The Advent of Formal Diagrammatic Reasoning Systems

Frithjof Dau

SAP Research CEC Dresden

Abstract. In knowledge representation and reasoning systems, diagrams have many practical applications and are used in numerous settings. Indeed, it is widely accepted that diagrams are a valuable aid to intuition and help to convey ideas and information in a clear way. On the other side, logicians have viewed diagrams as informal tools, but which cannot be used in the manner of formal argumentation. Instead, logicians focused on symbolic representations of logics. Recently, this perception was overturned in the mid 1990s, first with seminal work by Shin on an extended version of Venn diagrams. Since then, certainly a growth in the research field of formal reasoning with diagrams can be witnessed. This paper discusses the evolution of formal diagrammatic logics, focusing on those systems which are based on Euler and Venn-Peirce diagrams, and Peirces existential graphs. Also discussed are some challenges faced in the area, some of which are specifically related to diagrams.

1 Introduction

Formal Concept Analysis (FCA) is a mathematical theory applied successfully in a wide range. The impact and success of FCA and the large number of applications in the real world cannot be explained solely with the mathematical results and the mathematical power of FCA. The driving force behind FCA lies in the understanding of mathematics as a science which encompasses the philosophical basis and the social consequences of this discipline as well, and a main goal of FCA from its very beginning has been the support of rational communication and the representation and processing of knowledge. Lattice theory is reworked in order to integrate and to rationalize origins, connections to and interpretations in the real world. As Wille says in [78]:

> The aim is to reach a structured theory which unfolds the formal thoughts according to meaningful interpretations allowing a broad communication and critical discussion of the content.
>
> Wille, 1996

Thus the results of lattice theory had in FCA to be presented in a way which makes them understandable, learnable, available and criticizable, particularly for non-mathematicians. One means to achieve this goal is the diagrammatic representations in form of their Hasse diagrams. FCA, being is a mathematical theory

formalizing the philosophical notion of concepts, has been later extended to *contextual logic* in order to revitalize the traditional philosophical understanding of logic which is based the the doctrines of concepts, judgments and conclusions. Again, an important aspect of contextual logic is that its core notions, i.e. both judgments and conclusions, can be diagrammatically represented (for this purpose, Sowa's conceptual graphs [70] are utilized).

It is widely accepted that diagrams play an important role for representing information in accessible and intuitive ways. In mathematics, however, there does still exist a long-standing prejudice against non-symbolic representation, particularly in mathematical logic. Without doubt diagrams are often used in mathematical reasoning, but usually only as illustrations or thought aids. Diagrams, many mathematicians say, are not rigorous enough to be used in a proof, or may even mislead us in a proof. Thus diagrams have been excluded from formal proof techniques and were considered only as a heuristic aid.

Interestingly, most of the ancient systems which can be considered as predecessors of formal logic are diagrammatic systems. We name Euler circles, Venn diagrams and Venn-Peirce diagrams, Frege's Begriffsschrift and Peirce's existential graphs. It was not until the end of the 19th century that symbolic notations took over in mathematical logic.

After more than a century of an absolute dominance of symbolic notations for logic, the last two decades show an increasing interest in formal elaborations of diagrammatic reasoning systems (DRSs), that is, formal logic systems with a precise syntax, semantics, and (diagrammatic) reasoning facilities. In this paper, we investigate this advent of diagrammatic reasoning systems from various perspectives. In order to do so and to motivate the use of DRSs, we first discuss in Sec. 2 possible applications of DRSs in software engineering and knowledge representation. An introduction into the Euler-Venn-Peirce family of diagrams as well as into existential graphs is provided in Sec. 3. Seminal work in the field of DRSs which present and elaborate these historical system is presented in section 4. Although that this seminal work laid the path towards a mathematically precise development of DRSs, it can be argued that they still do not fulfill the requirements of a rigorous mathematical system. Sec. 5 presents methodologies for developing DRSs in a precise manner. Contemporary systems which follow these methodologies are then presented in Sec. 6. Finally, we conclude with a discussion of the area.

2 Examples of Application Areas

In order to motivate the development of DRSs, we present in this section two different application areas of DRSs in the fields of software engineering and knowledge representation in the Semantic Web.

2.1 Software Engineering

In Software Engineering, we observe an increasing demand and development of diagrammatic languages which are used for describing, specifying and communi-

cating a wide range of modeling aspects. In software engineering, different kinds of stakeholders are involved in the modeling process. Besides IT professionals like programmers and developers, this includes people like managers and customers, which often do not have a dedicated technical expertise. As there is a need of software specifications being accessible to all stakeholders, symbolic languages and formalizations are not suited for modeling purposes, and various diagrammatic languages like UML (unified modeling language)[1] or BPMN (business process modeling notation)[2] have been developed.

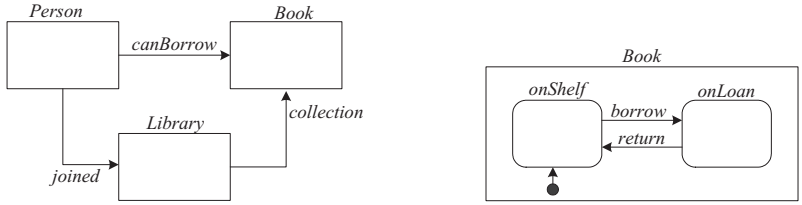

Fig. 1. A UML class diagram and a UML state chart

UML is in fact a whole suite of (mostly) diagrammatic notations, including *class diagrams*, *state charts* and various others. In Fig. 1, a UML class diagram and a UML state chart are depicted. Both of them refer to a scenario for describing a library lending system. The class diagram expresses relationship between the classes *Person*, *Library* and *Book*: persons can borrow books, persons can join libraries, and libraries have collection of books. Books can be in two different states, namely $onShelf$ or *onLoan*, as it is expressed by the state chart. Initially (expressed by the bold dot and arrow), books are on the shelf. If a book is borrowed, its state changes to being on loan, and vice versa, a book on loan becomes being on shelf if it is returned.

There does exist a part of the UML which is non-diagrammatic: This is the Object Constraint Language (OCL) which has been developed to describe formal constraints on software models. As a very simple example of a constraint we might wish to enforce on a library lending system is that people can only borrow books that are in the collections of libraries they have joined. Using the OCL, this is achieved as follows:

```
Context Person inv:((self.joined.collection->asSet)->includesAll(self.canBorrow))
```

Being a symbolic notation, OCL does not follow the general diagrammatic approach of UML and is for this reason probably not as easily understandable diagrammatic parts of UML. Thus it is reasonable to develop a diagrammatic variant of OCL. A groundbreaking approach towards this aim has been undertaken by Kent, who introduced in [41] the class of *constraint diagrams*. An

[1] `http://www.uml.org`
[2] `http://www.bpmn.org`

example for such a diagram is provided in Fig. 2. This diagrams expresses the same constraint as its symbolic counterpart we have just given, but compared to the symbolic notation, the diagram fits better in the general visual theme of OCL.

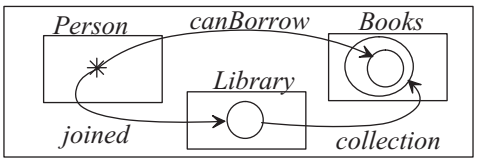

Fig. 2. A constraint diagram

Mainly driven by the Visual Modeling Group (VMG) in Brighton, UK[3], constraint diagrams are developed as a formal DRS, including a formal syntax, FOL-based, semantics, and diagrammatic reasoning facilities.

2.2 Knowledge Representation and the Semantic Web

Is has long been argued that diagrams are particularly useful for knowledge representation systems [56, 28, 48]. In this section, we focus on knowledge representation within the Semantic Web, particularly on RDF(S) and OWL.

The underlying layer for knowledge representation within the Semantic Web is the Resource Description Framework (RDF) and its extension RDF Schema (RDFS). The essential idea of RDF is that each atomic piece of information can be represented as a triple `(subject predicate object)`, and an RDF Knowledge Base is a set of RDF triples. There are different representations of RDF KBs. First of all, there exists a machine processable XML serialization of RDF. Secondly, we have the notion of an RDF KB as a set of RDF triples. Thirdly, we can represent an RDF KB by means of an labeled graph, where a triple `(s p o)` is modeled by two vertices labeled with `s` and `o`, respectively, and which are connected by an edge labeled with `o`.

Tim-Berners Lee, inventor of the WWW and the driving force behind the semantic web, recently reformulated his vision of the Semantic Web as a "Giant Global Graph".[4] This understanding lead to a significant change in the architecture of the WWW: Now the W3C considers RDF *graphs* as basic data structures for representing information.[5] This paradigmatic shift clearly hints to the importance of graph-based logics within the Semantic Web framework.

A more sophisticated level in the W3C stack (the hierarchy of W3C description languages) is the level of ontologies. The W3C recommendation is OWL (Web Ontology Language), which has recently been extended to OWL 2.0. The

[3] `http://www.cmis.brighton.ac.uk/research/vmg`
[4] `http://dig.csail.mit.edu/breadcrumbs/node/215` created 2007/11
[5] `http://www.w3.org/Consortium/technology`, created 2008/01

formal background of OWL and OWL 2.0 is the family of Description Logics (DLs, see [1]). DLs are a common family of knowledge representation formalisms tailored to express knowledge about concepts and concept hierarchies. They include sound and complete decision procedures for reasoning about such knowledge.

The formal notation of DLs has the flavor of a variable-free first order logic. In fact, DLs correspond to (decidable) fragments of first order logic, and they have a well-defined, formal syntax, a semantics in the form of Tarski-style models, and sound and complete calculi (e.g. based on Tableaux-algorithms). It is often emphasized that DLs offer, in contrast to other knowledge representation languages, sound, complete and (empirically) practically useful reasoning services.

The fact that the notation of DLs is variable-free makes them easier to comprehend than the common first order logic formulas which include 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. In [55], the introduction to the *Description Logic Handbook*, Nardi and Brachman write 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."

For RDF, mathematical elaborations based on graph theory have been developed. They include Tarski-style semantics as well as sound and complete calculi, latter either based on "projections" (see [4, 5]) or on diagrammatic rules (see [19]). That is, RDF is developed as a fully-fledged diagrammatic logic. A first attempt at a diagrammatic representation for DL is can be found in [28], where Gaines elaborates a graph-based representation for the textual DL CLASSIC. 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, [7] provides a UML-based, diagrammatic representation for OWL DL. In these approaches, the focus is on a graphical *representation* of DL, however, as often emphasized, *reasoning* is seen as a distinguishing feature of DL and such reasoning is not supported diagrammatically by that treatment. A first step towards developing DLs as fully fledged diagrammatic logics, based on Peirce's Existential Graphs, has been carried out for the DL $\mathcal{ALC}$ (the smallest propositionally closed DL) in [27]. Research is in progress to extend this approach to more expressive DLs.

3 Historical Systems

In this section, we shortly discuss the historical background of DRSs, namely Euler circles (a.k.a. Euler diagrams) [24], Venn diagrams [77], Venn-Peirce diagrams [59] and Peirce's existential graphs [33, 60, 31].

An Euler diagram is a finite collection of closed curves drawn in the plane. For example, in Fig 3, d_1 is an Euler diagram which expresses that nothing is both a car and a van. Venn diagrams differ from Euler circles in the respect that the curves are drawn in a way that all possible set combinations are shown in the diagram, and shadings are used to show that certain set combinations must be empty. So the Venn diagram d_2 expresses the same information as d_1. Finally, in Venn-Peirce diagrams, o-signs are utilized to assert the emptiness of a set. More importantly, Peirce also introduced $\otimes$-signs which denote elements so that –in contrast to Euler Circles or Venn-diagrams– now the non-emptiness of sets can be explicitely expressed. These signs can be assembled with lines to sequences, and the lines are read in a disjunctive manner. If we consider the diagram d_3, the outer spider asserts that the set $cars \cap \overline{Vans}$ is non-empty (denoted by the two $\otimes$-signs) or that the set $Cars \cap Vans \cap \overline{Bikes}$ is empty (denoted by the o-sign). So the d_3 is a diagrammatic representation of the formula

$$(Cars \cap \overline{Vans} \neq \emptyset \vee Cars \cap Vans \cap \overline{Bikes} = \emptyset) \wedge$$
$$(Cars \cap \overline{Vans} \neq \emptyset \vee Cars \cap Vans \cap Bikes = \emptyset).$$

which expresses that either $Cars \cap Vans = \emptyset$ or $Cars \cap \overline{Vans} \neq \emptyset$.

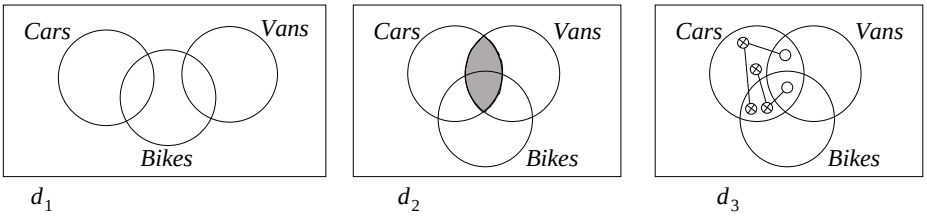

Fig. 3. An Euler diagram, a Venn diagram, and a Venn-Peirce diagram

Existential graphs (existential graphs) are a different diagrammatic system with a focus on representing and reasoning with relations. The system of existential graphs is divided into three parts: Alpha, Beta and Gamma, which presuppose and are built upon each other. Alpha corresponds to propositional logic, Beta corresponds to first order logic, and 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 paper, only Alpha and Beta are introduced.

The existential graphs of Alpha consist only of two different syntactical entities: (atomic) propositions, and so-called *cuts* which are represented by fine-drawn, closed, doublepoint-free curves. Essentially, writing different graphs on the plane expresses their conjunction, and enclosing a graph by a cut denotes its negation. Below, two examples of Alpha graphs are given. In the left graph, the propositions 'it rains', 'it is stormy' and 'it is cold' are written side by side,

thus the graph means ‘it rains and it is stormy and it is cold‘. The right graph has the meaning ‘it is not true that it rains and it is stormy and that it is not cold’, i.e. ‘if it rains and if its stormy, then it is cold’.

Fig. 4. Two Alpha graphs

If we go from the Alpha part of existential graphs to the Beta part, predicate names of arbitrary arity may be used, and a new sign, the LINE OF IDENTITY, is introduced. Lines of identity are used to denote both the existence of objects and the identity between objects. They can be connected to networks. The meaning of the Beta graphs in Fig. 5 are ‘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’.

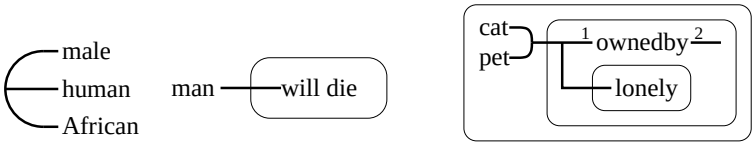

Fig. 5. Three Beta graphs

4 Seminal Work on Historical Systems

In this section, we provide an overview on important seminal work which aims at elaborating the historical systems of the last section.

4.1 Shin’s Extended Venn Diagrams

Certainly a landmark work on diagrammatic reasoning is Shin’s elaboration of Venn-Peirce-diagrams in [67]. In fact, the main point of [67] is to argue that developing a diagrammatic reasoning system which possesses the rigor of formal mathematical logic is possible; the elaboration of Venn-Peirce-diagrams can be seen as a proof of concept for her approach.

We shorty discuss Shin’s so called *Venn-II system*. Essentially, Venn-II is based on Venn diagrams where Peirce’s $\otimes$-sequences are added and where diagrams can be taken in disjunction. The Venn-II diagram in Fig. 6 expresses the

same as the Venn-Peirce diagram d_3 in Fig. 3; the line connecting the two boxes represents disjunction (similar to the lines in $\otimes$-sequences).

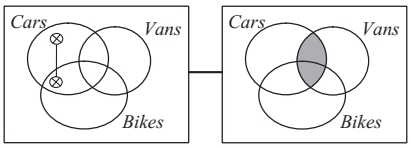

Fig. 6. A Venn-II diagram

The semantics are formalized in much the same way as traditional approaches. Shin defines *set assignments* which are analogous to structures and then specifies conditions under which a set assignment satisfies a diagram; see [67] for more details. In [67], ten reasoning rules are defined for Venn-II and shown to form a sound and complete set.

Shin shows that Venn-II is equivalent in expressive power to Monadic First Order Logic (MFOL) without equality. This is a noteworthy result: Though it is straightforward to convert Venn-II diagrams into MFOL sentences; the converse requires some effort as there is not a natural mapping from arbitrary MFOL sentences to Venn-II diagrams.

Shin claims that her system is formally rigorous, but a closer observation reveals that this claim cannot be sustained. The main reason is that Shin defined the syntax of Venn-II at the concrete (drawn) diagram level. This resulted in a lack of mathematical preciseness in her work, leading to unclear definitions and errors in proofs (see [23]). Of course, this lack of rigor should not detract from the importance of Shin's work because she laid the essential foundations for the acceptance of diagrams as formal tools.

4.2 Seminal Work on Existential Graphs

The main treatises on existential graphs are probably the books of Zeman [80], Roberts [65] and Shin [68]). Each of this books focuses on different aspects of existential graphs.

Probably the most prominent book on existential graphs is D. Robert's 'The Existential Graphs of Charles S. Peirce'. This book offers the most comprehensive description of the whole system of existential graphs –Alpha, Beta and Gamma– and its genesis. Particularly, Gamma is described to a large degree. Robert's treatise is definitely an outstanding work. However, from a mathematical point of view, this book is clearly insufficient. Roberts does not provide any (technical or mathematical) definitions for existential graphs, neither their syntax, semantic, nor inference rules, and he relies solely on the graphical representations of graphs.

In contrast to Roberts, J. J. Zeman's book 'The Graphical Logic of C. S. Peirce' is, from a mathematical point of view, the best of the books which are here discussed. Zeman provides a mathematical elaboration of Alpha, Beta, and

the part Gamma which extend Beta by adding the broken cut. Whereas Roberts solely relies on the graphical representations of graphs, Zeman defines existential graphs inductively as abstract structures, i.e. in fact as sequences of signs. Like in the other treatises, Zeman does not provide a mathematical, extensional semantics for Peirce's graphs. Instead of that, he defines mappings between the systems Alpha, Beta, Gamma and appropriate systems of propositional, first order, and modal logic. These translations from graphs to symbolic are correct, but they are arguably very technical and clumsy. Even quite simple existential graphs are translated to rather complex first order logic-formulas.

Sun Joo Shin's book 'The Iconic Logic of Peirce's Graphs' discusses only Alpha and Beta. Her interest in existential graphs is philosophically driven, and she uses existential graphs as a case study for her goal to provide a formal approach to diagrammatic reasoning. As the title of her book suggests, she focuses on the diagrammatic aspects, particularly the iconicity, of existential graphs. She compares symbolic and diagrammatic approaches to mathematical logic and works out that the *long-standing prejudice against non-symbolic representation in logic, mathematics, and computer science* is due to the fact that diagrammatic systems are evaluated in terms of symbolic systems. Then, based on her arguments, she reconsiders Alpha and Beta from iconic aspects and rewrites the reading algorithms, that is, the translations from graphs to symbolic logic, and the transformation rules in order to improve their iconicity and naturalness. mathematical preciseness. Similar to her approach for Venn diagrams, Shin only uses the graphical representation of existential graphs (which is quite surprising, as Shin elaborates carefully the semiotic aspects of existential graphs), which again results in a lack of mathematical preciseness. Particularly, Shin's definitions (and later on, theorems and proofs) cannot be considered to be mathematical. This leads to a mistake in her reading algorithm, and –even worse– some of her newly implemented transformation rules are not sound [18]. Finally, Shin does not provide an extensional semantics for Peirce's graphs: her reading algorithms are translations to symbolic logic, thus translations from one formal system to another.

5 Methodologies for Formalizing Diagrams

In the previous sections on seminal work on the historical systems, we already addressed that from a mathematical point of view, the elaborations of the historical systems are not sufficient due to a lack of formal preciseness. The formal shortcomings of the seminal works are mainly due to the fact that a general methodology for a formal elaboration of diagrammatic logics was missing. To put it more precisely: A thorough scrutiny on how to deal with diagrams (instead of symbolic notations like formulas) was not carried out.

In order to elaborate diagrammatic logics in a solid formal manner, it is crucial to note that we deal with two different entities: A mathematical structure and its diagrammatic representation. In any diagrammatic representation of a mathematical structure, we have to disregard certain graphical properties of the

diagrams, while other properties are important. This shall be exemplified with the following diagrams of Alpha existential graphs in Fig. 7.

Fig. 7. Three diagrams of one Alpha graph

The shape of the cuts (negation ovals) or the place of propositional variables and other cuts in the area of a given cut has no significance, thus all diagrams convey the same meaning. They can be read as 'it is not true that A and B, but not C hold', i.e., 'A and B imply C'. Peirce did not understand EGs as graphical entities at all. For him, the three diagrams are not *different graphs* with the same meaning, but *different representations*, i.e., diagrams, of the same graph. This is a crucial distinction, which obviously corresponds to the distinction between *types* (graphs) and *tokens* (graph replicas), as it is known from philosophy. The type-token issue if far from being settled; nonetheless, this important distinction helps us to draw the following conclusion: In any elaboration of a DRS, the diagrams should not be defined as graphical entities. Instead, we need a definition of mathematical structures which encompass exactly the facts which are represented in the diagrams, and the diagrams should be understood as (mere) representations of these structures.

Two thorough discussions on the type-token-issue for DRSs can be found in [34, 17]. Both papers argue that it is essential to provide mathematical definitions for the type-level (called *abstract syntax* in [34]). For the token-level (called *concrete syntax* in [34]), the papers come to different conclusions: Whereas [34] argues that the token-level as well as the relationship between these two levels has to be formally captured as well, [17] argues that this is in fact not needed. For the purpose of this paper, this difference has no significance. The crucial point here is the following: In symbolic approaches to formal logic, once we have chosen a formula –i.e. a sequence of signs– is chosen, the representation of the formula is uniquely given. So in symbolic logic, we have only one level of representation to deal with. In graph-based DRSs on the other hand, a given graph can have very different diagrammatic representations. For example, in Fig. 7 we have different representations of the same Alpha graph. So, for representing information, symbolic and linear notions of logic have a **one layer architecture**, and diagrammatic logic systems have a **two layer architecture**.

The general advantages of this two layer structure for the pragmatics of diagrams is already discussed to a large extent [49, 66, 6, 68, 57]. It is usually argued that the additional diagrammatic layer, often referred to as 'secondary notation', provides the essential means to improve the pragmatics of a representation system. As Oberlander writes in [57]: "secondary notation is the very stuff of graphical pragmatics–meaningful structures which go beyond the plain semantics of the system."

6 Contemporary Systems

The last two decades witness the rise of formal diagrammatic reasoning systems in two respects: First, more expressive diagrammatic reasoning systems have emerged and, more importantly, are formalized in a mathematical precise manner. Second, for some of these logics, computer programs have been developed and applied to various settings. This section summarizes some recent advances in the field.

6.1 Spider and Constraint Diagrams

Spider diagrams (SDs) and constraint diagrams (CDs) are a DRS based on Euler circles and Venn-Peirce diagrams. They are elaborated as abstract mathematical structures, including extensional semantics and inference rules. Their development is mainly driven by the Visual Modeling Group in Brighton.

Spider Diagrams Spider diagrams combine features of Venn diagrams and the more user friendly Euler diagrams. They can be thought of as extending Venn-II diagrams. Various different systems exist today, for example [35, 36, 38, 40, 74]. In Fig.8, two examples of so-called *unary* SDs are depicted.

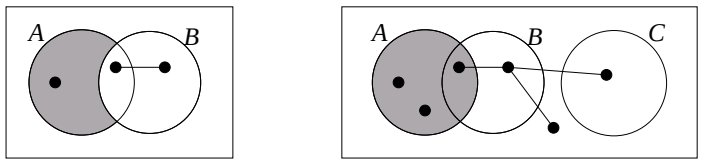

Fig. 8. Two spider diagrams

The left diagram contains two *existential spiders*. Each spider denotes a uniquely given object (i.e., different spiders necessarily denote different objects). In contrast to Venn-Peirce diagrams, shading a region does not necessarily mean the corresponding set is empty. Instead, 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 right diagram, 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, v and w (represented by the three spiders) such that $u, v \in A$ and $w \notin A - B$. Further, due to its shading, the set $A - B$ must not contain any other elements than u and v, i.e. it contains

[6] The usage of disjoint features in Euler circles has a drawback: not all abstract SDs are drawable (see [75]).

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.

Unary SDs can be propositionally combined with the logical operators $\sqcap$ ('and') and $\sqcup$ ('or'). In Fig. 9 an SD which uses these conjunctors is shown. It has been shown that SDs are equivalent in expressive power to MFOL with equality [72]; thus they have a higher expressiveness than Venn-II.

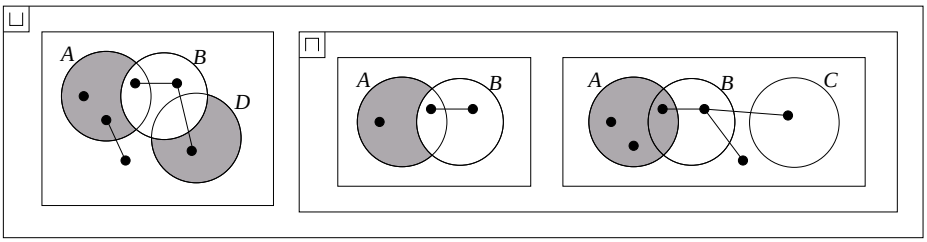

Fig. 9. A compound spider diagram

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 spiders in SDs are 'existential spiders' as they can be read as existentially quantified objects. CONSTRAINT DIAGRAMS are essentially an extension of SDs with UNIVERSAL SPIDERS (quantifiers) and ARROWS which represent binary relations. A full constraint notation was introduced by Kent [42] 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. For example, in [26], 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 [73] but compared to Kent's approach it has several limitations.

An example for a constraint diagram was already given in Fig. 2. In the formal approach to constraint diagrams, we have both existential and universal spiders, which renders the formalization of the diagrams difficult, as there is no natural order placed on the quantifiers. This difficulty is overcome by augmenting each diagram with a *reading tree*; the formalization of constraint diagrams can be found in [25]. An example can be seen in Fig. 10. This diagram expresses that *Book*, *Title* and *Author* are pairwise disjoint, *Fiction* and *NonFiction* form a partition of *Book*, and finally every book x has a unique name which is its title and x has at least one main author.

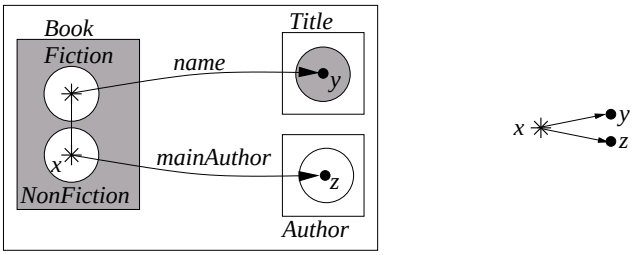

Fig. 10. A constraint diagram with a reading tree

Applications For Euler diagrams, the Visual Modelling Group has developed a diagrammatic theorem prover called Edith.[7] Edith automates the search for Euler diagram proofs using a variety of reasoning rule sets and heuristics [71]. For a given rule set, Edith finds a shortest proof. In Edith, users can create diagrams, apply rules to write a proof and ask Edith to seek a proof from one diagram to another. With regard to applications, SDs have been used to detect component failures in safety critical hardware [12] and (slight variations of them) to represent non-hierarchical file systems [21] and for viewing clusters which contain concepts from multiple ontologies [32].

As mentioned in section 2, one area that could benefit from the development of diagrammatic logics is software engineering. Constraint diagrams were designed with this application area in mind and extend the spider diagram language. Constraint diagrams have been used in a variety of areas including formal object oriented specification [37, 44] and a visual semantic web editing environment [51, 81]. Prototype tools to support their use are available from [64].

6.2 Conceptual Graphs

John Sowa's conceptual graphs (CGs) are based on Peirce's existential graphs. Sowa writes that they are an *extension of existential graphs with features adopted from linguistics and AI. The purpose of the system is to express meaning in a form that is logically precise, humanly readable, and computationally tractable.* CGs are designed to be used in fields like software specification and modeling, knowledge representation, natural language generation and information extraction, and these fields have to cope with problems of implementational, mathematical, linguistic and even philosophical nature. In CGs, we can diagrammatically represent various entities and logical constructors, like concepts and relations, individuals, quantifiers, conjunction, different levels of negations, contexts, etc.

In Fig. 11, three well-known examples of CGs are provided. The graph d_1 has the meaning that Yoyo is a cat and Yoyo is on some (unnamed) mat. In d_2, so-called *contexts* are used. The meaning of this graph it that the person Tom believes the proposition that the person Mary wants the situation that Mary

[7] `http://www.cmis.brighton.ac.uk/research/vmg/autoreas.htm`

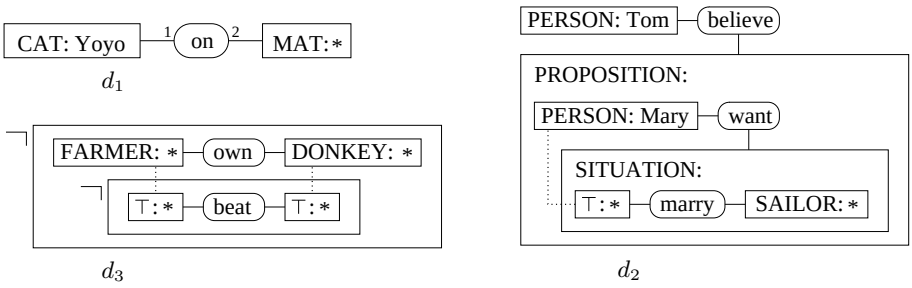

Fig. 11. Three conceptual graphs

marries a sailor. In short: The person Tom believes that the person Mary wants to marry a sailor. Finally, in d_3, special contexts denoting negation are used. The device of two nested negation contexts can be understood as an implication. So the meaning of the graph is 'if a farmer owns a donkey, then he beats it'.

Due to the complexity of the system of CGs, it is nearly impossible and perhaps not even desirable to consider the overall system as an approach to formal logic. In fact, the whole system of CGs, as described by Sowa, goes beyond FOL. It can be argued that Sowa's original system does not fully meet the requirements of a formal logic system [20], but several fragments of CGs have been elaborated in a mathematically rigorous manner.

The most prominent fragment are *simple* CGs, where no sort of context or negation is used. For this fragment, there does exist a variety of different formalizations with different notations which only differ in details. A comprehensive comparison of the different approaches can be found in [39]. For all these approaches, a formal syntax, semantics (either via a translation of CGs to FOL or via direct model-theoretic approach) and reasoning facilities are provided. Examples are the works of Prediger and Dau, where the reasoning facilities come in form of graph transformation rules [61, 62, 9, 52, 15], works of Chein/Mugnier et al, where the entailment between CGs are described by meaning-preserving graph homomorphisms called *projections* [8, 52, 11], or works where so-called *standard-models* are considered [61, 62, 15]. One can even express "if-then"-statements with simple CGs, at is has been discussed in [2, 52, 3].

Simple CGs with contexts go beyond FOL. For these graphs, there exist different formalizations as well. We want to mention the work of Prediger [61, 63], where the semantics for the graphs is based on triadic Formal Concept Analysis. Prediger provides a sound and complete set of rules for these graphs. A different approach has been taken by Simonet in [69, 10] by translating the graphs to FOL-formulas, where the contexts are modeled by assigning to each concept box an additional argument which models the nesting of the box. In this approach, the notion of projections is accordingly extended.

For CG without contexts, different kinds of negation have been added. First we have to mention approaches where only relations, but not whole subgraphs, can be negated. A first step has been taken bay Kerdiles in [43]. Kerdiles ex-

tends the projections of simple CGs to simple CGs with atomic negation. This approach has been extended in [50, 54], where for this framework, different logical approaches –classical logic, intuitionistic logic, logic with a closed-world semantics– are elaborated and compared. Finally, Klinger provides in [45–47] a different FCA-based approach which extends Predigers work to atomic negation.

Finally, we can consider CGs with full negation, where whole subgraphs can be negated. Kerdiles considers in [43] full negation as well. A comprehensive approach to add full negation to CGs is provided by Dau in [13, 14, 16]. In [13], Dau argues that to express full negation to CGs, a new syntactical entity to express negation has to be added, and he suggests to use the negation ovals of Peirce's existential graphs. These CGs with cuts are equivalent to full FOL with equality and it is equipped with a sound and complete set of graph transformation rules.

CGs have been implemented in a variety of computer programs. Two comprehensive frameworks are provided by Amine[8] and Cogitant[9]. Amine is a java open source platform for the development of intelligent systems and multi-agent systems covering several topics of the theory of CGs and AI. Cogitant is a set of C++-classes enabling to easily handle CGs as well as other objects of the CG-framework like the taxonomies of concepts and relations as well as rules.

7 Discussion

Steps like realizing the importance of the distinction between abstract and concrete syntax, the development of sound and complete diagrammatic reasoning systems or the development of DRS applications show that the research field of DRSs has made significant progress in the last decade, but there is still much more to be done.

Diagrammatic representations of information is investigated from various perspectives to a large extent, and their usability is sustained both by empirical and theoretical research. Anyhow, for the particular field of DRSs, this research has to be conducted to investigate the claim that the diagrams of DRSs are from a usability point of view superior to symbolic notations. This research must not be restricted to the diagrams: A distinguishing feature of DRSs is the provision of diagrammatic *reasoning* facilities, which thus have to be investigated as well.

It might be doubted that diagrams are *generally* easier to understand than symbolic notations. Instead, though diagrams certainly provide very effective means of displaying some information (like subset relationships), for other kinds of statements symbolic notations might turn out to be better suited. For this reason, developing heterogeneous or hybrid notations that incorporate both symbolic and diagrammatic parts is a promising approach. There has already some research been conducted in this direction, e.g. [30, 76]. In fact, when a formal language is developed with the goal of a high usability, the design of the language depends on the requirements of intended application areas. This might render finding the right language difficult.

[8] `http://amine-platform.sourceforge.net`
[9] `http://cogitant.sourceforge.net`

The syntax, semantics and reasoning facilities for DRSs take place on the abstract level. In symbolic notations, there does not exist a distinction between the abstract and concrete level, which leads to straightforward ways of representing statements in these notations. For DRSs, the situation is different. For a given statement on the abstract level, we have to find a convenient concrete representation. In other words: The automatic drawing of diagrams is a core research challenge for developing DRSs applications.

Finally, research on theorem provers for DRSs is in its infancy. There is certainly a need for sophisticated theorem provers which both work in an efficient way (though it is not a goal to outperform existing theorem provers for symbolic notations) and which produce proofs where the steps of the proofs are easily understood by users. Again, such theorem provers are needed to support the usability claim of diagrams.

The research questions can only be addressed by a joint effort of researchers from different fields like mathematicians, computer scientists, (cognitive) psychologists, or even designers. Thus it will be challenging to find appropriate answers. Anyhow, as the rise of diagrammatic representations in several computer science areas like software engineering, knowledge representation or semantic web shows, solving these questions is of highly practical interest.

Acknowledgment The author wants to thank Gem Stapleton from the Visual Modelling Group for her valuable help and provision of diagrams.

References

1. 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.
2. J.-F. Baget. A simulation of co-identity with rules in simple and nested graphs. In Wiliam Tepfenhart and W. Cyre, editors, *Conceptual Structures: Standards and Practices*, volume 1640 of *LNAI*, pages 442–455. Springer, Berlin – Heidelberg – New York, 1999.
3. J.-F. Baget and M.-L. Mugnier. Extensions of Simple Conceptual Graphs: The Complexity of Rules and Constraints. *JAIR*, 16:425–465, 2002.
4. 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.
5. 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.
6. Shai Berger. Studies on the uses and usefulness of diagrams. Master's thesis, ILLC, University of Amsterdam, 2000.
7. 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.

8. M. Chein and M.-L. Mugnier. Conceptual Graphs: Fundamental Notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.
9. M. Chein and M.-L. Mugnier. Conceptual graphs are also graphs. Technical report, LIRMM, Université Montpellier II, 1995. Rapport de Recherche 95003.
10. M. Chein, M.-L. Mugnier, and G. Simonet. Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*. Morgan Kaufmann, 1998. Revised version available at http://www.lirmm.fr/~mugnier/.
11. Michel Chein and Marie-Laure Mugnier. Concept types and coreference in simple conceptual graphs. In Wolff et al. [79], pages 303–318.
12. R.P. Clark. Failure mode modular de-composition using spider diagrams. In *Proceedings of Euler Diagrams 2004*, volume 134 of *Electronic Notes in Theoretical Computer Science*, pages 19–31, 2005.
13. Frithjof Dau. Negations in simple concept graphs. In Ganter and Mineau [29], pages 263–276.
14. Frithjof Dau. Concept graphs and predicate logic. In Delugach and Stumme [22], pages 72–86.
15. 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 [16] as well.
16. 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.
17. Frithjof Dau. Types and tokens for logic with diagrams: A mathematical approach. In Wolff et al. [79], pages 62–93.
18. 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.
19. 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.
20. Frithjof Dau. Formal, diagrammatic logic with conceptual graphs. In Pascal Hitzler and Henrik Scharfe, editors, *Conceptual structures in Practice*. CRC Press (Chapman and Hall/Taylor & Francis Group, 2009.
21. R. DeChiara, U. Erra, and V. Scarano. VennFS: A Venn diagram file manager. In *Proceedings of Information Visualisation*, pages 120–126. IEEE Computer Society, 2003.
22. Harry S Delugach and Gerd Stumme, editors. *Conceptual Structures: Broadening the Base*, volume 2120 of *LNAI*, Stanford, USA, July,, 2001. Springer, Berlin – Heidelberg – New York.
23. P. Scotto di Luzio. Patching up a logic of Venn diagrams. In *Proceedings 6th CSLI Workshop on Logic, Language and Computation*. CSLI Publications, 2000.
24. L. Euler. Lettres a une princesse dallemagne sur divers sujets de physique et de philosophie. *Letters*, 2:102–108, 1775. Berne, Socit Typographique.
25. A. Fish, J. Flower, and J. Howse. The semantics of augmented constraint diagrams. *Journal of Visual Languages and Computing*, 16:541–573, 2005.

26. Andrew Fish, Jean Flower, and John Howse. The semantics of augmented constraint diagrams. *Journal of Visual Languages and Computing*, 16:541–573, 2005.
27. Peter Eklund Frithjof Dau. A diagrammatic reasoning system for the description logic $\mathcal{ALC}$. *Journal of Visual Languages and Computing*, 2008. To be published.
28. Brian R. Gaines. An interactive visual language for term subsumption languages. In *IJCAI*, pages 817–823, 1991.
29. Bernhard Ganter and Guy W. Mineau, editors. *Conceptual Structures: Logical, Linguistic and Computational Issues*, volume 1867 of *LNAI*, Darmstadt, Germany, 2000. Springer, Berlin – Heidelberg – New York.
30. E. Hammer. *Logic and Visual Information*. CSLI Publications, 1995.
31. Charles Hartshorne, Paul Weiss, and A. W. Burks, editors. *Collected Papers of Charles Sanders Peirce*, Cambridge, Massachusetts, 1931–1935. Harvard University Press.
32. P. Hayes, T. Eskridge, R. Saavedra, T. Reichherzer, M. Mehrotra, and D. Bobrovnikoff. Collaborative knowledge capture in ontologies. In *Proceedings of the 3rd International Conference on Knowledge Capture*, pages 99–106, 2005.
33. Nathan Houser, Don D. Roberts, and James Van Evra, editors. *Studies in the Logic of Charles Sanders Peirce*. Indiana University Press, 1997.
34. 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.
35. J. Howse, F. Molina, and J. Taylor. On the completeness and expressiveness of spider diagram systems. In *Proceedings of 1st International Conference on the Theory and Application of Diagrams*, pages 26–41, Edinburgh, UK, September 2000. Springer.
36. J. Howse, F. Molina, J. Taylor, S. Kent, and J. Gil. Spider diagrams: A diagrammatic reasoning system. *Journal of Visual Languages and Computing*, 12(3):299–324, June 2001.
37. J. Howse and S. Schuman. Precise visual modelling. *Journal of Software and Systems Modeling*, 4:310–325, 2005.
38. J. Howse, G. Stapleton, and J. Taylor. Spider diagrams. *LMS Journal of Computation and Mathematics*, 8:145–194, 2005.
39. Michel Chein Jean-Pierre Aubert, Jean-Francois Baget. Simple conceptual graphs and simple concept graphs. In Øhrstrøm et al. [58], pages 87–101.
40. C. John. Reasoning with projected contours. In *Proceedings of 3rd International Conference on the Theory and Application of Diagrams*, volume 2980 of *LNAI*, pages 147–150, Cambridge, UK, 2004. Springer.
41. S. Kent. Constraint diagrams: Visualizing invariants in object oriented modelling. In *Proceedings of OOPSLA97*, pages 327–341. ACM Press, October 1997.
42. Stuart Kent. Constraint diagrams: Visualizing assertions in object-oriented models. In *OOPSLA*, pages 327–341. ACM Press, 1997.
43. G. N. Kerdiles. *Saying it with Pictures: A Logical Landscape of Conceptual Graphs*, volume DS 2001-09. ILLC Dissertation Series, 2001.
44. S.-K. Kim and D. Carrington. Visualization of formal specifications. In *6th Aisa Pacific Software Engineering Conference*, pages 102–109, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.
45. J. Klinger. Simple semiconcept graphs: A boolean logic approach. In Delugach and Stumme [22].

46. J. Klinger. Semiconcept graphs with variables. In 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.
47. J. Klinger. *The Logic System of Protoconcept Graphs*. Shaker Verlag, Aachen, 2005. Dissertation, Darmstadt University of Technology.
48. 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.
49. Jill H. Larkin and Herbert A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–100, 1987.
50. Michel Leclère and Marie-Laure Mugnier. Simple conceptual graphs with atomic negation and difference. In Øhrstrøm et al. [58], pages 331–345.
51. J. Lovdahl. *Towards a Visual Editing Environment for the Languages of the Semantic Web*. PhD thesis, Linkoping University, 2002.
52. M.-L. Mugnier. Knowledge representation and reasonings based on graph homomophism. In Ganter and Mineau [29], pages 172–192.
53. Marie-Laure Mugnier and Michel Chein, editors. *Conceptual Structures: Theory, Tools and Applications, 6th International Conference on Conceptual Structures, ICCS '98, Montpellier, France, August 10-12, 1998, Proceedings*, volume 1453 of *LNAI*. Springer, Berlin – Heidelberg – New York, 1998.
54. Marie-Laure Mugnier and Michel Leclère. On querying simple conceptual graphs with negation. *Data Knowl. Eng.*, 60(3):468–493, 2007.
55. Daniele Nardi and Ronald J. Brachman. An introduction to description logics. In Baader et al. [1], pages 1–40.
56. 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.
57. Jon Oberlander. Grice for graphics: pragmatic implicature in network diagrams. *Information Design Journal*, 8(6):163—179, 1996.
58. 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.
59. C. Peirce. *Collected Papers*, volume 4. Harvard University Press, 1933.
60. Charles Sanders Peirce. Reasoning and the logic of things. In K. L. Kremer and H. Putnam, editors, *The Cambridge Conferences Lectures of 1898*. Harvard Univ. Press, Cambridge, 1992.
61. Susanne Prediger. *Kontextuelle Urteilslogik mit Begriffsgraphen – Ein Beitrag zur Restrukturierung der Mathematischen Logik*. Shaker Verlag, Aachen, 1998. Dissertation, Darmstadt University of Technology.
62. Susanne Prediger. Simple concept graphs: A logic approach. In Mugnier and Chein [53], pages 225–239.
63. Susanne Prediger. Nested concept graphs and triadic power context families: A situation–based contextual approach. In Ganter and Mineau [29], pages 263–276.
64. Reasoning with Diagrams. http://www.cs.kent.ac.uk/projects/rwd/, 2006.
65. Don D. Roberts. *The Existential Graphs of Charles S. Peirce*. Mouton, The Hague, Paris, 1973.
66. 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`.

67. S.-J. Shin. *The Logical Status of Diagrams*. Cambridge University Press, 1994.
68. Sun-Joo Shin. *The Iconic Logic of Peirce's Graphs*. Bradford Book, Massachusetts, 2002.
69. G. Simonet. Two fol-semantics for simple and nested conceptual graphs. In Mugnier and Chein [53], pages 240–254.
70. John F. Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley, Reading, Mass., 1984.
71. G. Stapleton, J. Masthoff, J. Flower, A. Fish, and J. Southern. Automated theorem proving in Euler diagrams systems. *Submitted to Journal of Automated Reasoning*, 2005.
72. G. Stapleton, S. Thompson, J. Howse, and J. Taylor. The expressiveness of spider diagrams. *Journal of Logic and Computation*, 14(6):857–880, December 2004.
73. 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`.
74. Gem Stapleton and John Howse. Enhancing the expressiveness of spider diagram systems. In *Accepted for Distributed Multimedia Systems, International Workshop on Visual Languages and Computings*, Grand Canyon, USA, 2006. Knowledge Systems Institute.
75. 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.
76. N. Swoboda and G. Allwein. Heterogeneous reasoning with Euler/Venn diagrams containing named constants and FOL. In *Proceedings of Euler Diagrams 2004*, volume 134 of *ENTCS*. Elsevier Science, 2005.
77. J. Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *Phil.Mag*, 1880.
78. Rudolf Wille. Restructuring mathematical logic: An approach based on peirce's pragmatism. In A. Ursini and P. Agliano, editors, *Logic and Algebra*, pages 267–281. Marcel Dekker, New York, 1996.
79. 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.
80. 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/`.
81. Y. Zhao and J. Lövdahl. A reuse based method of developing the ontology for e-procurement. In *Proceedings of the Nordic Confernce on Web Services*, pages 101–112, 2003.