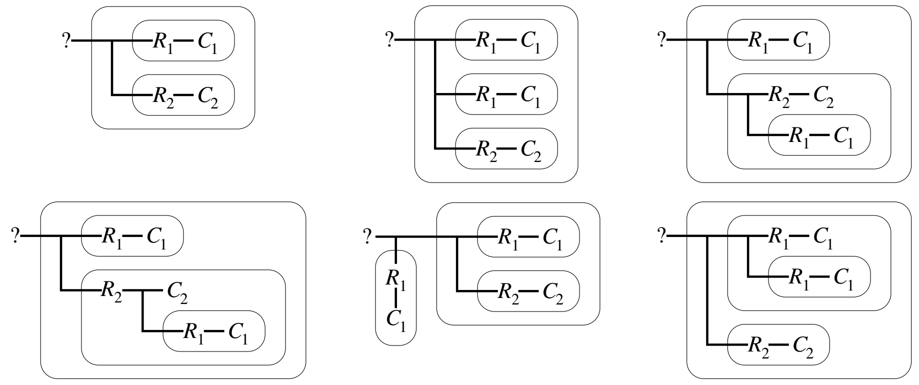
Erasure from even, Insertion into odd (era. and ins.): *Let $\mathbf{T} :=$ be an $\mathcal{ALC}$-tree with a positively enclosed subtree $\mathbf{S} \subseteq \mathbf{T}$. Then for $\mathbf{T}' := \mathbf{T}[\,\top \,/\, \mathbf{S}\,]$ we set $\mathbf{T} \vdash \mathbf{T}'$. We say that $\mathbf{T}'$ 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 that often hold together with the addition and removal of $\top$. Examples will be given later.

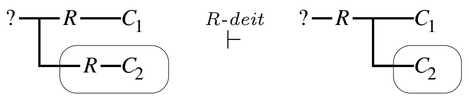
Iteration and Deiteration (it. and deit.): *Let* $\mathbf{T} := (T, \geq, \nu)$ *be an* $\mathcal{ALC}$*-tree with with a subtree* $\mathbf{S} := (S, \geq_S, \nu_S) \subseteq \mathbf{T}$. *Let* s *be the greatest element of* $\mathbf{S}$, *let* t *be the parent node of* s *in* $\mathbf{T}$. *Let* $\nu(t) = \sqcap$, *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* $\mathbf{T} \dashv\vdash \mathbf{T}'$. *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 the most complex rules. Consider the following six Peirce graphs. The second and the third graph can be derived from the first graph by iterating the subgraph* 

(preceded 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* $v < t$. *Finally, the sixth graph violates* $v \notin S$.



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* $\mathbf{S}_a := R\mathbf{S}_1 \sqcap \neg R\mathbf{S}_2$ *and* $\mathbf{S}_b := R(\mathbf{S}_1 \sqcap \neg \mathbf{S}_2)$. *If* $\mathbf{S}_a \subseteq \mathbf{T}$ *is positively enclosed, 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 negatively enclosed, 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 is provided.*



Definition 4 (Proof). *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. Let* $\mathfrak{T}$ *be a set of* $\mathcal{ALC}$*-Trees and let* $\mathbf{T}$ *be an* $\mathcal{ALC}$*-Tree. We set* $\mathfrak{T} \vdash \mathbf{T}$ *if and only if there are* $\mathbf{T}_1, \ldots, \mathbf{T}_n \in \mathfrak{T}$ *with* $\mathbf{T}_1 \sqcap \ldots \sqcap \mathbf{T}_n \vdash \mathbf{T}$

As proved in [14], the calculus is sound and complete.

Theorem 1 (Soundness and Completeness). *Let* $\mathfrak{T}$ *be a set of* $\mathcal{ALC}$*-Trees and let* $\mathbf{T}$ *be an* $\mathcal{ALC}$*-Tree. Then we have* $\mathfrak{T} \models \mathbf{T} \iff \mathfrak{T} \vdash \mathbf{T}$.

5 Metarules and Examples

In this section we firstly present two helpful metarules and then some examples to illustrate the Peirce-style calculus for $\mathcal{ALC}$.

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 that can be applied to positively enclosed subtrees, then we always have a rule in the converse direction that can be applied to negatively enclosed subtrees (and visa versa). Due to these structural properties of rules, we immediately obtain the following helpful lemma (adapted from [9]).

Lemma 1 (Cut-and-Paste). *Let $\mathbf{S}_a, \mathbf{S}_b$ be two $\mathcal{ALC}$-trees with $\mathbf{S}_a \vdash \mathbf{S}_b$. Let $\mathbf{T}$ be an $\mathcal{ALC}$-tree. Then if $\mathbf{S}_a \subseteq \mathbf{T}$ is a positively enclosed subtree of $\mathbf{T}$, we have $\mathbf{T} \vdash \mathbf{T}[\,\mathbf{S}_b\,/\,\mathbf{S}_a\,]$. Visa versa, if $\mathbf{S}_b \subseteq \mathbf{T}$ is negatively enclosed, we have $\mathbf{T} \vdash \mathbf{T}[\,\mathbf{S}_a\,/\,\mathbf{S}_b\,]$.*

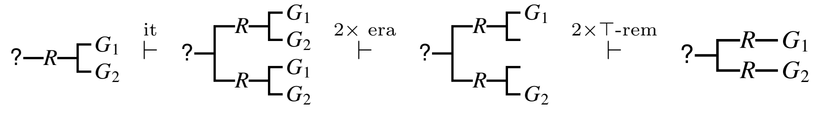
An immediate consequence of the lemma is: If we have two $\mathcal{ALC}$-trees $\mathbf{T}_a, \mathbf{T}_b$ and if $(\mathbf{T}_1, \ldots, \mathbf{T}_n)$ is a proof for $\mathbf{T}_a \vdash \mathbf{T}_b$, then $(\neg\mathbf{T}_n, \ldots, \neg\mathbf{T}_1)$ is a proof for $\neg\mathbf{T}_b \vdash \neg\mathbf{T}_a$. This will be used later.

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

Theorem 2 (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 $\mathfrak{T} \cup \{\mathbf{T}_1\} \vdash \mathbf{T}_2 \iff \mathfrak{T} \vdash \neg(\mathbf{T}_1 \sqcap \neg\mathbf{T}_2)$.*

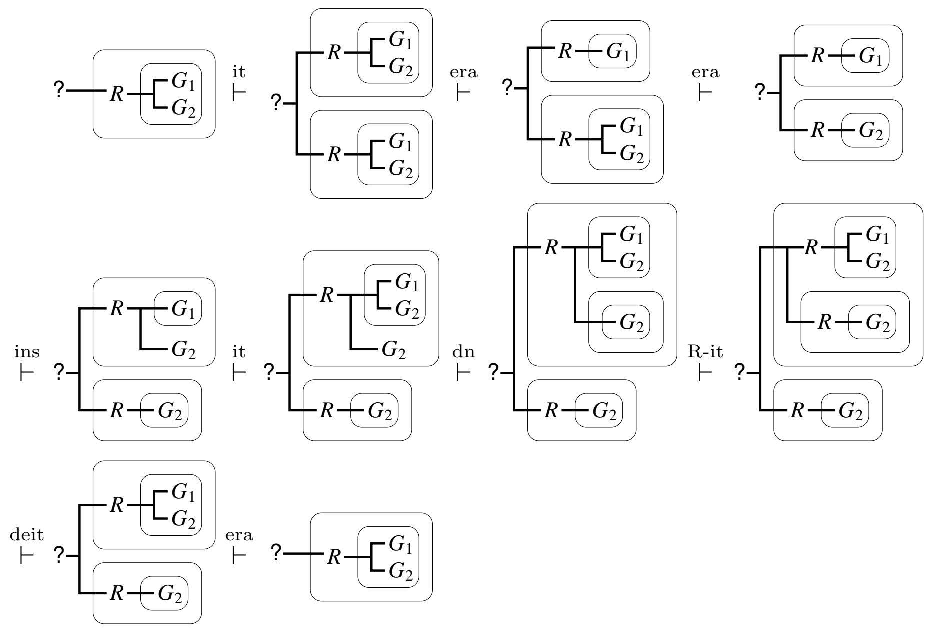
Proof: See [14]

In the following, some examples that correspond to distributing quantifiers in $\mathcal{ALC}$-concepts are provided. We start by moving existential quantifiers inwardly, when followed by a conjunction. In the symbolic notation of $\mathcal{ALC}$, we have the following subsumption relation: $\exists R.(C_1 \sqcap C_2) \sqsubseteq \exists R.C_1 \sqcap \exists R.C_2$. A proof for this relation by means of $\mathcal{ALC}$-trees is given below. In this and the next proof we assume $\Psi(C_1) = ?\!-\!\!-G_1$ and $\Psi(C_2) = ?\!-\!\!-G_2$.

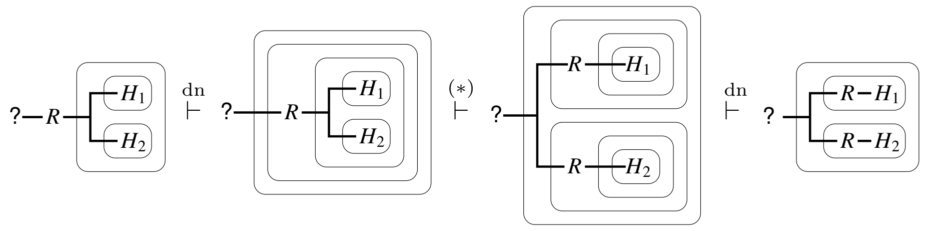


Next, we consider the equivalence of $\forall R.(C_1 \sqcap C_2)$ and $\forall R.C_1 \sqcap \forall R.C_2$. Below, a formal proof with Peirce's graphs is provided. The first and the (identical) last graph correspond to the concept $\forall R.(C_1 \sqcap C_2)$, and the fourth graph corresponds to the concept $\forall R.C_1 \sqcap \forall R.C_2$. So the first three steps prove the subsumption relation $\forall R.(C_1 \sqcap C_2) \sqsubseteq \forall R.C_1 \sqcap \forall R.C_2$, and the last six steps prove the subsumption relation $\forall R.C_1 \sqcap \forall R.C_2 \sqsubseteq \forall R.(C_1 \sqcap C_2)$.

As we have already said, the deiteration-rule and the erasure rule are usually followed by the $\top$-removal rule, and conversely, the iteration rule and the insertion rule are usually preceded by the $\top$-addition rule. In the proof, these two steps are combined without explicitly mentioning the $\top$-removal/addition rule.



A similar example is the equivalence of $\exists R.(C_1 \sqcup C_2)$ and $\exists R.C_1 \sqcup \forall R.C_2$. Compared to the previous example, we exchanged the quantifiers $\forall$ and $\exists$ and the junctors $\sqcap$ and $\sqcup$. The symmetry of these examples is (thanks to Lem. 1), reflected by the calculus. The proof for $\exists R.(C_1 \sqcup C_2) \sqsubseteq \exists R.C_1 \sqcup \forall R.C_2$ is given below. In order to render this proof more understandable, we now set $\Psi(C_1) = ?\!—\!H_1$ and $\Psi(C_2) = ?\!—\!H_2$. The interesting part of the proof is the middle step, i.e., $(*)$. According to the remark after Lem. 1, we can carry out the last 6 steps in the previous proof in the inverse direction, if each graph in this proof is additionally enclosed by a cut (then the proof consists of the rules insertion, deiteration, R-deiteration, double negation, deiteration and erasure). When we replace in this proof each subgraph $?\!—\!G_1$ by $?\!—\!\boxed{H_1}$ and each subgraph $?\!—\!G_2$ by $?\!—\!\boxed{H_2}$, we obtain the proof for $(*)$. The steps before and after $(*)$ are simple helper steps in order to add or remove some double negations.



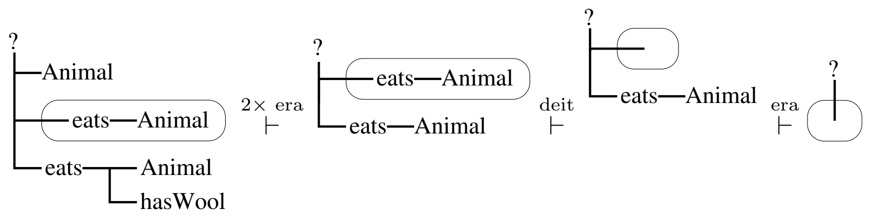
We see that the the proof for $\exists R.(C_1 \sqcup C_2) \sqsubseteq \exists R.C_1 \sqcup \forall R.C_2$ is essentially the inverse direction of the proof for $\forall R.C_1 \sqcap \forall R.C_2 \sqsubseteq \forall R.(C_1 \sqcap C_2)$. The proof for

$\exists R.C_1 \sqcup \forall R.C_2 \sqsubseteq \exists R.(C_1 \sqcup C_2)$ can be obtained similarly from the proof for $\forall R.(C_1 \sqcap C_2) \sqsubseteq \forall R.C_1 \sqcap \forall R.C_2$. This shows some benefit of the symmetry of Peirce's rules.

The final example, the *mad cow ontology*, is a popular example for $\mathcal{ALC}$-reasoning. Consider the following $\mathcal{ALC}$-definitions:

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

A question to answer is whether this ontology is consistent. Such a 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 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. Again, each application of the erasure-rule is followed by an application of the $\top$-removal rule (although this is not explicitly mentioned in the proof).



We started with the Peirce graph for the given concept and derived the absurd concept, thus the ontology is not satisfiable. The madcow ontology is therefore inconsistent and this is proved using the calculus developed,

6 Conclusion and Further Research

This paper provides steps toward a diagrammatic representation of DLs, including importantly 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 RGs further as diagrammatic versions of corresponding DLs.

The approach taken can also be extended to other variants of DL. For instance, a major task is to incorporate individuals, 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 DLs more user-friendly through a diagrammatic correspondence, diagrammatic systems need to be evaluated against the traditional textual form of DL in order to measure readability improvement. Cognition and usability experiments with such a evaluation in mind are planned as future work.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. In Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: Description Logic Handbook, Cambridge University Press (2003)
2. Larkin, J.H., Simon, H.A.: Why a diagram is (sometimes) worth ten thousand words. Cognitive Science **11**(1) (1987) 65–100
3. Shimojima, A.: On the Efficacy of Representation. PhD thesis, The Department of Philosophy, Indiana University (1996) Available at: `http://www.jaist.ac.jp/ ashimoji/e-papers.html`.
4. Shin, S.J.: The Iconic Logic of Peirce's Graphs. Bradford Book, Massachusetts (2002)
5. Gaines, B.R.: An interactive visual language for term subsumption languages. In: IJCAI. (1991) 817–823
6. Kremer, R.: Visual languages for konwledge representation. In: Proc. of 11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98) Banff, Alberta, Canada, Morgan Kaufmann (1998)
7. Nosek, J.T., Roth, I.: A comparison of formal knowledge representation schemes as communication tools: Predicate logic vs semantic network. International Journal of Man-Machine Studies **33**(2) (1990) 227–239
8. Coupey, P., Faron, C.: Towards correspondence between conceptual graphs and description logics. In Mugnier, M.L., Chein, M., eds.: ICCS. Volume 1453 of LNAI., Springer, Berlin – Heidelberg – New York (1998) 165–178
9. Sowa, J.F.: Conceptual structures: information processing in mind and machine. Addison-Wesley, Reading, Mass. (1984)
10. Brockmans, S., Volz, R., Eberhart, A., Löffler, P.: Visual modeling of owl dl ontologies using uml. In McIlraith, S.A., Plexousakis, D., van Harmelen, F., eds.: International Semantic Web Conference. Volume 3298 of Lecture Notes in Computer Science., Springer, Berlin – Heidelberg – New York (2004) 198–213
11. Stapleton, G.: Reasoning with Constraint Diagrams. PhD thesis, Visual Modelling Group, Department of Mathematical Sciences, University of Brighton (2004) Available at: `http://www.cmis.brighton.ac.uk/Research/vmg/GStapletonthesis.html`.
12. Stapleton, G., Howse, J., Taylor, J.: Spider diagrams. LMS Journal of Computation and Mathematics **8** (2005) 145–194
13. Zeman, J.J.: The Graphical Logic of C. S. Peirce. PhD thesis, University of Chicago (1964) Available at: `http://www.clas.ufl.edu/users/jzeman/`.
14. Dau, F., Eklund, P.: Towards a diagrammatic reasoning system for description logics. Submitted to the Journal of Visual Languages and Computing, Elsevier. Available at `www.kvocentral.org`. (2006)
15. Burch, R.W.: A Peircean Reduction Thesis: The Foundation of Topological Logic. Texas Tech. University Press, Texas, Lubbock (1991)
16. Pollandt, S.: Relation graphs: A structure for representing relations in contextual logic of relations. In Priss, U., Corbett, D., Angelova, G., eds.: Conceptual Structures: Integration and Interfaces. Volume 2393 of LNAI., Borovets, Bulgaria, July, 15–19, Springer, Berlin – Heidelberg – New York (2002) 24–48
17. Dau, F.: Fixing shin's reading algorithm for peirce's existential graphs. In Barker-Plummer, D., Cox, R., Swoboda, N., eds.: Diagrams. Volume 4045 of LNAI., Springer, Berlin – Heidelberg – New York (2006) 88–92