A Diagrammatic Reasoning System for the Description Logic $\mathcal{ALC}$

Frithjof Dau, Peter Eklund

University of Wollongong, Wollongong, Australia

Abstract

Diagrammatic reasoning is a tradition of visual logic that allows sentences that are equivalent to first order logic to be written in a visual or structural form: usually for improved usability. A calculus for the diagram can then be defined that allows well-formed formulas to be derived. This calculus is intended is the analog of logical inference.

Description logics (DLs) have become a popular knowledge representation and processing language. DLs correspond to decidable fragments of first order logic; their notation is in the style of symbolic, variable-free formulas. Moreover, DLs are equipped with tableau theorem provers that are proven to be sound and complete.

Although DLs have roots in diagrammatic languages (such as semantic networks), they are elaborated in a purely symbolic manner. This paper discusses how DLs can be equivalently represented in terms of a diagrammatic reasoning system.

First, existing diagrammatic reasoning systems, namely spider- and constraint diagrams, as well as existential and conceptual graphs, are investigated to determine if they are compatible with DLs. It turns out that Peirce's existential graphs are better suited for this purpose than the alternatives we examine.

The paper then redevelops the DL $\mathcal{ALC}$, which is the smallest propositional DL, by means of labeled trees, and provides a diagrammatic representation for these trees in the style of Peircean graphs. We provide a calculus based on C.S. Peirce's calculus for existential graphs and prove the soundness and completeness of the calculus. The calculus acts on labeled trees, but can be best understood as a diagrammatic calculus whose rules modify the Peircean-style representation of $\mathcal{ALC}$.

Key words:
Description Logics, Spider Diagrams, Constraint Diagrams, Conceptual Graphs, Existential Graphs, Logic, Syntax, Semantics, Calculus

1. Introduction

Description logics (DLs) are a common family of knowledge representation languages tailored to express knowledge about concepts and concept hierarchies. They include sound and complete decision procedures for reasoning about such knowledge. One of the main applications of DLs is their use as the basis for an ontology language, especially popular for the Semantic Web. In particular, the Ontology Web Language (OWL) – a W3C recommendation for the knowledge language of the Semantic Web – is based on a specific and expressive DL termed $\mathcal{SHOIN}(D)$ [1].

The basic building blocks of DLs are concepts (unary predicates), roles (binary relations) and sometimes individuals, which can be composed by language constructs such as intersection, union, value or number restrictions to build more complex well-formed formulas that themselves represent complex concepts and roles. For example, when Man, Female, Male, Rich, Happy are predefined concepts and if hasChild is a predefined role, then

$$\text{Man} \sqcap \exists \text{hasChild}.\text{Female} \sqcap \exists \text{hasChild}.\text{Male} \sqcap \forall \text{hasChild}.(\text{Rich} \sqcup \text{Happy}) \quad (1)$$

describes the concepts of men who have both male and female children, and where all the children are rich or happy. Let us call a concept defined in this way as 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, and like FOL, DLs have a well-defined, formal syntax, a semantics in the form of Tarski-style models, and a sound and complete calculi (based on Tableaux-algorithms). It is often emphasised that DLs offer, in contrast to other knowledge representation languages, sound, complete and (empirically) tractable reasoning services. A comprehensive overview on DLs is given in the *Description Logic Handbook* [4].

The notation of DLs is in the style of the usual linear and symbolic [2] notations of FOL. The fact that the notation of DLs is variable-free makes them easier to comprehend than the common FOL formulas which include free variables. Nonetheless, for untrained users, the symbolic notation of DLs can be hard to learn and comprehend.

A main alternative to the symbolic notation is the development of a diagrammatic representation of DLs. It is well accepted that diagrams are in many cases easier to comprehend than symbolic notations (see for example [48,65,8,67]), and in particular it has been argued that they are useful for knowledge representation systems [35,46]. This has been acknowledged by the DL community and is a common view among the broader knowledge representation community [53]. In [52], the introduction to the *Description Logic Handbook*, Nardi and Brachman write that besides the possibility of "providing a syntax that resembles more closely natural language", a "major alternative for increasing the usability of Description Logics as a modeling language" is to "implement interfaces where the user can specify the representation structures through graphical operations."

Email address: dau@dr-dau.net,peklund@uow.edu.au (Frithjof Dau, Peter Eklund).

[1] In $\mathcal{SHOIN}(D)$, $\mathcal{S}$ stands for the basic DL $\mathcal{ALC}$ (equivalent to the propositional modal logic extended with transitive roles), $\mathcal{H}$ stands for role hierarchies, $\mathcal{O}$ stands for nominals (classes whose extension is a single individual), $\mathcal{N}$ stands for unqualified number restrictions and D stands for datatypes)[42]

[2] We use the term 'symbolic' according to C.S. Peirce's classification of signs into 'icons', 'indices' and 'symbols'. See Shin [67] for an introduction.

A first attempt at a diagrammatic representation for DL is can be found in [35], where Gaines elaborates a graph-based representation for the textual DL CLASSIC, part of the KL-ONE-framework. More recently, the focus has shifted from the development of proprietary diagrammatic representations to representations within the framework of UML (Unified Modeling Language). In 2003, the *Object Management Group* requested a metamodel for the purpose of defining ontologies. Following this proposal, [12] provides a UML-based, diagrammatic representation for the OWL DL. In these approaches, the focus is on a graphical *representation* of DL, however, as emphasized in many works on DL (see for example [4]), *reasoning* is seen as a distinguishing feature of DL and such reasoning is not supported diagrammatically by that treatment. Correspondences between graphical representation of the DL and the DL reasoning system are therefore important but remain largely unelaborated to date. Similar arguments hold for other popular diagrammatic languages like UML and ORM [3] the difference being that, that unlike DLs, these diagrammatic modeling languages provide no extensional, mathematical semantics, nor any automated reasoning facilities.

On the other hand, there are some candidate diagrammatic reasoning systems that have the expressiveness of fragments of FOL or even full FOL. In this paper, we will evaluate two families of contemporary mathematical diagrammatic reasoning systems, which have the following two (historical) origins:

(i) The system of Euler Circles and Venn-Diagrams, the latter enriched by C.S. Peirce to Venn-Peirce-Diagrams (see [66,38]).
(ii) The system of Peirce's Existential Graphs [4].

The first system is the background to the contemporary development of spider- and constraint diagrams, the latter is background to a contemporary interpretation as conceptual graphs [70]. Why is it worth considering these diagrammatic reasoning systems as a starting point for a diagrammatic version of DL? Firstly, for all these systems mathematical elaborations in the general style of mathematical logic exists. These include:

- A well-defined syntax: usually, the syntax is defined at an abstract level (for example in terms of graph theory), such that the well-formed formulas have diagrammatic representations. Moreover, the syntax is – like the syntax for DL – variable-free.
- Extensional – Tarski-style semantics – and/or translations to formulas of FOL.
- Sound and complete calculi: usually the rules are defined on the abstract syntax but mostly they can be understood as manipulations of the represented diagrams.

Therefore, in contrast to data modeling languages like UML and ORM, we have a well-defined syntax and semantics for the candidate diagrammatic representations. Moreover, if we adopt one of these systems, we can adopt (partly or completely) the existing calculi as a diagrammatic reasoning service for DL. Finally, over and above the general arguments for diagrammatic systems, some of the systems investigated in this paper – constraint and spider diagrams and conceptual graphs – have had their diagrammatic benefits investigated in user-evaluations [83,64,31,30].

As a specific DL is the corresponding logic behind the OWL, the results of this paper benefit not only DL, but the Semantic Web more generally. Developing a Semantic Web language as a mathematical diagrammatic reasoning system has already been carried

[3] http://www.orm.net/index.html
[4] http://www.existentialgraphs.com/ or (a different site) http://www.existential-graphs.net/

out for a much simpler language: namely RDF. For RDF, mathematical elaborations based on graph theory, including a Tarski-style semantics and a sound and complete calculus based on "projections" (see [5,6]) or via diagrammatic rules (see [24]) have been elaborated.

In the next three Sections, 2, 3 and 4, the basic notions of description logics, spider- and constraint diagrams, and existential- and conceptual graphs, are introduced. In the subsequent Sections 5 and 6, we investigate whether spider- and constraint diagrams and existential- and conceptual graphs respectively are suited as a starting point for developing DL as a diagrammatic reasoning system. It will turn out that there are prospects to use spider- and constraint diagrams for this purpose. However, conceptual graphs and, as we will show, existential graphs are a much better match. In the three Sections that follow, $\mathcal{ALC}$ is developed as a diagrammatic reasoning system based on Peirce's existential graphs. In Section 7, a formalization of $\mathcal{ALC}$ by means of labeled trees is given. Section 8 provides the Peirce style representation of such trees. In Section 9, a calculus for $\mathcal{ALC}$-trees, based on Peirce's calculus for existential graphs, is given, and soundness and completeness proven. Finally, in Section 10, further research is discussed. As we use several abbreviations in this paper, these are presented in Section 11.

2. Introduction into Description Logics

The vocabulary of any DL consists of CONCEPTS, which denote sets of individuals, and ROLES, which denote binary relationships between individuals. The starting points are ATOMIC CONCEPTS AND ROLES – CONCEPT AND ROLE NAMES – from which more complex concepts and roles are built with constructs such as intersection, union, value or number restrictions, and so on. Moreover, we usually consider concept vocabularies where we have a universal concept $\top$. The tables of Figures 1 and 2 provide an overview of the most important entities and constructors. In both tables, the first column provides the common name for the entity or constructor, in the second column the syntax is presented.

In DLs concept descriptions gain meaning when they are interpreted in a model. An INTERPRETATION is a pair $(\Delta^{\mathcal{I}}, \mathcal{I})$, consisting of an nonempty DOMAIN OF THE INTERPRETATION Δ and INTERPRETATION FUNCTION $\mathcal{I}$ which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. For each atomic role, let $\mathcal{I}(R^{-}) = \{(b, a) \mid (a, b) \in \mathcal{I}(R)\}$, we require $\mathcal{I}(\top) = \Delta^{\mathcal{I}}$. The interpretation function is then extended to arbitrary roles and concept descriptions, as depicted in the third column of the tables of Figures 1 and 2. In the third column of Figure 2 we use the following notation which will be used in the next section as well: if U is a set, $S \subseteq U \times U$ is a relation, and if $A \subseteq U$ and $a \in U$, then let $A.R$ denotes the set $\{u \in U \mid \exists a \in A.aRu\}$ (the image of A under R). The set $\{a\}.R$ is abbreviated $a.R$. The sets $R.A$ and $R.a$ are defined analogously.

Depending on the choice of constructors we get different DLs with different levels of expressiveness. The last column of the table in Figure 2 provides the common letters which are used to abbreviate the constructors. In this paper, we focus on the description logic $\mathcal{ALC}$. Literally, this logic has conjunction, negation and value restriction as constructors, but as $C_1 \sqcup C_2$ is equivalent to $\neg(\neg C_1 \sqcap \neg C_2)$, and as $\exists R.C$ can be replaced by $\neg\forall R.\neg C$, it is also assumed that $\mathcal{ALC}$ contains disjunction and existential quantification, i.e. $\mathcal{ALC}$

is the smallest propositionally closed description logic.

atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
universal concept	$\top$	$\Delta^{\mathcal{I}}$
atomic role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
individuals	a	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

Fig. 1. Entitites in Description Logics

Class Constructors

Construct Name	Syntax	Semantics	Symbol
conjunction	$C_1 \sqcap C_2$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$\mathcal{AL}$
disjunction	$C_1 \sqcup C_2$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}$
negation	$\neg C_1$	$\Delta^{\mathcal{I}} - C^{\mathcal{I}}$	$\mathcal{C}$
value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in xR^{\mathcal{I}} : y \in C^{\mathcal{I}}\}$	$\mathcal{A}$
exists restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in xR^{\mathcal{I}} : y \in C^{\mathcal{I}}\}$	$\mathcal{E}$
nominals	$\{o_1, \ldots, o_n\}$	$\{o_1^{\mathcal{I}}, \ldots o_n^{\mathcal{I}}\}$	$\mathcal{O}$
unqualified number restriction	$\geq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert xR^{\mathcal{I}}\rvert \geq n\}$	$\mathcal{N}$
unqualified number restriction	$\leq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert xR^{\mathcal{I}}\rvert \leq n\}$	$\mathcal{N}$
qualified number restriction	$\geq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert xR^{\mathcal{I}} \cap C^{\mathcal{I}}\rvert \geq n\}$	$\mathcal{Q}$
qualified number restriction	$\leq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert xR^{\mathcal{I}} \cap C^{\mathcal{I}}\rvert \leq n\}$	$\mathcal{Q}$

Role Constructors

Construct Name	Syntax	Semantics	Symbol
inverse role	R^-	$\{(y, x) \mid (x, y) \in R^{\mathcal{I}}\}$	$\mathcal{I}$
role intersection	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$	$\mathcal{R}$

Fig. 2. Constructors in Description Logics

3. Spider and Constraint Diagrams

Spider diagrams (SDs) and constraint diagrams (CDs) are a diagrammatic reasoning system based on Euler circles and Venn-Peirce diagrams. They are inspired by a fragment of UML which is used for software specifications as an alternative to UML's textual Object Constraint Language (OCL). They are elaborated as abstract mathematical structures, including an extensional semantics and inference rules. In this section, SDs and CDs are briefly and informally introduced. Particularly, only the *diagrams* of SDs and CDs are provided, not the underlying mathematical structures. For the formal definitions we refer the reader to [77,79].

3.1. *Spider Diagrams*

Since the first paper on SDs [37], several elaborations of the ideas have been published [84]. Most variations of SDs are based on Venn diagrams as a foundation, which can be propositionally combined in conjunctive normal form. It turns out however that SDs foundation in Venn-diagrams renders many of them difficult to read: the restriction of combining these diagrams in conjunctive normal form yields a number of technical problems that result from the rendering of four or more sets. The final version of SDs has been published in the PhD thesis of Stapleton [77] – a useful 20 page overview can be found at [84]. Another source is [79], which provides the definitive SDs paper. In that treatment SDs are a combination of Venn diagrams and the more user friendly Euler diagrams [5] and the restriction to conjunctive normal form is removed. Below, two examples of so-called *unary* SDs are depicted.

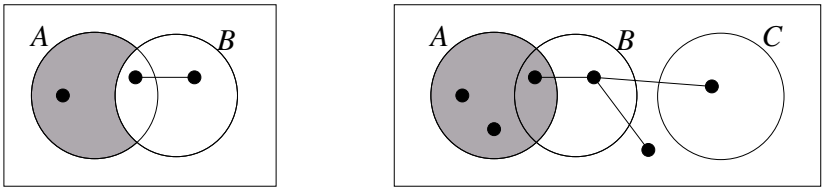


Fig. 3. Two Spider Diagrams

As in Venn diagrams the bounding rectangles depict the UNIVERSE (OF DISCOURSE), i.e., the ground set of the respective models in which the diagrams are evaluated. We use the letter U to denote the universe of discourse. The first diagram to the left has two CONTOURS: these are the circles labeled A and B, representing the sets A and B. The diagram has four ZONES which represent all combinations of A and B, i.e. $A \cap B$, $A - B$, $B - A$, and $U - (A \cup B)$. As all possible combinations of A and B are represented, this SD is a VENN-PEIRCE diagram as well. The diagram contains two spiders: The shaded zone representing $A - B$ is the HABITAT of the first spider, the REGION composed of the zones $A - B$ and $B - A$, i.e. the region which represents B, is the habitat of the second spider. Each spider denotes a uniquely given object (i.e., different spiders necessarily denote different objects) which is a member of its habitat. In VENN-PEIRCE diagrams, shading a region means that the corresponding set is empty. Thus, when read as a VENN-PEIRCE diagram, the diagram left contains the information that – due to the shading – $A - B$ is empty, and – due to the spider – there is an element in $A - B$, so the diagram is contradictory. In SDs, the semantics for shading is slightly different from that of VENN-PEIRCE diagrams: a region does not contain *more* elements than the elements represented by some spiders. So the SDs reads as follows: there are two sets A and B, the set $A - B$ contains exactly one element, and the set B contains at least one element.

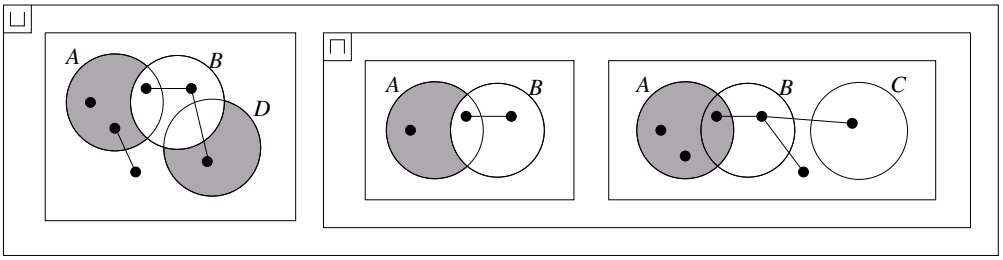
In the second diagram to the right, a third contour representing a set C is involved. This diagram uses the notions of Euler circles: as the contour labeled C does not overlap with the A and B-contours, C and $A \cup B$ are disjoint [6]. Moreover, there are three elements u,

[5] A survey of Euler diagram-based reasoning systems and notations is given in [78], or in the introduction of [77].

[6] The usage of disjoint features in Euler circles has a drawback: not all abstract SDs are drawable (see [79]).

v and w (represented by the three spiders) such that $u, v \in A$ and $w \in U - A$. Further, due to its shading, the set $A - B$ must not contain any other elements than u and v, i.e. it contains exactly two elements, and $A \cap B$ must not contain any other elements than w, i.e., it contains no element or at most one element.

The above diagrams are UNARY SDs. They can be propositionally combined with the logical operators $\sqcap$ ('and') and $\sqcup$ ('or'). Below is an example of an SD which uses these conjunctors is shown.



In SDs, the spiders can be understood to correspond to existentially quantified objects. Later in our presentation, a different kind of spider, which corresponds to universally quantified objects in constraint diagrams, is introduced; this then allows one to speak about *existential* and *universal* spiders. Nonetheless, since zones can be shaded, we can express universal statements as well. In fact, Stapleton [77] has shown that SDs are equivalent in expressive power to monadic FOL *with equality*, MFOLe [7] . Therefore the system of SDs is semantically an extension of Shin's Venn-II-system [66] which is equivalent in expressive power to monadic FOL *without equality*. This feature is obtained by the change of the semantics that includes the use of shading.

The Stapleton system of SDs [77] is equipped with a sound and complete calculus which is a combination of five diagrammatic rules and of eight 'propositional logic style'-rules that encompass the fact that unary diagrams can be assembled with the propositional conjunctors $\sqcap$ and $\sqcup$. Furthermore, Stapleton proved that the system is decidable [8] , which makes SDs attractive for application to DLs.

Finally, there have been attempts to develop automated theorem provers for SDs. Flower in [32,33] investigates a heuristic approach to generating proofs in the SD system and further in [34] presents a theorem proving tool: available from [84]. Moreover, a tableaux system for SDs has also been implemented by Patrascoiu [56].

3.2. *Constraint Diagrams*

SDs only allow reasoning about sets, namely unary predicates, and provide no possibility to represent or reason about any sort of relations. Moreover, as already mentioned, the

[7] The proof is sketched as follows: in one direction, it is easy to assign to each diagram a corresponding MFOLe-formula. The other direction is more challenging: first, to each MFOLe-formula, a finite set of finite models which are prototypic for *all* models of the formula are assigned, and to each of these models a unary SD is assigned.

[8] It has to be emphasised that in the semantics for SDs (and CDs), in contrast to FOL, allows *empty* models. For this reason, one cannot immediately transfer decidability results for fragments of FOL – like the well-known Bernays-Schönfinkel-fragment of FOL – to SDs.

spiders in SDs are 'existential spiders' as they can be read as existentially quantified objects. On the other hand, constraint diagrams are an extension of SDs with UNIVERSAL SPIDERS (quantifiers) and ARROWS which represent binary relations (to be precise: the arrow labels stand for binary relations, thus an arrow represents a property of the relation denoted by its label). A full constraint notation was introduced by Kent [45] in an informal manner. Since then, several papers attempt to elaborate a full mathematical treatment of Kent's vision, including syntax, semantics, and a sound and complete calculus for constraint diagrams. Let us first depict a constraint diagram (taken from [77]):

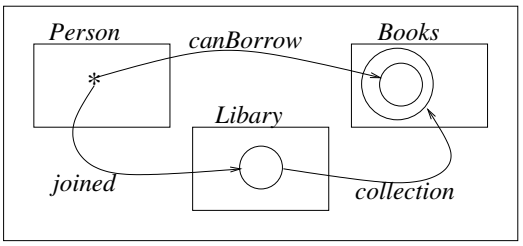


Fig. 4. A Constraint Diagram

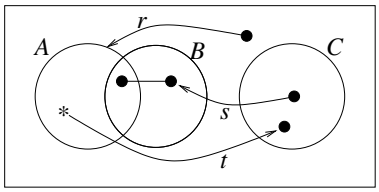
The core information expressed in this diagram is that "every person can only borrow books that are in the collections of libraries they have joined".

In Fish et al [29], the syntax and semantics of full constraint diagrams is developed but a sound and complete calculus is elusive. The first ever constraint reasoning system (i.e., including a sound and complete calculus) was developed by Stapleton [77] but compared to Kent's approach it has several limitations. For example, universal spiders inhabit single zones only, with at most one spider in any given zone (in Kent's full system, universal spiders are allowed to inhabit more than one zone, and a zone could contain more than one universal spider), and each universal spider in a unitary diagram must be the source of at least one arrow. In Kent's full constraint diagram systems there are unlabeled contours, termed DERIVED CONTOURS. For example, the targets of the arrows labeled with 'canBorrow' and 'collection' in Figure 4 are derived contours. In Stapleton's system, there is at most one derived contour, which must be the target of at least one arrow. Moreover, the inside of a derived contour is always shaded, and must therefore necessarily represent the empty set.

In Stapleton's constraint diagrams, the source of an arrow is an (existential or universal) spider, and its target can either be an existential spider or a contour. Depending on the source, we distinguish existential and universal arrows. As each universal spider is the source of at least one arrow, in order to clarify the semantics of Stapleton's constraint diagrams, we have to make clear how arrows are read. As already stated however, the label of an arrow denotes a binary relation on the universe U.

Let l be an existential arrow with source s (an existential spider) and target t (an existential spider or a contour). Let R be the relation represented by the label of l. Let S be the object denoted by s and let T be the set denoted by t (for simplicity reasons we agree that an existential spider denotes a singleton set). Then l stands for the condition $S.R = T$. If l is an universal spider, the source now stands for *the set* denoted by the contour of the spider's habitat, i.e., if the habitat of l is the contour c which represents the set C, then *each element* $x \in C$ has to satisfy the condition $x.R = T$. In the semantics

existential quantifiers take precedence over universal quantifiers (i.e., existential spiders are read first). For example, we consider the following constraint diagram:



The reading of the diagram is as follows:

$$\overbrace{C \cap (A \cup B) = \emptyset}^{\text{compare to Fig. 3}} \wedge \overbrace{\exists u \in U - (A \cup B \cup C).\, \exists b \in B.\, \exists c_1, c_2 \in C :}^{\text{objects denoted by exist. spiders}}$$
$$(c_1 \neq c_2 \wedge \underbrace{uR = A \wedge c_1 S = b \wedge \forall x \in A - B : xT = \{c_2\}}_{\text{reading the arrows}})$$

Similar to SDs, Stapleton [77] developed a sound and complete calculus for constraint diagrams. It contains 25 rules, 22 are needed in the completeness proof (the other three rules are useful shortcuts). In contrast however to SDs, there are unary constraint diagrams which are not satisfiable. To compensate, Stapleton provided a *compatibility relation* on the arrows that can be used to classify satisfiable constraint diagrams.

Finally, after discussing the syntax and semantics of CDs, we need to mention that Fish et al [31,30] provide empirical studies on how human users read CDs.

After this brief introduction into SDs and CDs, we introduce in the next section the system of existential and conceptual graphs.

4. Existential and Conceptual Graphs

In this section, we introduce Peirce's existential graphs (EGs), Sowa's conceptual graphs (CGs), particularly its most common fragment of simple conceptual graphs (SCGs) and the system of concept graphs with cuts (CGwCs) introduced by one of the authors. Similar to spider and constraint diagrams, there exist mathematically precise elaborations for each of these systems. Again, like spider and constraint diagrams, the diagrammatic benefits of EGs or CGs (used in teaching) has been investigated by Schäfe [64] and also by Pollant [83]. Furthemore, a theorem prover for existential graphs by developed by Stewart [80].

4.1. *Existential Graphs*

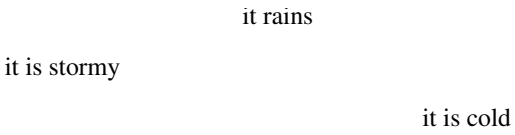
Existential graphs [39] are a diagrammatic logic invented by C.S. Peirce (1839-1914) in the last two decades of his life. Existential graphs are divided into three parts called Alpha, Beta and Gamma. The three parts build on one another, Beta builds upon Alpha, and Gamma builds on both Alpha and Beta. 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 Alpha and Beta are introduced. For further readings we recommend Roberts [62], which is the standard book on EGs and which describes in a more informal manner the complete system; (see also Roberts [63]); Zeman's formal elaboration in [89] (including a fragment of Gamma); [67], a book where Shin provides her interpretation of Alpha and Beta which focuses on its iconic features; [58], which is a tutorial of Peirce himself with commentary by Sowa, or a short introduction by Sowa in [73]. A mathematical elaboration of Alpha and Beta has been produced but is currently unpublished [9].

We start with the description of Alpha. The EGs of Alpha consist only of predicate names of arity zero, which are called MEDADS, and of closed, doublepoint-free curves which are called CUTS (or sometimes SEPS) and used to negate the enclosed subgraph. Medads can be considered as propositions. Propositions can be written down on an area (the term Peirce uses instead of 'writing' is *'scribing'*), and writing down a proposition is to assert it. [10] The area where the proposition is written or drawn (scribed) is what Peirce called the SHEET OF ASSERTION. It may be a sheet of paper, a blackboard or any other surface. Scribing several propositions next to each other (this operation is called a JUXTAPOSITION) asserts the truth of each proposition, i.e. the juxtaposition corresponds to the conjunction of the juxtaposed propositions. For example, scribing the propositions 'it rains', 'it is cold' and 'it is stormy' next to each other yields the graph

which means 'it rains, it is cold, and it is stormy'.

The location of the proposition on the sheet of assertion has no significance. Below, another diagram with the same meaning is given.



The possibility that an existential graph can have different representations is addressed by Peirce at the beginning of [57], where he writes:

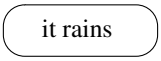
> Convention No. Zero. Any feature of these diagrams that is not expressly or by previous conventions of languages required by the conventions to have a given character may be varied at will. This "convention" is numbered zero, because it is understood in all agreements.

To explain, the two diagrams above are *different representations of the same graph*. 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 graph. Similar

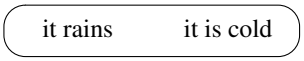
[9] The reader is encourage to refer and correspond to Dau[23] for his forthcoming habilitation thesis on this subject.
[10] An *asserted* proposition is called a judgement.

distinctions are made between *types* and *tokens*, known from philosophy, or *abstract* and *concrete* syntax, used widely in Computer Science. As discussed in [43,21], 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. After evaluating different diagrammatic reasoning systems for using them for DLs, in Section 7, 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.

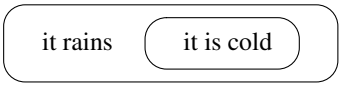
So far, we have only seen how conjunction is expressed in the system of Peirce's graphs. Next, negating a graph is depicted by encirling it. The space within a cut is called its CLOSE or AREA. For example, the graph



has the meaning 'it is not true that it rains', i.e. 'it does not rain'. The graph

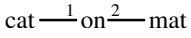


has the meaning 'it is not true that it rains and that it is cold', i.e. 'it does not rain or it is not cold'. Cuts must not overlap, but they may be nested. The next graph has two nested cuts.



This graph has the meaning 'it is not true that it rains and that it is not cold', i.e. 'if it rains, then it is cold'. The device of two nested cuts is called a SCROLL. From the last example we learn that a scroll can be read as an implication. A scroll with nothing on its first area is called DOUBLE CUT (it corresponds to a double negation). As mentioned before, the space within a cut is called the AREA of the cut. In the example above, we therefore have three distinct areas: all the space outside the outer cut, i.e. the sheet of assertion, the space between the outer and the inner cut, which is the area of the outer cut, and the space inside the inner cut, which is the area of the inner cut. An area is ODDLY ENCLOSED if it is enclosed by an odd number of cuts, and it is EVENLY ENCLOSED if it is enclosed by an even number of cuts. As we have the possibility to express conjunction and negation of propositions, we see that Alpha has the expressiveness of propositional logic.

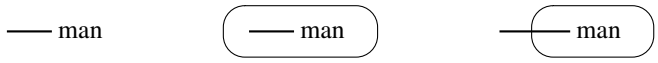
If we go from the Alpha part of EGs to the Beta part, predicate names of arbitrary arity may be used, and a new symbol, the LINE OF IDENTITY, is introduced. Lines of identity are used to denote both the existence of objects and the identity between objects. Lines of identity are attached to predicate names. Peirce drew them in bold. Consider the following graph:



This contains two lines of identity, hence it denotes 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.

Consider the following graphs, where cuts are involved.



The meaning of the first graph is clear: it 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 (it is not a line of identity, which will be discussed shortly) 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 4.116 (we adopt the usual convention to refer to his collected papers [39]), 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, lines of identity are allowed where they connect directly on a cut. Due to this possibility, ligatures are permitted which cross a cut.

Let us first consider the three EGs of Fig. 5.

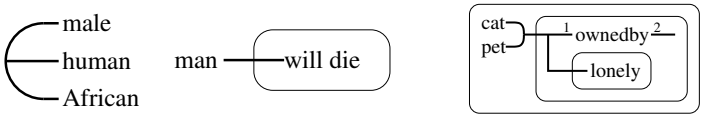


Fig. 5. Four Peirce graphs with so-called single-object-ligatures

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 5 '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'.

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 6. 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.

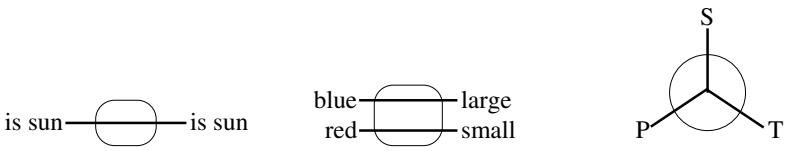


Fig. 6. Three Peirce graphs with non-single-object ligatures

In every graph in Figure 6 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 (for example, Peirce writes (in 4.459) that "a sep [sep is another word Peirce used for cut] which is vacant, except for a line of identity traversing it, expresses with its contents the non-identity of the extremities of that line."), thus a ligature containing such a element l usually denotes different objects. But if such a element does not occur, it can be shown that the ligature denotes a single object. For this reason, a ligature L, such that no part of L traverses any cut, will be called SINGLE-OBJECT-LIGATURE (SO-LIGATURE).

A complete discussion existential graphs with non-single-object-ligatures goes beyond the scope of this paper, see [22] for a more detailed discussion. It will turn out that the ligatures we have to deal with in this paper are all single-object-ligatures, so we will not run into problems caused by non-single-object-ligatures.

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 existential graphs corresponds to FOL (without object names and without function names). Moreover, Peirce equipped EGs with a set of five sound and complete inference rules.

4.2. *Conceptual Graphs*

Sowa [70] developed CGs in the 1970s on the back of EGs (Sowa [71] provides an abbreviated overview). "Conceptual graphs are an extension of existential graphs with features adopted from linguistics and AI" [73], and the purpose of the system is to express meaning in a form that is "logically precise, humanly readable, and computationally tractable." The term 'extension' in the above is to be understood semantically, not syntactically: Sowa adopted the ideas of EGs, but CGs have a different and richer syntax, "Besides Peirce's primitives, CGs provide a means of representing case relations, generalized quantifiers, indexicals and other aspects of natural languages." [73]. Another fundamental aspect of CGs is that different kinds of referents exist, in particular names for objects. Moreover, conceptual graphs may serve as referents themselves (which is similar to a feature of Peirce's Gamma EGs). This is called NESTING in CGs. Both aspects will be explained in the rest of this section.

In order to illustrate the broad range of CGs, we will provide several examples. The first two follow:

The meanings of these graphs are 'a cat is on a mat' and 'the cat Yoyo is on a mat' respectively. In each concept box, we have a TYPE LABEL t and a REFERENT r. Sowa

calls the boxes CONCEPTS, but in this paper the term CONCEPT BOX will be used to avoid confusion with the use of the word concept in DLs. In the CG above, we have the types CAT and MAT. The referents are 'Yoyo' and the GENERIC MARKER '$*$'. Type labels are ordered in a sub-type-relation, according to their level of generality. For example, CAT is a subtype of ANIMAL. An ordered set of types is often called SUPPORT, TYPE HIERARCHY, TAXONOMY or even ONTOLOGY. Type hierarchies usually contain a greatest type $\top$, the UNIVERSAL TYPE, containing every object (of the respective universe of discourse) in its extension. In the graph above, the two concept boxes contain two different kinds of referents. The referent 'Yoyo' of the first box is a name for an object. The referent '$*$' of the second box is a fixed symbol which does not denote a particular individual, but it denotes an individual which is not further specified. So the star '$*$' can be read as an existential quantifier. It is called *generic marker*, hence the second box is called GENERIC CONCEPT BOX. The ovals are (CONCEPTUAL) RELATIONS between the referents of the concept boxes which are linked to the oval.

Sowa uses various kinds of referents. For example, the following graphs have the meanings 'all men married a certain woman' and 'she is eating four bones'.

It is clear that these various kinds of referents go beyond the expressiveness of first order logic. In particular they are not adapted from EGs because they use universal quantifiers and numbered restriction. Moreover, even from the first example of this section (i.e., the graph with the meaning 'a cat is on a mat') a crucial difference between existential graphs and conceptual graphs is apparent. The lines of identity in EGs serve different purposes: first, they are used for existential quantification, second, they are used to connect arguments to relations, and third they are used to show identity between arguments. On the other hand in CGs these functions are separated. Existential quantification is expressed by generic concept boxes. The referents of concept boxes serve as arguments for relations, and the connection between the arguments and the relations is drawn by relation ovals. We need moreover to clarify how identity is expressed in CGs. For this, a new syntactical element is used. In [71], Sowa says "Two concepts that refer to the same individual are coreferent. [...] are used to show that they are coreferent, i.e. concepts are connected with a dotted line, called a *coreference link*." This can be illustrated using the graph of Fig. 7. This says that 'Mary is a person and there is a teacher who is the same as Mary', or 'Mary is a teacher'.

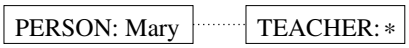


Fig. 7. conceptual graph for 'Mary is a teacher'

The second fundamental difference between EGs and CGs is the nesting of CGs. When graphs are nested, some CGs serve as referents in concept boxes of other CGs themselves. In this manner, some CGs make statements and assertions about other CGs, i.e., the system of CGs therefore offers the possibility of meta-level statements. Concept boxes whose referents are CGs are called CONTEXTS [11] . Peirce's sheet of assertion or the surface

[11] The term 'context' occurs in several meanings and implementations in logics, linguistics or artificial intelligence. However, as Sowa says in [69]: "The notion of context is indispensable for any theory of meaning, but no consensus has been reached about the formal treatment of context." A treatment of the various ideas of contexts is beyond the scope of this paper (refer to [69] or Chapter 5 of [76] for an

where a CG is scribed can equally be understood as a context, the OUTERMOST CONTEXT.

In Figure 8 we present a well-known example of a nested CG: It contains two contexts, namely the concept boxes of type PROPOSITION and SITUATION (which are common types of contexts). The graph can be read as follows: 'The person Tom believes a proposition, which is described by a graph itself'. The proposition says that 'the person Mary wants a situation, which again is described by a graph'. In this situation we have a concept box $\boxed{\top : *}$ that is connected with a coreference link to the concept box PERSON: Mary in the context above. So the situation is that 'Mary marries a sailor'. The formal understanding of the whole graph is now: 'The person Tom believes the proposition that the person Mary wants the situation that Mary marries a sailor'. In short: 'Tom believes that Mary wants the situation in which she marries a sailor', even more succinctly: 'Tom believes that Mary wants to marry a sailor'.

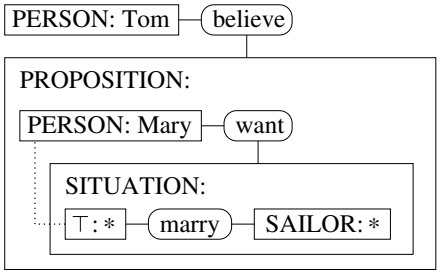


Fig. 8. A nested conceptual graph

Even negation is modeled with a specific context. As Sowa writes in [73]: "The EG negative contexts are a special case of the CG contexts. They are represented by a context of type Negation whose referent field contains a conceptual graph that states the proposition which is negated." Concept boxes of type NEGATION are introduced by Sowa as abbreviations for contexts of type proposition with an unary relation 'NEG' attached (see Sowa [71], "Negation (NEG) is one of the most common relations attached to contexts"), and Sowa often abbreviates these contexts by drawing a simple rectangle with the mathematical negation symbol ¬ (see Sowa [75]).

Two examples for CGs with negations are presented in Figure 9. The left graph states 'there are two suns, but it is not true that there is a thing which is identical to the two suns', i.e. 'there are at least two suns' (compare this to Figure 6). In the right graph, the device of two nested negation contexts corresponds to scrolls in EGs and can be understood as an implication. Hence, the meaning of the graph is 'if a farmer owns a donkey, then he beats it'.

Fig. 9. Conceptual graphs with negation

introduction and discussion of contexts in CGs. Also recommended are the works of McCarthy [51], and Barwise and Perry [7]).

There is presently no mathematical elaboration for the complete system of CGs [85,19]. Particularly troublesome is the handling of negation as specific contexts lead to several problems [19]. In the following, we will first discuss the most prominent fragment of CGs, namely simple conceptual graphs (SCGs). Then the system of conceptual graphs with cuts (CGwCs) is described. Both systems are elaborated in a formally precise manner.

4.2.1. *Simple Conceptual Graphs*

The most prominent and best investigated fragment of CGs is the system of simple conceptual graphs, SCGs. In this system, no contexts are allowed, so SCGs correspond to the conjunctive, positive and existential fragment of FOL. This fragment is decidable, nonetheless, a decision procedure to determine whether one SCG implies another is NP-complete [14].

There are two different approaches to a mathematical elaboration of SCGs. In both approaches, mathematical graphs are used to formalize SCGs. In the first, which follows the work of Chein and Mugnier [14,15], bipartite labeled graphs are used: both concept boxes and relation ovals are vertices in the graphs, and edges correspond to the lines connecting concept boxes and relation nodes. The other approach follows Wille [86] where concept boxes are modeled as vertices of a graph, whereas a relation oval, including all lines from it to concept boxes, is modeled as a directed hyper-edge. In this case, labeled directed multi-hypergraphs form the mathematical basis of SCGs. Nonetheless, both formalizations can easily be translated to the other [44].

Chein and Mugnier provide a translation of SCGs to FOL as semantics for SCGs[14]. Moreover, the authors provide a sound and complete syntactical entailment relation between SCGs based on graph homomorphisms, called PROJECTIONS. Briefly put, a SCG G implies a SCG H iff there is a graph homomorphism $f : H \to G$ which respects the underlying type hierarchy. In their first elaborations of projections [14], a further restriction on G was considered: namely it had to be in NORMAL FORM, i.e., no referent was allowed to occur in distinct concept boxes. In Chein and Mugnier [16], this restriction is dismissed by extended projection to so-called COREF-PROJECTIONS which are graph-homomorphisms that do not map vertices to vertices but rather maps sets of coreferent vertices to sets of coreferent vertices.

In Dau [18], one of the authors provides an extensional semantics for SCGs based on Wille's formal concept analysis ([36]). Moreover, a sound and complete calculus, consisting of fairly simple manipulations of the diagrams, is also provided. Thus, in both of the popular mathematical elaborations of SCGs, a calculus is defined that is equivalent to reasoning in a restricted form of FOL corresponding to the conjunctive, positive and existential fragment of FOL.

4.2.2. *Conceptual Graphs with Negation*

As mentioned, employing negation in CGs, by means of introducing a special context, yields problems. In order to overcome these difficulties, and to provide a mathematical elaboration of that fragment of CGs which corresponds to full FOL, one of the authors has extended in [19] the Wille-style formalization of SCGs by adding the cuts of Peirce's EGs. The resulting system is called the system of CONCEPTUAL GRAPHS WITH CUTS

(CGwCs). Similar to Peirce's EGs, the cuts are in CGwCs are drawn as ovals, but in order to distinguish them from the relation ovals, they are drawn in bold. Besides the addition of cuts, the coreference-links in CGwCs are replaced by relation ovals, labeled with the relation sign '='. For example, the CGs of Fig. 9 can now be represented with the additional of cuts as follows:

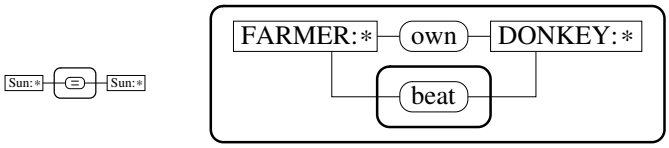


Fig. 10. Conceptual graphs with negations

CGwCs are equipped with an extensional semantics, a sound and complete diagrammatic calculus (based on the calculus for Peirce's EGs), and meaning-preserving translation to and from FOL. Particularly, they have the expressiveness of FOL, including equality and constants, but absent are function names.

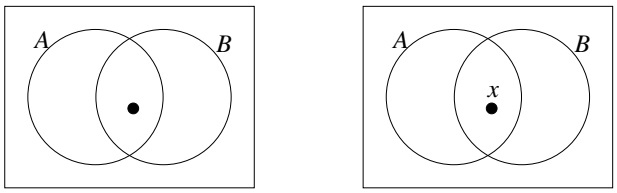
5. Spider or Constraint Diagrams for DL

At a first glance, there are some striking similarities between DL and the system of SDs and CDs. Most importantly, both correspond to decidable fragments of FOL where only unary predicates and binary relations are used. Both systems have sound and complete calculi which are implemented on tableaux-based algorithms. It is therefore worth investigating whether spider and constraint diagrams can be used as a diagrammatic representation for DL.

In our scrutiny of SDs and CDs, we first restrict ourselves to a discussion of SDs only. That is, on the DL side, we consider class constructors where only concepts, but no roles are used. This is an unreasonable restriction for DLs, but it serves well to identify some early differences between SDs and DL and test their compatability.

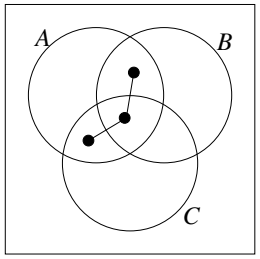
Firstly, an obvious difference between DL and SDs is that DLs are used to express concepts, which can be translated into FOL-formulas with one free variable, whereas SDs are translated into sentences, i.e. FOL-formulas without free variables. So we have to extend the syntax and semantics of SDs slightly to accommodate this. This could be done by labeling spiders with variables. Let us consider a conjunction of two concepts A and B, i.e. $A \sqcap B$, as a very simple example for a class constructor. With a SD, we can express the formula $\exists x.(A(x) \wedge B(x))$. The variable x will be replaced by a spider. By labeling the corresponding spider with x, we provide a simple SD-like representation for the formula $A(x) \wedge B(x)$, where x is now free.

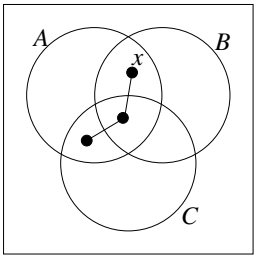
$A \sqcap B$



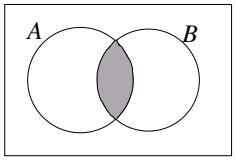
A more complex example is the concept description $A \sqcap (B \sqcup C)$. Again, we can express $\exists x.(A(x) \wedge (B(x) \vee C(x))$ with a SD. Each formula in propositional logic can be converted into its disjunctive normal form, where in each of the conjuncts, all concepts of the formula appear as literals. For example, instead of $A \sqcap (B \sqcup C)$, we consider the equivalent formula $(A \sqcap B \sqcap \neg C) \sqcup (A \sqcap B \sqcap C) \sqcup (A \neg \sqcap B \sqcap C)$. As each conjunct corresponds to one minimal region in the diagram, we therefore have the opportunity to express each concept by means of a single spider in SDs. Similar to the last example, we provide the concept description, the SD and the corresponding DL-SD, including a free variable x.

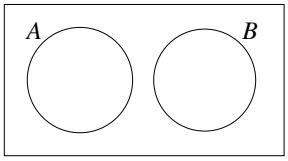
$A \sqcap (B \sqcup C)$





The purpose of SDs is to express *relationships* between predicates, not to express the *construction* of new predicates from those given. For example, we can express the disjointness of two predicates A and B by either using a shading in the corresponding Venn-diagram, or by using the features of Euler circles. Both possibilities are presented below.

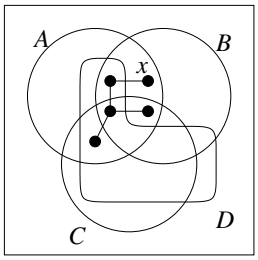




Atomar DL-concepts are represented in SDs by contours. The Euler-features of SDs can be used to represent entailment or disjointness of the atomar DL-concepts represented by contours. A definition of a *single* DL-concept does not express such set-theoretical relationships. Thus if we model a single DL-concept by means of SDs, we cannot draw benefit from the Euler-features of SDs: We need only the Venn-diagram part of SDs without shadings. On the other hand, it is possible to express relationships between sets in DL-knowledge-bases: The axiom $\bot \equiv A \sqcap B$ expresses that A and B are disjoint, and $A \sqsubseteq B$ expresses that A is a subset of B. That is, when we attempt to diagrammatically represent a single DL-concept of a set of DL-concepts and DL-axioms, some of the axioms can incorporated in the diagram, expressed by means of Euler-features or shadings. An example for this is given below.

In any event, the contours of *mutually independent* atomar DL-concepts give rise to a Venn-diagram. It is possible to draw Venn-diagrams for an arbitrary number of predicates, but if more than three predicates are used, the diagrams become difficult to read. Below is an example of a concept definition where four concepts are involved, and the corresponding SD-diagram. A more complex example will be given below.

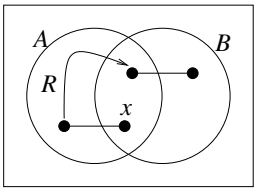
$A \sqcap (B \sqcup (C \sqcap D))$



Besides the disadvantage that Venn-diagrams with more than three predicates are hard to read, SDs present a possible diagrammatic representation for DL when no roles are used. In the following, roles are taken into account.

Let us consider two very simple concept description with a role involved: $A \sqcap \exists R.B$ and $A \sqcap \forall R.B$. For the first concept, it seems self-suggesting to use an existential arrow to express the $\exists R.B$-part of this description. Similarly, for the second concept, using a universal spider seems appropiate. Therefore, the following CDs seem to be the canonical CD-style representation of the concept descriptions:

$A \sqcap \exists R.B$



$A \sqcap \forall R.B$

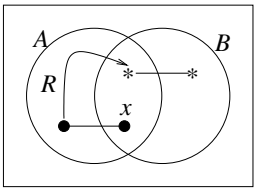


Fig. 11. A first, failed attempt to express DL-restrictions

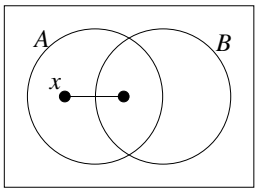
However the semantics of these CDs with a free variable and the intended DL-semantics of the concept description differ.

	Left Diagram	**Right Diagram**
SD-semantics	$\{x \mid x \in A \wedge \exists b \in B : xR^{\mathcal{I}} = \{b\}\}$	$\{x \mid x \in A \wedge \forall b \in B : xR^{\mathcal{I}} = \{b\}\}$
DL-semantics	$\{x \mid x \in A \wedge \exists b \in xR^{\mathcal{I}} : x \in B\}$	$\{x \mid x \in A \wedge \forall b \in xR^{\mathcal{I}} : x \in B\}$
	$= \{x \mid x \in A \wedge xR^{\mathcal{I}} \cap B \neq \emptyset\}$	$= \{x \mid x \in A \wedge xR^{\mathcal{I}} \subseteq B\}$

We see that arrows having existential spiders as targets are not suited to express the 'exists restriction' constructor of DLs, and similarly, arrows having universal spiders as targets are not suited to express the 'value restriction' constructor of DLs. Moreover, note that the right diagram is not a well-formed diagram according to CDs as defined by Stapleton [77] – because the universal spider inhabits not a single zone, and the universal spider is not the source (but the target) of an arrow. On the other hand, both diagrams are well-formed CDs according to the full CD system developed by Fish et al. [29] (but recall that Fish et al. [29] do not provide a calculus for the full system).

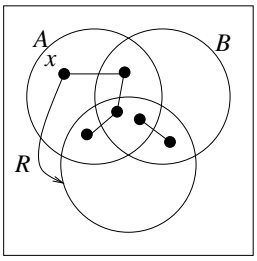
In the CDs of Fish et al. [29], we have arbitrary many derived contours. Derived contours are basically unlabeled contours which are targets of arrows. An arrow represents a relation, so a target of arrows denotes the range of this relation. Thus derived contours provide a means to express how the range of a relation is set-theoretically related to other sets. In fact, instead of spiders, derived contours are the right entities to express

the 'exists restriction' and 'value restriction' constructors of DLs. Let us consider the two DL-concepts again. The concept A is of course expressed by



For the first concept, i.e. $A \sqcap \exists R.B$, we have to add a derived contour to this diagram, being the target of an arrow labeled with R. We do not know any subset-relationships between $xR^{\mathcal{I}}$ and A or B, so the diagram we obtain is similar to a Venn-diagram with three sets. However we know that the intersection of $xR^{\mathcal{I}}$ and B is not empty, so we add an additional existential spider to the region that depicts $xR^{\mathcal{I}} \cap B$. For the second concept, i.e. $A \sqcap \forall R.B$, we have again to add a derived contour to this diagram. Now this contour has to express that $xR^{\mathcal{I}} \subseteq B$ holds. We can use the Euler-features of CDs to express this. In Figure 12 the two CDs that express the two DL-concepts are depicted.

$A \sqcap \exists R.B$



$A \sqcap \forall R.B$

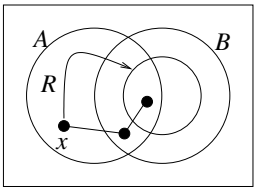


Fig. 12. Expressing DL-restrictions with constraint diagrams

There remains a slight flaw in the CD with the exists restriction: in the system of CDs distinct spiders denote distinct objects so this diagram is not an appropiate translation of the exists restriction with CDs. In [37], the notation of *strands* has been introduced: In the diagrams, strands are wavy lines between different spiders, and they are used to indicate that these spiders do not necessarily denote different objects. So we could add a strand to our diagram. Despite such repair, strands are not considered in [29], where the syntax and semantics of full CDs is developed. Another possibility is to slightly alter the semantics of [29] by dismissing the condition that different spiders denote different objects. Let us assume this approach in the following.

With the discussion so far, we can now provide a CD that corresponds to the DL concept HappyMan of equation (1). First of all, this concept contains five atomic concepts, namely Man, Female, Male, Rich, and Happy. The definition of HappyMan does not provide information on how these concepts are related, so a CD which corresponds to equation (1) is based on a Venn-Diagram with five contours. The diagram to the left of Figure 13 is such a diagram. As mentioned earlier, Venn-diagrams with more than three contours become difficult to read. To make the example easier to comprehend, let us assume that besides the definition of HappyMan, we have moreover DL-axioms $\bot \equiv \text{Male} \sqcap \text{Female}$ and $\text{Man} \sqsubseteq \text{Male}$. Then we can start with a more simpler Euler-diagram, as depicted right in Figure 13.

The CD corresponding to HappyMan will be based on the right diagram. To make this CD more comprehensible, it will be build inductively. The next two diagrams in Figure 14 depict the DL concepts Man and Man $\sqcap \forall$hasChild.(Rich $\sqcup$ Happy), respectively.

Note that in the second step, we had to add a derived contour, so we split some spiders as well. This contour depicts in an iconic, Euler-style way, that we have $xR^{\mathcal{I}} \subseteq B$ for each x in the concept extension. On the other hand, we already have six contours, which are not easy to comprehend. In the next two steps, which are depicted in Figure 15, we add the conjuncts $\exists$hasChild.Female and $\exists$hasChild.Male to the concept definition.

In each step, another derived contour is added to the diagram. In contrast to the second step, these new contours depict sets that have a set-theoretical relationship to the other sets: we have to add the contours in a way that they separate each minimal region into two parts. This yields a diagram which unfortunately lacks readability.

We haven't disussed a complete translation of DL-concepts into CDs, but it seems there is some prospect that a general translation can be provided. However, we have to deal with the following problems: first, we have slightly altered the semantics of CDs by dismissing the condition that different spiders denote different objects. Second, neither for the system of CDs with the original semantics, nor for our system with the slightly altered semantics, do a sound and complete calculus exists. Finally, as the previous example shows, each atomic concept and each (value or exists) restriction in DL-concepts needs a contour in the corresponding CD which quickly yields CDs that, by any reasonable measure, are less readable than the orginating DL-concepts.

6. Conceptual or Existential Graphs for DL

Both CGs and DLs have some common background: they are both developed and intended as knowledge representation systems to include reasoning facilities, they were developed at nearly the same time and they have an important common ancestor, namely the semantic networks of AI (and Minski's frame systems). In fact, in 1979, the well-known system Kl-One was developed on the basis of informal use of semantic networks in AI (see [10,11]). Kl-One was itself a diagrammatic formalism and in 1987 a well-defined, extensional, Tarski-style semantics was developed for it. Thus, in contrast to semantic networks, the diagrammatic entities in Kl-One were equipped with precise meaning. This was the starting point for the first DL [49].

Thanks to this common background, there had been several attempts to identify how and where DL and CGs were related: namely specifying where DLs are related to corresponding fragments of CGs. Coupey and Faron [17] focus on SCGs where a rather restricted DL, namely a restricted version of $\mathcal{ALEOI}$, is considered. In this case only the following class constructors are considered: inverse roles (i.e., R^-), a restricted version of conjunction ($C_1 \sqcap C_2$), and existential restriction ($\exists R.C$). It is argued that nominals can easily be added. More importantly, due to the lack of negation in SCGs, the authors cannot express the negation of concepts $\neg C$, disjunction $C_1 \sqcup C_2$, or value restriction $\forall R.C$. Baader et al.[3] extend the results of Coupey and Faron by allowing the existential intersection of concept descriptions, intersection of roles and unary 'one-of' concepts, i.e. they determine a fragment of conceptual graphs that corresponds to the DL $\mathcal{ELIRO}_1$. Again, due to the lack of negation, this DL is not propositionally closed.

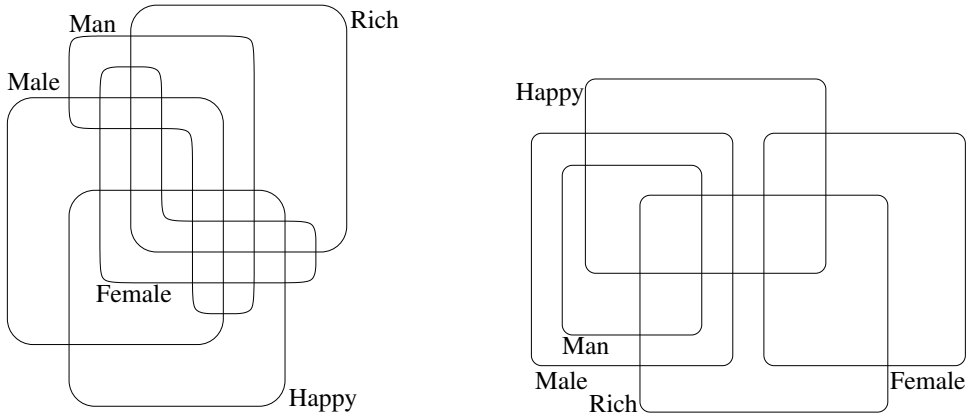


Fig. 13. A Venn Diagram and an Euler Diagram for the Atomar Concepts of HappyMan

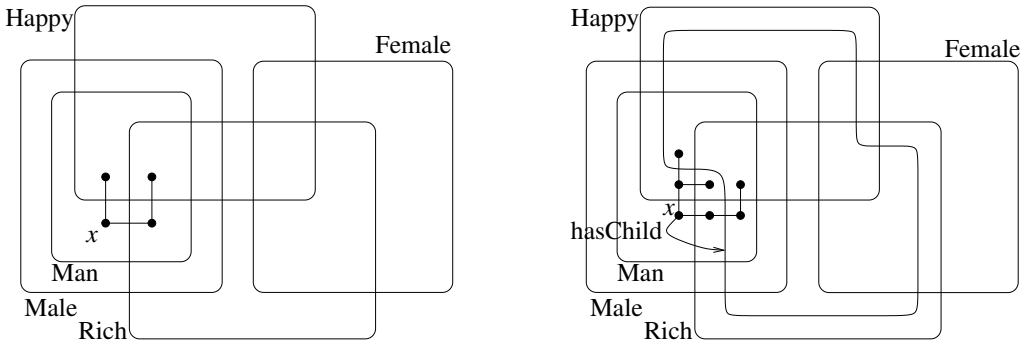


Fig. 14. CDs for Man and Man ⊓ ∀hasChild.(Rich ⊔ Happy)

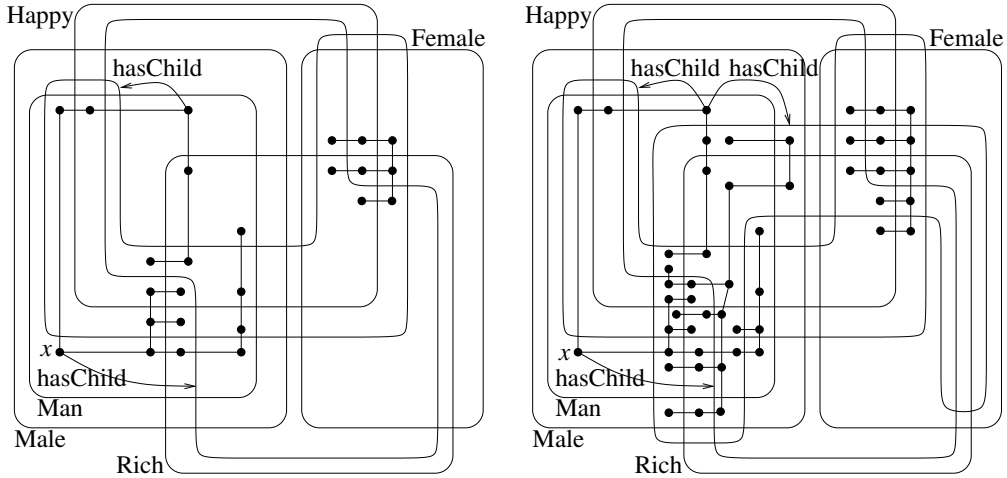


Fig. 15. CDs for Man ⊓ ∃hasChild.Female ⊓ ∀hasChild.(Rich ⊔ Happy) and HappyMan

As well similarities between DL and CGs, there are important differences. Firstly, similar to SDs and CGs, all graphs (i.e., both CGs and EGs) correspond to closed FOL formulas. Next, in the graphs, relations of arbitrary arities are allowed. There is no correspondence to the type-hierarchy of CGs in DL. Finally, the syntactical possibilities of the graphs, including identity, graphs that contain circles, graphs that are not connected etc, allow graphs to be constructed that do not have counterparts in DL. However, our intention with this paper is not to explore direct correspondences between graphs and DL, but rather using the graphs as a diagrammatic version of DL. Therefore we are free to dismiss the features of CGs not needed (for example, it is reasonable to consider only graphs where binary relations are allowed and where we have only a flat type hierarchy). On the other hand, the lack of free variables in the graphs is a *absent* feature which we need for DLs. As such, a first task is to discuss how free variables can be added to graphs.

Adding free variables to graphs is relatively straight forward. For existential graphs, Burch [13] focuses on an algebraic elaboration of EGs. Inspired by Burch, we find other approaches in the works of Pollandt [59,60], Wille [87], Hereth-Correia and Pöschel [40,41], and Hereth-Correia and Dau [27]. [12] The resulting graphs are usually called RELATION GRAPHS (RG), as they describe relations instead of propositions [13].

The diagrammatic rendering of free variables is via numbered question markers. In Figure 16 a relation graph, and the corresponding CGwC with free variables, are depicted. Both graphs have two query markers ?1, ?2, and thus they describe the relation of all pairs of objects (o_1, o_2) such that, if ?1 and ?2 are replaced by o_1 and o_2 respectively we obtain a valid graph. So the graphs describe the binary relation `is_stepmother_of`.

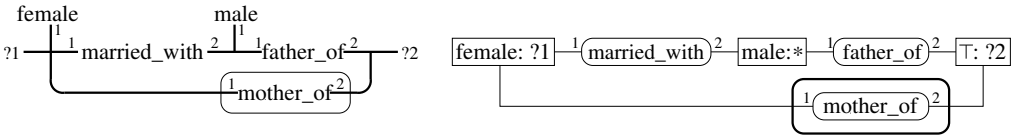


Fig. 16. An relation graph and the CGwCs with free variables

In the following, we focus on the logic $\mathcal{ALC}$, i.e. the DL which encompasses inverse roles, conjunction, disjunction, negation, value restriction and exists restriction. The main reason for this choice is that $\mathcal{ALC}$ is the smallest *propositionally closed* DL $\mathcal{ALC}$. As we go for a propositionally closed DL, we do not consider SCGs, but only RGs and CGwCs.

Considering other DL-constructs is the subject of further research. As briefly discussed in the outlook, first unary 'one-of' concepts, inverse roles, and the intersection of roles will be targeted, as is it likely that adding these constructs will not be too problematic. Once this is done, we obtain a strict extension of the the results of [17,3].

We consider again the DL-concept of equation (1) as an example. First, it should be noted that in the graph-based systems of RGs and CGwCs we have no genuine opportunity to express disjunction or universal quantification. However this does not present any difficulty since both can be alternatively expressed by means of conjunction, negation, and existential quantification (the basic operators in RGs and CGwCs). On the DL side,

[12] Another source is the appendix of [23], where the calculus of EGs is extended to a a sound and complete calculus for RGs.
[13] For CGwCs, free variables have been added to them in [20].

$C_1 \sqcup C_2$ can be replaced by $\neg(\neg C_1 \sqcap \neg C_2)$, and $\forall R.C$ can be replaced by $\neg\exists R.\neg C$. Therefore, we consider the equivalent formula,

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

and provide "naive" translations of it to CGwCs and an RG.

Baader et al. [3] consider SCGs that have one distinguished node termed ROOT. We adopt this idea by providing CGwCs where a distinguished concept box $\boxed{\top : ?}$, placed on the sheet of assertion, is used to describe the concept description (note that as concept descriptions of DL correspond to FOL-formulas with only one free variable, there is no need to use *numbered* question markers).

There are two possibilities to transform this concept description into a CGwC. On the one hand, we can translate the atomic concepts of DL to *types* in CGwCs, i.e. they appear in concept boxes. On the other hand, we can both translate atomic concepts and roles to relations in CGwCs, i.e. roles appear in relation ovals. The CGwCs corresponding to the given concept description are depicted in Figure 17.

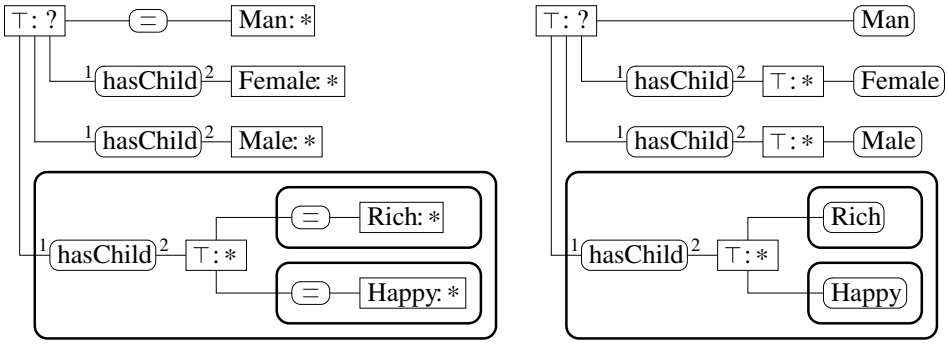


Fig. 17. Concept descriptions as CGwCs, left with atomic classes as types, right with atomic classes as unary relations

Note that for the left graph of Figure 17, we need a generic marker for each concept of the concept description, as well as for each subformula $\exists R.C$ or $\forall R.C$, where C is not an atomic concept. In the graph to the right, we need a generic marker for each role of the concept description, which now appears in concept boxes $\boxed{\top : *}$. In the graph left, we need additional identity links [14] .

Next in Figure 18 an RG for the concept description is depicted. In EGs and RGs both existential quantification and the identity of objects is modeled with ligatures. We can rid ourselves of identity links in CGwCs (see left example in Figure 17) and concept boxes $\boxed{\top : *}$ (see right example in Figure 17) .
These graphs give rise to the following conclusions: first, it is possible to provide fragments of CGwCs or RGs which correspond to $\mathcal{ALC}$. Second, as both identity links and concept boxes $\boxed{\top : *}$ are encompassed by lines of identity in RGs, the RG is easier to comprehend than the two CGwCs.

Compared to SDs and CDs we see a more natural translation from $\mathcal{ALC}$-concepts to existential graphs than to CDs. Moreover, for RGs, there exists a sound and complete

[14] A short discussion on the need for identity links in translating DLs to CGs is provided in [17].

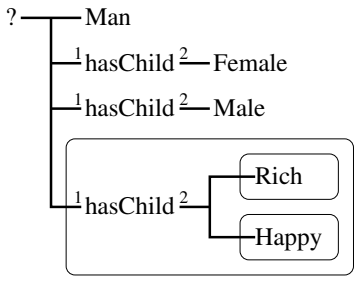


Fig. 18. Concept description as EG

calculus, whereas such a calculus is absent for CDs. Although CDs might still be elaborated as a diagrammatic reasoning system for $\mathcal{ALC}$the discussion of the last sections leads to the conlusion that RGs seem the most promising system to formalize the DL $\mathcal{ALC}$ as a diagrammatic reasoning system. For this reason, in the next sections we will determine a fragment of RGs corresponding to $\mathcal{ALC}$.

7. A Tree-based Formalization of The Description Logic $\mathcal{ALC}$

In this section, the syntax and semantics for the DL $\mathcal{ALC}$ is provided. First of all, as mentioned earlier in Section 2, the DL $\mathcal{ALC}$ has conjunction, negation and value restriction as constructors. As the RGs of Peirce have conjunction, negation and existential quantification as constituents, we can consider $\mathcal{ALC}$ made up from conjunction, negation and existential restriction (instead of value restriction).

Let $\mathrm{RG}^{\mathcal{ALC}}$ denote the class of RGs that will be used to formalize $\mathcal{ALC}$. The graphs of $\mathrm{RG}^{\mathcal{ALC}}$ will be called $\mathcal{ALC}$-GRAPHS. The class $\mathrm{RG}^{\mathcal{ALC}}$ is a subclass of all RGs and the graphs of $\mathrm{RG}^{\mathcal{ALC}}$ exhibit significant additional constraints. First, for a graph in $\mathrm{RG}^{\mathcal{ALC}}$, only unary predicates and binary relations are used. The graph has exactly one pending edge labeled with '?' and this is placed on the sheet of assertion. Moreover, each cut is crossed exactly once by a heavily drawn line (particularly, we must have no empty cuts). Most importantly, the underlying graph structure without the cuts is a *tree*. The existing formalizations of the full system of RGs have to capture more complex structures. For the purpose of this paper, these formalizations are technically overloaded, thus we provide a new formalization by means of *labeled trees*. Using trees is, on the one hand, close to the usual approach to inductively define the formulas of $\mathcal{ALC}$. On the other hand, we can easily provide for each tree the corresponding RG-diagrams (which will be done in the next section), and labeled trees are better suited than formulas when we finally employ a Peirce-style calulus for $\mathcal{ALC}$.

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 labeled trees are implicitly identified.

Next, we introduce operations to inductively construct labelleled trees. These operations

will be used to define the syntax of $\mathcal{ALC}$. 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 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 $\nu := \nu' \cup \{(v_1, l_1), \ldots, (v_n, l_n)\}$., i.e,, $\mathbf{T}$ 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 labelled trees and $\mathbf{S} := (S, \geq_s, \nu_s)$ a subtree of $\mathbf{T}_1$. Then $\mathbf{T} := \mathbf{T}_1[\,\mathbf{T}_2 \,/\, \mathbf{S}\,]$ or $\mathbf{T}_1 \left[\frac{\mathbf{S}}{\mathbf{T}_2}\right]$ denotes the labelled tree obtained from $\mathbf{T}_1$ when $\mathbf{S}$ is substituted by $\mathbf{T}_2$. Formally, we define $\mathbf{T} := (T, \geq, \nu)$ where we set $T := (T_1 - S) \cup T_2$, $\geq := \geq_1 \big|_{T_1 - S} \cup \geq_2 \cup \{(w_1, w_2) \mid w_1 > v,\, w_1 \in T_1 - S,\, w_2 \in T_2\}$, 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.

Before the syntax and semantics of $\mathcal{ALC}$based on labeled trees is introduced, let us more clearly define the vocabulary we will use. As we focus on $\mathcal{ALC}$ we only need to consider atomic concepts and atomic roles but no individuals. Therefore a vocabulary is defined to be a pair $(\mathcal{A}, \mathcal{R})$ where $\mathcal{A}$ is a set of (ATOMIC) CONCEPTS and $\mathcal{R}$ is a set (ATOMIC) ROLES. We also assume to have a top-concept $\top \in \mathcal{A}$. In an interpretation $(\Delta^{\mathcal{I}}, \mathcal{I})$, the interpretation function $\mathcal{I}$ assigns to every atomic concept $A \in \mathcal{A}$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role $R \in \mathcal{R}$ a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and we have $\mathcal{I}(\top) = \Delta^{\mathcal{I}}$.

Now the tree-based syntax and semantics of $\mathcal{ALC}$is defined. To distinguish this syntax from the usual notation of $\mathcal{ALC}$ by means of formulas, this system is called $\mathcal{ALC}^{Tree}$.

Definition 7.1 (Syntax and Semantics of $\mathcal{ALC}^{Tree}$) *Let $(\mathcal{A}, \mathcal{R})$ be a vocabulary, where $\mathcal{A}$ is a set of atomic concept names and where $\mathcal{R}$ is a set of atomic role names. We assume we have a further atomic name $\top$. Moreover, 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 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 Tree for $\top$: *The labeled tree $\top := (v, \top)$ is in $\mathcal{ALC}^{Tree}$. We set $\mathcal{I}(\mathbf{T}) = \Delta^{\mathcal{I}}$.*

Atomic Tree for A: *If A is an atomic concept, then the tree $\mathbf{T}_A := (v, A)$ is in $\mathcal{ALC}^{Tree}$. We set $\mathcal{I}(\mathbf{T}) = A^{\mathcal{I}}$.*

Negation: *Let $\mathbf{T} \in \mathcal{ALC}^{Tree}$, let v be a fresh vertex. Then $\mathbf{T}' := \neg\mathbf{T} := \neg\mathbf{T}$ 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 $\mathbf{T} := (\mathbf{T}_1 \sqcap \mathbf{T}_2) := \sqcap(\mathbf{T}_1, \mathbf{T}_2)$ 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* $\mathbf{T}' := R\mathbf{T} := R\mathbf{T}$ *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 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 to be* 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 $\mathbf{T}_1 \sqcap \ldots \sqcap \mathbf{T}_n := \top$.

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

Definition 7.2 (Semantical Entailment) *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*

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

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* $\mathbf{T}' \models \mathbf{T}$*.*

8. A Diagrammatic Representation for $\mathcal{ALC}^{Tree}$

In Section 4.1, we briefly discussed that C.S. Peirce distinguished between *graphs* and *graph replicas*. As it is is more generally argued in [43,21], any formalization of logic by means of diagrams has to distinguish these two levels, usually called *types* or *abstract syntax* (which correspond to Peirce's graphs), and *tokens* or *concrete syntax* (corresponding to (Peirce's replicas). The $\mathcal{ALC}$-trees, as they have been defined in the last section, are obviously the abstract syntax. In this section we explain how each $\mathcal{ALC}$-tree can be diagrammatically represented in the style of Peirce's graphs, i.e., the token-level is introduced.

First, we will use the letter G, sometimes with indices, to denote the graphs (i.e., diagrams) of $\mathrm{RG}^{\mathcal{ALC}}$. As stated in the last section we will only consider RGs that have a single query marker placed on the sheet of assertion, i.e. there will be one heavily drawn line on the sheet of assertion with a "?" attached to it. We will call this the PENDING EDGE of the graph. In order to emphasize this pending edge of a graph G we will often represent it in the following manner: ?—G.

To each $\mathcal{ALC}$-tree $\mathbf{T}$, we now assign corresponding relation graph diagrams $\Psi(\mathbf{T})$ with one 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:

$\Psi(\top) := ?$—

$\Psi(A) := ?$—A

$\Psi(\mathbf{T}_1 \sqcap \mathbf{T}_2) := ?$—┬$G_1$ / └G_2

$\Psi(\neg\mathbf{T}) := ?$—(—$G$—)

$\Psi(R\mathbf{T}) := ?$—R—G

In the following, the term '$\mathcal{ALC}$-graph' will be used to denote the diagrammatic, Peirce-style representation of $\mathcal{ALC}$-trees, i.e. with '$\mathcal{ALC}$-tree' we refer to the type-level and with '$\mathcal{ALC}$-graph' we refer to the token-level.

In this definition we implicitly used diagrammatic conventions to denote some operations on RGs. For example, in the 3rd step (corresponding to conjunction), the pending edges of ?——G_1 and ?——G_2 have been joined in a newly generated branching point; and in the 4th step, the pending edge of ?——G is extended outwards through the newly generated cut. According to Peirce's convention *No. Zero*, we may vary certain features of the diagram, like the shape of the cut or the arrangement of the entities on the plane. Considering our $\mathcal{ALC}$-formula example given in the introduction, the corresponding $\mathcal{ALC}$-tree (the type), and two corresponding Peircean diagrams (two tokens), are provided in Figure 19. Finally, in the full system of RGs, we need to label the edges adjacent to a relation name R with numbers indicating the order of the arguments of R. As the RGs of $\mathrm{RG}^{\mathcal{ALC}}$ have a tree-like structure, we can ommit this labeling.

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

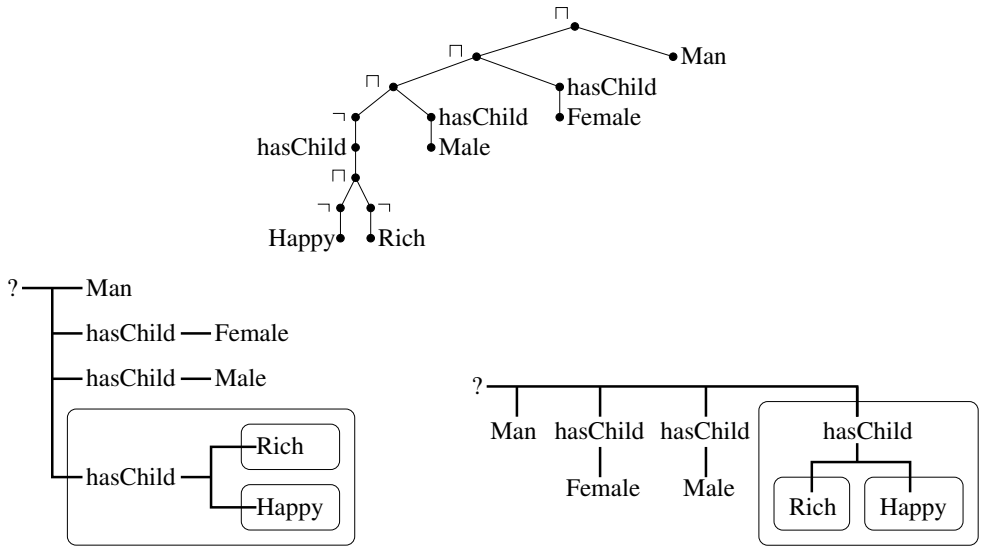


Fig. 19. An $\mathcal{ALC}$-formula, the corresponding $\mathcal{ALC}$-tree, and two diagrams of Peirce graphs.

We have to this point provided a translation of $\mathcal{ALC}$-trees into diagrams of Peirce's graphs. If we have on the other hand a graph ?——G, the first diagrammatic entity we reach from the question mark is either (if non-empty) an atomic concept, an atomic role, a branching point, or a cut, and from this entity we see the last step in the construction

of the $\mathcal{ALC}$-tree that corresponds to the diagram [15]. Therefore, for any given diagram of $\mathrm{RG}^{\mathcal{ALC}}$ we can reconstruct the corresponding $\mathcal{ALC}$-tree **with ambiguity**.

We have established a one-to-one correspondence between $\mathcal{ALC}$-trees and their diagrammatic representations in the form of graphs of $\mathrm{RG}^{\mathcal{ALC}}$. The rules of the forthcoming calculus can be best understood to be carried out on the diagrammatic representations. Further, the ongoing formal proofs with $\mathcal{ALC}$-trees will be depicted this way.

9. The Calculus for $\mathcal{ALC}^{Tree}$

C.S. Peirce provided a set of five rules for the system of existential graphs termed *erasure*, *insertion*, *iteration*, *deiteration*, *double cut*. It is possible to elaborate Peirce's graphical calculus, including a proof of its soundness and completeness, in a mathematical precise manner, and to extend the calculus for the system of RGs, but this goes beyond the scope of this paper.

$\mathrm{RG}^{\mathcal{ALC}}$ is a fragment of the full system of RGs. As one would expect, the rules for RGs are also sound rules for $\mathrm{RG}^{\mathcal{ALC}}$. However it is not clear whether these rules are also complete. For two graphs $G_1, G_2 \in \mathrm{RG}^{\mathcal{ALC}}$ with $G_1 \models G_2$, we have a proof for $G_1 \vdash G_2$ within the full system of RGs, 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. Despite this it might happen that we do not have a proof that consists only of graphs of $\mathrm{RG}^{\mathcal{ALC}}$. In fact, in the calculus we provide, we require more rules. Besides some trivial rules, like rules which 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* (see below) are the most important examples of 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, examples and diagrams that illustrate the operation of the rules are provide within their definitions. For each rule name, we provide an abbreviation that will subsequently be used in the proofs.

Definition 9.1 *The calculus for $\mathcal{ALC}$-Trees over a given vocabulary $(\mathcal{A}, \mathcal{R})$ consists of the following rules:*

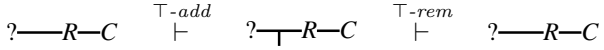
Addition and Removal of $\top$ ($\top$-add. and $\top$-rem.): *Let $\mathbf{T} := (T, \geq, \nu)$ be an $\mathcal{ALC}$-tree, let $\mathbf{S} \subseteq \mathbf{T}$ be a subtree.*

$$\textit{For} \quad \mathbf{T}' := \mathbf{T}\left[\frac{\mathbf{S}}{\sqcap(\mathbf{S}, \top))}\right] \quad \textit{we set} \quad \mathbf{T} \dashv\vdash \mathbf{T}' \,.$$

($\mathbf{T} \dashv\vdash \mathbf{T}'$ 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 $\mathcal{ALC}$-graphs,*

[15] In fact each $\mathcal{ALC}$-tree, in turn each graph of $\mathrm{RG}^{\mathcal{ALC}}$, has a unique derivational history. This is a common feature of linear and symbolic notations of logic. However, for inductively defined diagrammatic versions of logic, this feature is often absent. Having a unique derivational history provides a unique reading of a formula but diagrams often provide multiple readings, namely a given diagram can be read in different although semantically equivalent ways ([67]). The point is that in diagrammatic systems it is not necessarily desirable to provide a unique derivational history, see [21,67] for a deeper discussion on this issue.

this rule corresponds to adding a branch to a heavily drawn line, or removing it. A simple example is given below.

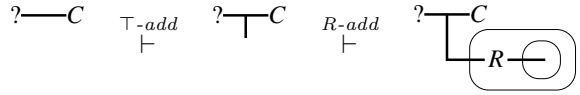


These rules are "technical 'helper' rules that will be often combined with other rules that add or remove subtrees. Examples will be given below.

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.*

$$\textit{For} \quad \mathbf{T}\left[\frac{\top}{\neg R\neg\top}\right] \quad \textit{we set} \quad \mathbf{T} \dashv\vdash \mathbf{T}' \,.$$

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 ALC-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 let* $\mathbf{S}_1 \sqcap (\mathbf{S}_2 \sqcap \mathbf{S}_3) \subseteq \mathbf{T}$ *be a subtree.*

$$\textit{For} \quad \mathbf{T}' := \mathbf{T}\left[\frac{\mathbf{S}_1 \sqcap (\mathbf{S}_2 \sqcap \mathbf{S}_3)}{(\mathbf{S}_1 \sqcap \mathbf{S}_2) \sqcap \mathbf{S}_3}\right] \quad \textit{we set} \quad \mathbf{T} \dashv\vdash \mathbf{T}' \,.$$

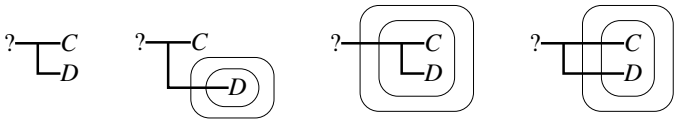
We say that $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *resp.* $\mathbf{T}$ *is derived from* $\mathbf{T}'$ *by* USING THE ASSOCIATIVITY OF CONJUNCTION.

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.*

$$\textit{For} \quad \mathbf{T}' := \mathbf{T}\left[\frac{\mathbf{S}}{\neg\neg\mathbf{S}}\right] \quad \textit{we set} \quad \mathbf{T} \dashv\vdash \mathbf{T}' \,.$$

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.

To see an example, consider the four $\mathcal{ALC}$*-graphs below. The second and third graph can be derived from the first by adding a double negation. The opposite direction can be applied symmetrically. 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 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* $\mathbf{S} \subseteq \mathbf{T}$*.*

$$\textit{For} \quad \mathbf{T}' := \mathbf{T}[\,\top / \mathbf{S}\,] \quad \textit{we set} \quad \mathbf{T} \vdash \mathbf{T}' \,.$$

We say that $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* ERASING $\mathbf{S}$ FROM EVEN*. In the opposite direction, let* $\mathbf{T} =$ *be an* $\mathcal{ALC}$*-tree with an negatively enclosed subtree* $\top \subseteq \mathbf{T}$*. Let* $\mathbf{S} \in \mathcal{ALC}^{Tree}$*.*

$$\textit{For} \quad \mathbf{T}' := \mathbf{T}[\,\mathbf{S} / \top\,] \quad \textit{we set} \quad \mathbf{T} \vdash \mathbf{T}' \,.$$

We say that $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* INSERTING $\mathbf{S}$ INTO ODD.

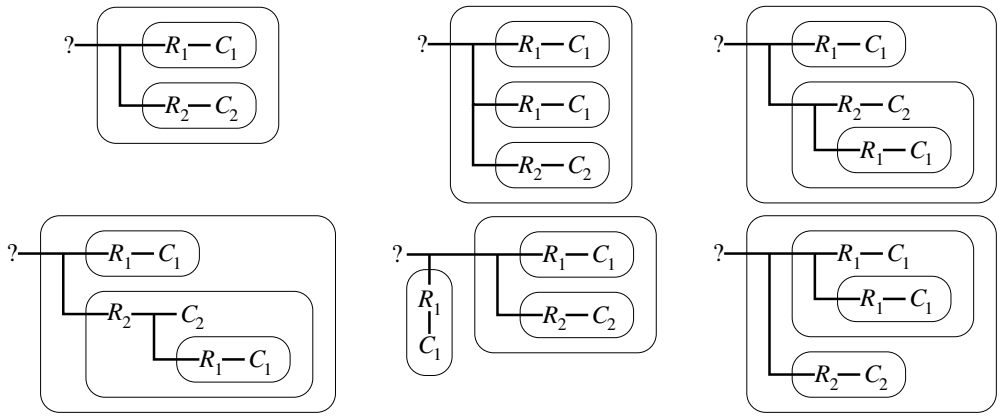
This is another set of rules that often go together with the addition and removal of $\top$. *Examples will be given later.*

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 root 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* $\nu(w) = \sqcap$.

$$\textit{For} \quad \mathbf{T}' := \mathbf{T}\left[\frac{\top}{\mathbf{S}}\right] \quad \textit{we set} \quad \mathbf{T} \dashv\vdash \mathbf{T}' \,.$$

(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.) We say that* $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* ITERATING $\mathbf{S}$ *and* $\mathbf{T}$ *is derived from* $\mathbf{T}'$ *by* DEITERATING $\mathbf{S}$.

Interation 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* $\mathcal{ALC}$*-graphs. The second and the third graph can be derived from the first graph by iterating the subgraph* ?—(—R_1—C_1) *(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* $v < t$. *Finally, the sixth graph violates the condition* $v \notin S$.



Iteration of Roles into even, Deiteration of Roles from odd (R-it., 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* $\mathbf{S}_b := R \sqcap (\mathbf{S}_1, \neg\mathbf{S}_2)$. *Moreover, let* $\mathbf{S}_a \subseteq \mathbf{T}$ *be a positively enclosed subtree.*

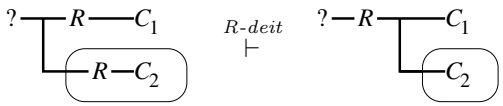
$$\textit{For} \quad \mathbf{T}' := \mathbf{T}[\,\mathbf{S}_b \,/\, \mathbf{S}_a\,] \quad \textit{we set} \quad \mathbf{T} \vdash \mathbf{T}' \,,$$

and we say that $\mathbf{T}'$ *is derived from* $\mathbf{T}$ *by* DEITERATING THE ROLE R FROM ODD. *Vice versa, let* $\mathbf{S}_b \subseteq \mathbf{T}$ *be a negatively enclosed subtree.*

$$\textit{For} \quad \mathbf{T}' := \mathbf{T}[\,\mathbf{S}_a \,/\, \mathbf{S}_b\,] \quad \textit{we set} \quad \mathbf{T} \vdash \mathbf{T}' \,,$$

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 9.2 *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 presented, we first present an example of a proof, using the Peirce-style diagrams, and then derive some useful meta-rules. The example and the meta-rules will give some insights in how the calculus works.

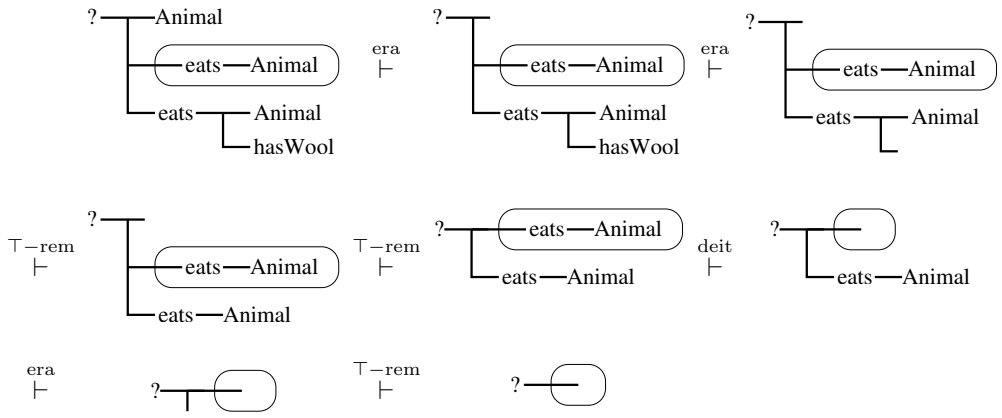
A popular toy example for reasoning within $\mathcal{ALC}$ is the mad cow ontology. Consider the following $\mathcal{ALC}$-definitions:

$$Cow \equiv Animal \sqcap Vegetarian \qquad Sheep \equiv Animal \sqcap hasWool$$
$$Vegetarian \equiv \forall eats.\neg Animal \qquad MadCow \equiv Cow \sqcap \exists eats.Sheep$$

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 investigating whether this concept is satisfiable, i.e., whether there exists as least one interpretation where this concept is 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 $\mathcal{ALC}$-graph for the given concept and derived the absurd concept, thus the ontology is not satisfiable.

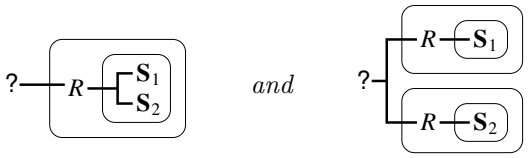
Next, we provide the above mentioned metarules, some of them will be used in the completeness proof.

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, although we need to keep track in order to distinguish whether a subtree is positively and negatively enclosed. 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 (adopted from [70]).

Lemma 9.1 (Cut-and-Paste Theorem) *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 $\mathbf{T} \vdash \mathbf{T}[\,\mathbf{S}_1 \,/\, \mathbf{S}_2\,]$.*

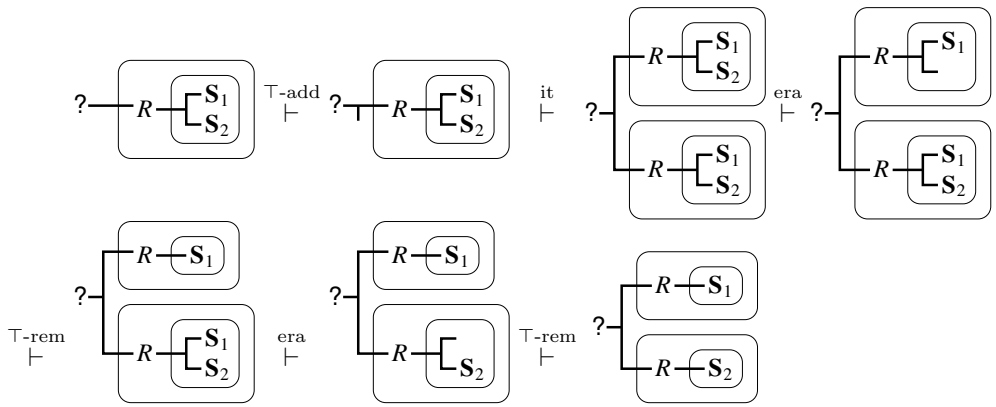
The next lemma corresponds to the equivalence of the $\mathcal{ALC}$-concepts $\forall R.(C \sqcap D)$ and $\forall R.C \sqcap \forall R.D$.

Lemma 9.2 (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*

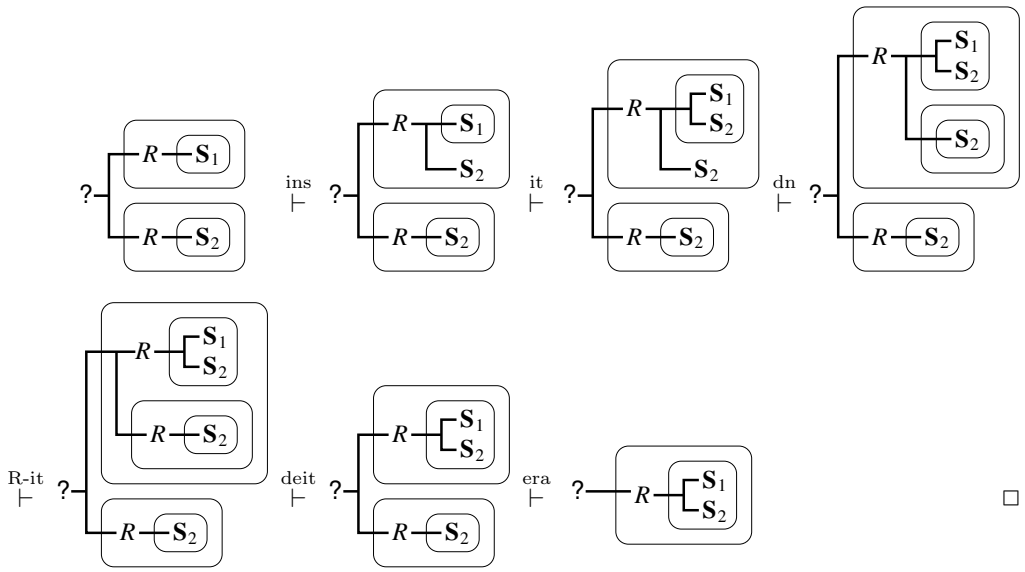


are equivalent.

Proof: We start with the direction from left to right.



Again, we see that 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 following, these two steps are combined without explicitly mentioning the $\top$-removal/addition rule.

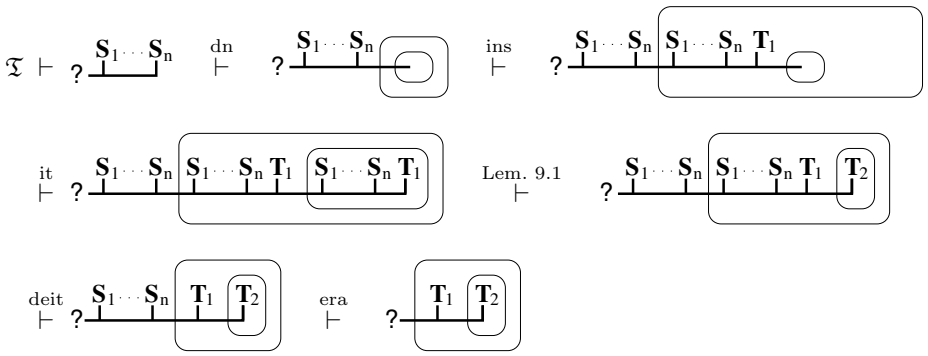


For $\mathcal{ALC}$, the full deduction theorem holds.

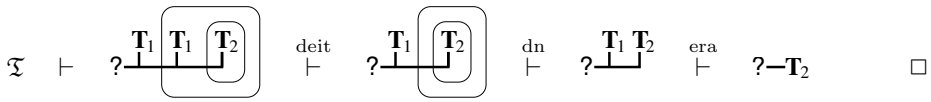
Theorem 9.3 (Deduction Theorem) *Let $\mathfrak{T}$ be a set of $\mathcal{ALC}$-trees, let $\mathbf{T}_1, \mathbf{T}_2$ be two $\mathcal{ALC}$-trees. Then we have*

$$\mathfrak{T} \cup \{\mathbf{T}_1\} \vdash \mathbf{T}_2 \quad \Longleftrightarrow \quad \mathfrak{T} \vdash \neg(\mathbf{T}_1 \sqcap \neg\mathbf{T}_2)$$

Proof: In the ongoing proofs, again applications of the $\top$-removal/addition rule are not explicitely mentioned. We start with '$\Rightarrow$'. 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:



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



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

Lemma 9.3 *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. Common calculi (like Hilbert-style calculi, natural deduction, or sequent calculi) are shallow in the understanding that the rules of the calculus modify formulas only at their top-level. In constrast, the rules presented in this paper 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 9.4 *Let $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{S}_1$, $\mathbf{S}_2$ be $\mathcal{ALC}$-trees with $\mathbf{T}_2 = \mathbf{T}_1[\,\mathbf{S}_2 \,/\, \mathbf{S}_1\,]$.*

(i) If $\mathbf{S}_1 \models \mathbf{S}_2$ and the substitution takes place in an even, then $\mathbf{T}_1 \models \mathbf{T}_2$.
(ii) If $\mathbf{S}_2 \models \mathbf{S}_1$ and the substitution takes place in an odd, then $\mathbf{T}_1 \models \mathbf{T}_2$.
(iii) 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 9.4 (Soundness of the Calculus) *If $(\Delta^{\mathcal{I}}, \mathcal{I})$ is a model and if $\mathbf{T}'$ is obtained from $\mathbf{T}$ by one of the rules of the calculus, 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 9.4(3). 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 9.4(1) yields the soundness of the erasure of a a positively enclosed subtree, and Lemma 9.4(2) 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 9.4(1) yields the soundness of deiterating a role R from an odd, and Lemma 9.4(2) 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,

$$a \in \mathcal{I}(\mathbf{T}_w) \quad \Longleftrightarrow \quad a \in \mathcal{I}(\mathbf{T}'_w) \tag{2}$$

holds for $w = t$. We have $\mathbf{T}' = \mathbf{T}[\,\mathbf{T}'_t \,/\, \mathbf{T}_t\,]$, so once Equation. (2) is proven for $w = t$, we can now apply Lemma 9.4(3) 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 Equation (2).

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 Equation (2) holds. Now let us assume the alternate case, that we have $a \in \mathcal{I}(\mathbf{S})$. We will prove that Equation (2) holds for each w with $t \geq w \geq v$ by induction over w.

We have $\mathbf{T}_v = \top$, $\mathbf{T}'_v = \mathbf{S}$, $a \in \mathcal{I}(\top)$ and $a \in \mathcal{I}(\mathbf{S}')$, so Equation (2) holds for $w = v$.

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 $\mathbf{T}'_w = (w, \neg) \oplus \mathbf{T}'_u$. 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 Equation (2) holds for w.

For $\nu(w) = \sqcap$, 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 Equation (2) holds.

This finishes the induction, so we conclude Equation (2) 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. The following proof is a standard proof for the completeness of a Hilbert-style calculus for the multi-modal logic K, adapted to our system of $\mathcal{ALC}$-trees.

Theorem 9.5 (Completeness of the Calculus) *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:*

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

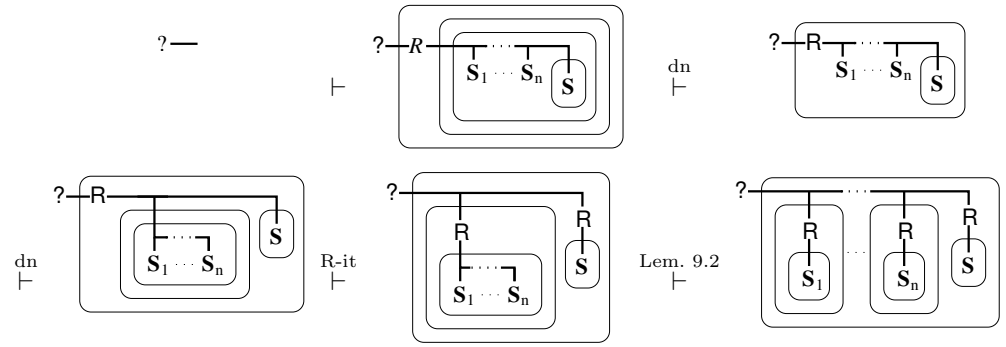
Proof: We assume that there is no derivation of $\mathbf{T}$ from $\mathfrak{T}$, and we show that $\mathfrak{T} \not\models \mathbf{T}$.

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:

$$\text{If } \mathfrak{S} \text{ is consistent, where } R\neg\mathbf{S} \in \mathfrak{S}, \text{ then } \mathfrak{S}^R \cup \{\neg\mathbf{S}\} \text{ is also consistent.} \tag{3}$$

It is easier to show the contraposition of Eqn. (3), 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 9.3 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. (3) 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

$$\mathbf{S} \in \mathfrak{S}_\mathrm{m} \Longleftrightarrow \neg\mathbf{S} \notin \mathfrak{S}_\mathrm{m} \tag{4}$$

$$\mathbf{S} \sqcap \mathbf{S}' \in \mathfrak{S}_\mathrm{m} \Longleftrightarrow \mathbf{S} \in \mathfrak{S}_\mathrm{m} \text{ and } \mathbf{S}' \in \mathfrak{S}_\mathrm{m} \tag{5}$$

for arbitrary $\mathcal{ALC}$-Trees $\mathbf{S}, \mathbf{S}'$. We only prove Eqn. (4), the proof of Eqn. (5) 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. 9.3 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 (4) is proven.

Now we construct a model $(\Delta^\mathcal{I}, \mathcal{I})$ as follows. We set

$$\begin{aligned}
\Delta^\mathcal{I} &:= \{\mathfrak{S}_\mathrm{m} \mid \mathfrak{S}_\mathrm{m} \text{ is maximal consistent}\} \\
\mathcal{I}(A) &:= \{\mathfrak{S}_\mathrm{m} \mid \mathbf{T}_A \in \mathfrak{S}_\mathrm{m}\} \\
\mathcal{I}(R) &:= (\mathfrak{S}_\mathrm{m}, \mathfrak{T}_\mathrm{m}) \mid \mathfrak{S}_\mathrm{m}^R \subseteq \mathfrak{T}_\mathrm{m}\}\}
\end{aligned}$$

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

$$\mathfrak{S}_\mathrm{m} \in \mathcal{I}(\mathbf{S}) \quad \Longleftrightarrow \quad \mathbf{S} \in \mathfrak{S}_\mathrm{m} \tag{6}$$

For a concept name A, Eqn. (6) holds by the definition of the model. For a tree $\neg\mathbf{S}$, we have $\mathfrak{S}_\mathrm{m} \in \mathcal{I}(\neg\mathbf{S}) \overset{\text{Def. } \mathcal{I}}{\Longleftrightarrow} \mathfrak{S}_\mathrm{m} \notin \mathcal{I}(\mathbf{S}) \overset{\text{I.H.}}{\Longleftrightarrow} \mathbf{S} \notin \mathfrak{S}_\mathrm{m} \overset{\text{Eqn. (4)}}{\Longleftrightarrow} \neg\mathbf{S} \in \mathfrak{S}_\mathrm{m}$. For a tree $\mathbf{S} \sqcap \mathbf{S}'$, Eqn. (6) is similarly proven using Eqn. (5). It remains to consider role names.

Let $R \in \mathcal{R}$. We first prove Eqn. (6) 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. (6) is proven for all $\mathcal{ALC}$-trees which have less occurrences of R than $\mathbf{S}'$. Def. 7.1 yields

$$\mathfrak{S}_\mathrm{m} \in \mathcal{I}(\neg R\neg\mathbf{S}) \quad \Longleftrightarrow \quad \text{for all } \mathfrak{T}_\mathrm{m} \text{ with } (\mathfrak{S}_\mathrm{m}, \mathfrak{T}_\mathrm{m}) \in \mathcal{I}(R) \text{ we have } \mathfrak{T}_\mathrm{m} \in \mathcal{I}(\mathbf{S}) \tag{7}$$

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. (7) yields $\mathfrak{S}_\mathrm{m} \in \mathcal{I}(\mathbf{S}')$. Next suppose $\mathbf{S}' \notin \mathfrak{S}_\mathrm{m}$. Eqn. (4) 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. (3). 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. (7). Hence Eqn. (6) is proven for $\mathbf{S}' = \neg R\neg\mathbf{S}$.

To finish the proof of Eqn. (6), 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. (6) 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 $\mathbf{S} \in \mathfrak{T} \overset{\text{Eqn. (6)}}{\iff} \mathbf{S} \in \mathfrak{S}_\mathrm{m}$ 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 $\mathfrak{T}_\mathrm{m}^0 \supseteq \mathfrak{T} \cup \{\neg\mathbf{T}\}$. On the one hand, we now have $\mathfrak{T}_\mathrm{m}^0 \in \bigcap_{\mathbf{S}\in\mathfrak{T}} \mathcal{I}(\mathbf{S})$. On the other hand, we have $\mathbf{T} \notin \mathfrak{T}_\mathrm{m}^0$, thus $\mathfrak{T}_\mathrm{m}^0 \notin \mathcal{I}(\mathbf{T})$ by Eqn. (6). So we obtain $\bigcap_{\mathbf{S}\in\mathfrak{T}} \mathcal{I}(\mathbf{S}) \not\subseteq \mathcal{I}(\mathbf{T})$, which means $\mathfrak{T} \not\models \mathbf{T}$. □

10. Conclusion and Further Research

This paper has discussed different diagrammatic reasoning systems – spider and constraint diagrams on the one hand, existential and conceptual graphs on the other – as candidates for a diagrammatic version of DL. It has been argued that spider and constraint diagrams are not as well suited to a diagrammatic version of DLs as conceptual graphs and existential graphs. More specifically, a version of conceptual graphs including negation and free variables, and relation graphs – existential graphs with free variables – are suited as a diagrammatic version of the smallest propositionally closed DL $\mathcal{ALC}$. We claim further that relation graphs are slightly easier to comprehend. We then elaborated a fragment $\mathrm{RG}^{\mathcal{ALC}}$ of relation graphs that correspond to the $\mathcal{ALC}$, and provided a diagrammatic, sound and complete calculus for $\mathrm{RG}^{\mathcal{ALC}}$, based on Peirce's rules for existential graphs.

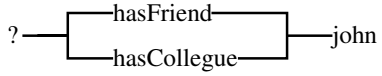
The results presented in this paper provide promising steps toward further investigating RGs as diagrammatic versions of DLs. There are several open problems that have to be addressed by extension research. First, our approach could be extended to other constructs of DL. For example, to investigate how the intersection of roles, inverse of roles, and unary (one-of) concepts can be built into the diagrammatic reasoning frameworks (this would obtain a strict extension of the results of [17,3]).

From the syntactical and semantical point of view this will not yield any more significant problems than those we have already encountered. We can extend the graphs of $\mathrm{RG}^{\mathcal{ALC}}$ such that we allow objects at the end of heavily drawn lines (adding constants and function names to RGs is elaborated in [26]) or cycles, where only two branching points are involved [16]. For example, the concept description

$$\textsc{Man} \sqcap \exists(\textsc{hasFriend} \sqcap \textsc{hasColleague}).\{\textsc{john}\}$$

[16] In [3] the authors allow the intersection of roles and write that the corresponding SCGs are trees, where they say that "a SCG t is called a tree iff t contains no cycles of length greater than 2."

describing 'some man having John as friend and colleague', could be depicted as follows:

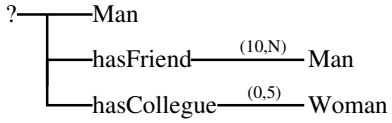


Similarly, from a syntactic and semantic point of view, allowing concepts of the form $\{o_1, \ldots, o_n\}$ is not problematic either.

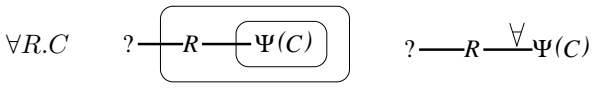
More importantly, the approach we present can to be extended to number restriction. In this case we encounter the $\mathcal{SH}$-family of DLs and in so doing have the capacity to provide diagrammatic representations of the underlying DLs used in OWL. From a representational point of view a commonly used notation for number restrictions could be used: such as the (min, max)-notation used widely in both entity-relationship-diagrams (EER) and UML. For example, the concept

$$\textsc{Man} \sqcap \geq 10\, \textsc{hasFriend}.\textsc{Man} \sqcap \leq 5\, \textsc{hasColleague}.\textsc{Woman}$$

of men who have at least 10 men as friends and not more than 5 female colleagues, could be depicted as follows:



Further the syntactical restrictions of DL yield corresponding syntactical restrictions in $\mathrm{RG}^{\mathcal{ALC}}$. It is possible to benefit from these restrictions by adding some "syntactic sugar" to the graphs, which is not possible for the general class of RGs. This applies for example to the universal quantifier of DL. Below is the general form of the value restriction of DL, its translation with Ψ, and an modified version without cuts, where the heavily drawn line on the right is now marked with an universal quantifier.



Similar considerations are imposed by allowing an expicit notation for disjunction. This would allow cut-free representations for all formulas of $\mathcal{ALE}$. For example, Fig. 20 provides an $\mathcal{ALC}$-concept and two possible improvements of the corresponding Peirce diagrams.

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

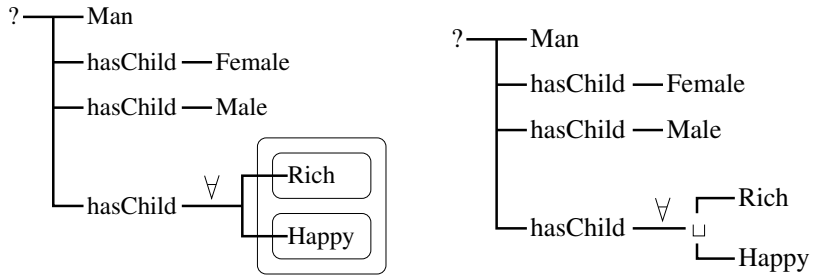


Fig. 20. An $\mathcal{ALC}$-concept descriptions and two RGs with abbreviation for ∀ and disjunction

Finally, a main aspect of research in DLs focuses on complexity. At first glance, the complexity issues should be independent from the notation but there are some promising

results for Peirce's existential alpha graphs (corresponding to propositional logic) in terms of computability that show that Peirce's rules have some nice complexity results. In Dau [25], it is shown that the erasure-rule, which is the only rule for Peirce's existential alpha graphs that does not satisfy the finite subformula property, is admissible, and in the remaining calculus, Statman's formulas can still be proven in polynomial time.

Roughly speaking, Peirce's calculus without the erasure rule has the nice computational properties of sequent calculi without the cut-rule [17], while it is as fast as sequent calculi that includes the cut-rule [18]. It can be investigated whether similar results can be obtained for the calulus presented in this paper.

For the moment, the first key issue addressed in his paper is the question of what (if any) existing and well-known diagrammatic reasoning frameworks can be suitably adapted to DL. We are confident that existential and conceptual graphs can be used as a diagrammatic framework for DL and have presented our reasons for this conclusion. The second key issue was to elaborate a diagrammatic reasoning system for $\mathcal{ALC}$ based on existential graphs, including a diagrammatic, sound and complete calculus. In the long run, we intend to develop major variants of DLs as mathematically precise diagrammatic reasoning systems. The intention is to render DL more user-friendly. Such systems need to be evaluated against the traditional textual form of DL in order to measure any improvement to the readability of the DL in its diagrammatic form. Such evaluations involve usability experiments and substantial empirical testing and are beyond the scope of this paper which restricts itself to the technical feasibility of the candidate diagrammatic reasoning frameworks to DL.

11. Appendix

The following abbreviations are used:

DL	description logic	**FOL**	first order logic
SD	spider diagram	**CD**	constraint diagram
CG	conceptual graph	**SCG**	simple conceptual graph
CGwC	concept(ual) graph with cuts	**EG**	existential graph
RG	relation graph	**OWL**	Ontology Web Language
UML	Unified Modeling Language	**RDF**	Resource Description Framework

12. Acknowledgements

We like to thank the reviewers for their helpful remarks and also Andrew Fish for his valuable comments on using CDs for $\mathcal{ALC}$.

[17] In sequent caluli the cut-rule does not satisfy the finite subformula property, which is one of the reasons that Gentzen's famous cut-elimination-theorem is so important.
[18] Statman's formulas need proofs in exponential time in the cut-free sequent calculi.

References

[1] M. Naci Akkøk. *Towards the Principles of Designing Diagrammatic Modeling Languages: Some Visual, Cognitive and Foundational Aspects*. PhD thesis, Department of Informatics Faculty of Mathematical and Natural Sciences, University of Oslo, 2004.

[2] F. Baader, R. Molitor, and S. Tobies. The guarded fragment of conceptual graphs. RWTH LTCS-Report. See also [3]. Available at: `http://lat.inf.tu-dresden.de`, 1998.

[3] F. Baader, R. Molitor, and S. Tobies. Tractable and decidable fragments of conceptual graphs. In Tepfenhart and Cyre [81], pages 480–493. Excerpt of [2].

[4] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[5] Jean-François Baget. Homomorphismes d'hypergraphes pour la subsumption en rdf/rdfs. *Langages et modèles à objets 2004 (actes 10e conférence), RSTI - L'objet (numéro spécial)*, 10(2-3):203–216, 2004.

[6] Jean-François Baget. Rdf entailment as a graph homomorphism. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 82–96. Springer, Berlin – Heidelberg – New York, 2005.

[7] J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, Cambridge, 1983. deutsch: Situationen und Einstellungen: Grundlagen der Situationssemantik, De Gruyter, Berlin 1987.

[8] Shai Berger. Studies on the uses and usefulness of diagrams. Master's thesis, Institute for Logic, Language and Computation (ILLC), University of Amsterdam, 2000.

[9] Alan Blackwell, Kim Marriott, and Atsushi Shimojima, editors. *Diagrammatic Representation and Inference: Third International Conference, Diagrams 2004*, volume 2980 of *LNAI*. Springer, Berlin – Heidelberg – New York, 2004.

[10] R. J. Brachman. On the epistemological status of semantic networks. In N. V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press, Orlando, 1979.

[11] Ronald J. Brachman and James G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[12] Sara Brockmans, Raphael Volz, Andreas Eberhart, and Peter Löffler. Visual modeling of owl dl ontologies using uml. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 198–213. Springer, Berlin – Heidelberg – New York, 2004.

[13] Robert W. Burch. *A Peircean Reduction Thesis: The Foundation of Topological Logic*. Texas Tech. University Press, Texas, Lubbock, 1991.

[14] M. Chein and M.-L. Mugnier. Conceptual graphs: Fundamental notions. *Revue d'Intelligence Artificiell*, 6:365–406, 1992.

[15] M. Chein and M.-L. Mugnier. Conceptual graphs are also graphs. Technical report, LIRMM, Université Montpellier II, 1995. Rapport de Recherche 95003.

[16] Michel Chein and Marie-Laure Mugnier. Concept types and coreference in simple conceptual graphs. In Wolff et al. [88], pages 303–318.

[17] Pascal Coupey and Catherine Faron. Towards correspondence between conceptual graphs and description logics. In Marie-Laure Mugnier and Michel Chein, editors, *ICCS*, volume 1453 of *LNAI*, pages 165–178. Springer, Berlin – Heidelberg – New York, 1998.

[18] Frithjof Dau. Concept graphs without negations: Standardmodels and standardgraphs. In Aldo de Moor, Wilfried Lex, and Berhard Ganter, editors, *Conceptual Structures for Knowledge Creation and Communication*, volume 2746 of *LNAI*, pages 243–256. Springer, Berlin – Heidelberg – New York, 2003. This paper is a part of [19] as well.

[19] Frithjof Dau. *The Logic System of Concept Graphs with Negations and its Relationship to Predicate Logic*, volume 2892 of *LNAI*. Springer, Berlin – Heidelberg – New York, November 2003.

[20] Frithjof Dau. Query graphs with cuts: Mathematical foundations. In Blackwell et al. [9], pages 32–50.

[21] Frithjof Dau. Types and tokens for logic with diagrams: A mathematical approach. In Wolff et al. [88], pages 62–93.

[22] Frithjof Dau. Fixing shin's reading algorithm for peirce's existential graphs. In Dave Barker-Plummer, Richard Cox, and Nik Swoboda, editors, *Diagrams*, volume 4045 of *LNAI*, pages 88–92. Springer, Berlin – Heidelberg – New York, 2006.
[23] Frithjof 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.
[24] Frithjof Dau. Rdf as graph-based, diagrammatic logic: Syntax, semantics, calculus, normalforms. In F. Esposito, Z.W. Ras, D. Malerba, and G Semeraro, editors, *Foundations of Intelligent Systems*, volume 4203 of *Lecture Notes in Computer Science*. Springer, Berlin – Heidelberg – New York, 2006.
[25] Frithjof Dau. Some notes on proofs with alpha graphs. In Øhrstrøm et al. [54], pages 172–188.
[26] Frithjof Dau. Constants and functions in existential graphs. In *Proceedings of the ICCS*. Springer, Berlin – Heidelberg – New York, 2007. To appear.
[27] Frithjof Dau and Joachim Hereth Correia. Two instances of peirce's reduction thesis. In Rokia Missaoui and Jürg Schmid, editors, *ICFCA*, volume 3874 of *Lecture Notes in Computer Science*, pages 105–118. Springer, 2006.
[28] G. Ellis, R. Levinson, W. Rich, and J. F. Sowa, editors. *Conceptual Structures: Applications, Implementation, and Theory*, volume 954 of *LNAI*. Springer, Berlin – Heidelberg – New York, 1995.
[29] Andrew Fish, Jean Flower, and John Howse. The semantics of augmented constraint diagrams. *Journal of Visual Languages and Computing*, 16:541–573, 2005.
[30] Andrew Fish and Judith Masthoff. Do monkeys like elephants or do elephants watch monkeys? an empirical study into the default reading of constraint diagrams. technical report vmg.05.2. Available at: [84], 2005.
[31] Andrew Fish and Judith Masthoff. An experimental study into the default reading of constraint diagrams. In *VL/HCC*, pages 287–289. IEEE Computer Society, 2005.
[32] Jean Flower, Judith Masthoff, and Gem Stapleton. Generating proofs with spider diagrams using heuristics. In *International Workshop on Visual Languages and Computing, San Francisco, September 2004*, 2004.
[33] Jean Flower, Judith Masthoff, and Gem Stapleton. Generating readable proofs: A heuristic approach to theorem proving with spider diagrams. In Blackwell et al. [9], pages 166–181.
[34] Jean Flower and Gem Stapleton. Automated theorem proving with spider diagrams. *Electr. Notes Theor. Comput. Sci.*, 91:246–263, 2004.
[35] Brian R. Gaines. An interactive visual language for term subsumption languages. In *IJCAI*, pages 817–823, 1991.
[36] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin – Heidelberg – New York, 1999.
[37] Joseph Gil, John Howse, and Stuart Kent. Formalizing spider diagrams. In *IEEE Symposium on Visual Languages*, pages 130–137, 1999.
[38] Eric M. Hammer. *Logic and Visual Information*. CSLI Publications, Stanford, California, 1995.
[39] Weiss Hartshorne and Burks, editors. *Collected Papers of Charles Sanders Peirce*, Cambridge, Massachusetts, 1931–1935. Harvard University Press.
[40] Joachim Hereth Correia and Reinhard Pöschel. The power of peircean algebraic logic (pal). In Peter W. Eklund, editor, *ICFCA*, volume 2961 of *Lecture Notes in Computer Science*, pages 337–351. Springer, Berlin – Heidelberg – New York, 2004.
[41] Joachim Hereth Correia and Reinhard Pöschel. The teridentity and peircean algebraic logic. In Øhrstrøm et al. [54].
[42] Ian Horrocks. Owl: A description logic based ontology language. In Peter van Beek, editor, *CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 5–8. Springer, 2005.
[43] J. Howse, F. Molina, Sun-Joo Shin, and J. Taylor. On diagram tokens and types. In Mary Hegarty, Bernd Meyer, and N. Hari Narayanan, editors, *Diagrams*, volume 2317 of *Lecture Notes in Computer Science*, pages 146–160. Springer, Berlin – Heidelberg – New York, 2002.
[44] Michel Chein Jean-Pierre Aubert, Jean-Francois Baget. Simple conceptual graphs and simple concept graphs. In *Proceedings of ICCS 2006*, LNAI. Springer, Berlin – Heidelberg – New York, 2006.
[45] Stuart Kent. Constraint diagrams: Visualizing assertions in object-oriented models. In *OOPSLA*, pages 327–341. ACM Press, 1997.
[46] 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.

[47] Zenon Kulpa. Diagrammatic representation and reasoning. *Machine Graphics and Vision*, 3(1/2):77–103, 1994.
[48] Jill H. Larkin and Herbert A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–100, 1987.
[49] Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.
[50] D. Lukose, editor. *Conceptual Structures: Fulfilling Peirce's Dream*, volume 1257 of *LNAI*. Springer, Berlin – Heidelberg – New York, 1997.
[51] J. McCarthy. Notes on formalizing context. In *IJCAI-93: 13th International Joint Conference on Artificial Intelligence*, pages 555–560. Morgan Kaufmann, 1993.
[52] Daniele Nardi and Ronald J. Brachman. An introduction to description logics. In Baader et al. [4], pages 1–40.
[53] John T. Nosek and Itzhak 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.
[54] Peter Øhrstrøm, Henrik Schärfe, and Pascal Hitzler, editors. *Conceptual Structures: Inspiration and Application*, volume 4068 of *Lecture Notes in Computer Science*. Springer, Berlin – Heidelberg – New York, 2006.
[55] Helmut Pape. *Charles S. Peirce: Phänomen und Logik der Zeichen*. Suhrkamp Verlag Wissenschaft, Frankfurt am Main, Germany, 1983. German translation of Peirce's Syllabus of Certain Topics of Logic.
[56] Octavian Patrascoiu, Simon Thompson, and Peter Rodgers. Tableaux for diagrammatic reasoning. In Philip Cox and Trevor Smedley, editors, *Proceedings of the 2005 International Workshop on Visual Languages and Computing*, pages 279–286, September 2005.
[57] Charles Sanders Peirce. *MS 478: Existential Graphs.* Harvard University Press, 1931–1935. Partly published in of [39] (4.394-417). Complete german translation in [55].
[58] Charles Sanders Peirce and John F. Sowa. Existential Graphs: MS 514 by Charles Sanders Peirce with commentary by John Sowa, 1908, 2000.
Available at: `http://www.jfsowa.com/peirce/ms514.htm`.
[59] Silke Pollandt. Relational constructions on semiconcept graphs. In B. Ganter and G. Mineau, editors, *Conceptual Structures: Extracting and Representing Semantics.* Manuscript Stanford, 2001.
[60] Silke Pollandt. Relation graphs: A structure for representing relations in contextual logic of relations. In Priss et al. [61], pages 24–48.
[61] Uta Priss, Dan Corbett, and Galia Angelova, editors. *Conceptual Structures: Integration and Interfaces*, volume 2393 of *LNAI*, Borovets, Bulgaria, July, 15–19, 2002. Springer, Berlin – Heidelberg – New York.
[62] Don D. Roberts. *The Existential Graphs of Charles S. Peirce.* Mouton, The Hague, Paris, 1973.
[63] Don D. Roberts. The existential graphs. *Computers Math. Appl..*, 23(6–9):639–63, 1992.
[64] Henrik Schärfe, Ulrik Petersen, and Peter Øhrstrøm. On teaching conceptual graphs. In Priss et al. [61], pages 285–298.
[65] Atsushi Shimojima. *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`.
[66] Sun-Joo Shin. *The Logical Status of Diagrams.* Cambridge University Press, 1994.
[67] Sun-Joo Shin. *The Iconic Logic of Peirce's Graphs.* Bradford Book, Massachusetts, 2002.
[68] John F. Sowa. Conceptual graphs: Draft proposed american national standard.
Old version: `http://www.jfsowa.com/cg/cgdpansw.htm`,
New version: `http://www.jfsowa.com/cg/cgstandw.htm`. See also [75].
[69] John F. Sowa. Semantic foundations of contexts. This paper is a revised merger of [72] and [74]. Available at: `http://www.jfsowa.com/ontology/contexts.htm`.
[70] John F. Sowa. *Conceptual structures: information processing in mind and machine.* Addison-Wesley, Reading, Mass., 1984.
[71] John F. Sowa. Conceptual graphs summary. In T. E. Nagle, J. A. Nagle, L. L. Gerholz, and P. W. Eklund, editors, *Conceptual Structures: current research and practice*, pages 3–51. Ellis Horwood, 1992.
[72] John F. Sowa. Syntax, semantics, and pragmatics of contexts. In Ellis et al. [28], pages 1–15. See also [69].
[73] John F. Sowa. Logic: Graphical and algebraic. manuscript, Croton-on-Hudson, 1997.

[74] John F. Sowa. Peircean foundation for a theory of context. In Lukose [50], pages 41–64. See also [69].
[75] John F. Sowa. Conceptual graphs: Draft proposed american national standard. In Tepfenhart and Cyre [81], pages 1–65. See also [68].
[76] John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole, Pacific Grove, CA, 2000.
[77] Gem Stapleton. *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`.
[78] Gem Stapleton. A survey of reasoning systems based on euler diagrams. *Electr. Notes Theor. Comput. Sci.*, 134:127–151, 2005.
[79] Gem Stapleton, John Howse, and John Taylor. Spider diagrams. *LMS Journal of Computation and Mathematics*, 8:145–194, 2005. J Howse: This is the definitive spider diagrams paper but is a little technical in places.
[80] John Stewart. Theorem proving using existential graphs. Master's thesis, University of California at Santa Cruz, 1996. advisor: Robert Levinson.
[81] Wiliam Tepfenhart and W. Cyre, editors. *Conceptual Structures: Standards and Practices*, volume 1640 of *LNAI*. Springer, Berlin – Heidelberg – New York, 1999.
[82] Bram van Heuveln. Existential graphs. Presentations and Applications at: `http://www.rpi.edu/~heuveb/research/EG/eg.html`, 2003.
[83] Adam Vile and Simon Polovina. Possibilites in peirce's existential graphs for logic education. available from the authors, 1998.
[84] VMG. The visual modeling group homepage, university of brighton. `http://www.cmis.brighton.ac.uk/Research/vmg/`.
[85] M. Wermelinger. Conceptual graphs and first-order logic. In Ellis et al. [28], pages 323–337.
[86] Rudolf Wille. Conceptual graphs and formal concept analysis. In Lukose [50], pages 290–303.
[87] Rudolf Wille. Boolean concept logic. In Bernhard Ganter and Guy W. Mineau, editors, *Conceptual Structures: Logical, Linguistic and Computational Issues*, volume 1867 of *LNAI*, pages 263–276, Darmstadt, Germany, 2000. Springer, Berlin – Heidelberg – New York.
[88] Karl Erich Wolff, Heather D. Pfeiffer, and Harry S. Delugach, editors. *Conceptual Structures at Work: 12th International Conference on Conceptual Structures, ICCS 2004, Huntsville, AL, USA, July 19-23, 2004. Proceedings*, volume 3127 of *Lecture Notes in Computer Science*. Springer, Berlin – Heidelberg – New York, 2004.
[89] Jay J Zeman. *The Graphical Logic of C. S. Peirce*. PhD thesis, University of Chicago, 1964. Available at: `http://www.clas.ufl.edu/users/jzeman/`.